



# Ericsson Catalog Manager and Ericsson Order Care

Realize Higher Consistency for Faster Time-to-Revenue

---

Velocity Studio User Guide



© Ericsson AB 2014

All rights reserved. The information in this document is the property of Ericsson. Except as specifically authorized in writing by Ericsson, the receiver of this document shall keep the information contained herein confidential and shall protect the same in whole or in part from disclosure and dissemination to third parties. Disclosure and disseminations to the receiver's employees shall only be made on a strict need to know basis.

## **The Concept of Metadata**

Applications are not written in Java, C#, C++ or any other third generation language. Instead they are non-procedural specifications of the application logic; look and feel generated in an XML-based language called Application Markup Language, or AML. Historically, to make the concept of AML more easily understandable, we refer to the AML specifications as metadata -- information that describes the data. Although not entirely correct in the given context, the term metadata gained popularity with users. We found it useful because it properly conveys the difference between the traditional products written in 3GL and the non-procedural interpretive applications. In this document we will be using the term metadata as a synonym to AML specification - an open XML based document that describes an order entry/order management application.

AML specifications (or metadata) are not created as text files using a general purpose XML editor. Instead, the product provides a powerful point-and-click graphic tool called the Velocity Studio that creates and modifies the metadata.

# Managing Metadata

---

There are two types of metadata:

1. **Application metadata** - metadata that is written for describing an application, and may include many product metadata.
2. **Product metadata** - also known as a library, it is metadata that is generalized and can be referenced by an application metadata to accelerate application development. These metadata can also be customized or extended by application metadata, and can reference other product metadata as well. However, it cannot be executed independently without application metadata.

## Application Metadata

During development and testing phases, Application Metadata is stored as folders and files in a local **Project Directory**. In general, each metadata object is stored as a stand-alone XML file in a folder that mimics the metadata hierarchy in the directory; see section [Understanding Your Metadata Files](#) for details on how metadata is structured. A source control system shall be applied to the directory to centrally store, log, revision control, and merge development work among team members.

**Note:** Application Metadata shall be managed with Velocity Studio, and not directly with XML files. An exception to this is when merging files among team members in source control system result in conflicts; in this case, proceed manual merging with care, beware of the integrity of the XML.

During metadata development and functional testing, there is no need to deploy the metadata. Simply run the metadata within Velocity Studio.

If Metadata in source format needs to be transferred (for example, send to another engineering team who does not have access to the source control system), simply zip the Project Directory with a third-party tool such as winzip before transferring.

## Product Metadata

Similar to Application Metadata, Product Metadata may consist of one or more namespaces that implement functionality; however, unlike Application Metadata, libraries are unchangeable in Velocity Studio. You can incorporate library functionality into your application, by either

1. referencing metadata objects (for example, create a menu item that shows a Document) of a library in application metadata; properties of the library metadata object is unchangeable.
2. copying metadata objects of a library and paste into application metadata. Functionality of the metadata object can then be customized accordingly. However, any future library updates by the library owner cannot be trivially incorporated into previously copied objects.
3. overriding existing functionality of library metadata objects by creating the same object type in application metadata, with the **Override** property set to the library object.

Velocity Studio has a common set of metadata libraries, available in **Library** tab of the Navigation Pane to assist your application development. They are system metadata that implements core functionality, and generalized metadata objects that serve common business objectives, such as Common Order Modelling, Order Management and Order Entry.

You can also create your own libraries to aid the distribution and organization of your application. For example, external interface integration based on technical contracts (such as WSDL) can be consumed and implemented in Velocity Studio as a project, and then use the [Build Library JAR](#) command to create the library. This library may then be re-used in your application, at the same time abstracting away interface setting details.

Product Metadata is packaged as a JAR file. To use a library, [include the JAR file](#) into your project. You can find the imported metadata in the **Library** tab, similar to Velocity Studio libraries.

## Metadata Objects Overview

Metadata consists of objects that are grouped in folders called namespaces. The metadata objects have unique names within the namespace. Namespaces are metadata objects too and must have unique names within the metadata. Although Velocity Studio does not explicitly disallow it, the names of the metadata objects should not start with the prefix cwf in any letter case, as this denotes a system-defined metadata object.

The top metadata object is called the metadata root object. Clicking this object displays four tabs that allow you to configure aspects of libraries, language and URL mapping.

Metadata objects can be grouped into wider families that include the following:

### Data Dictionaries

Metadata Object Type	Filename Prefix in Source	Description
 Data Type	<code>dtype_</code>	Atomic data objects (for example, a string or integer) that may have restrictions on the values they can hold (for example, lowest and highest number for a numeric data type) and validation rules with arbitrary complexity. The metadata data type object can specify formats that are used when displayed in the browser screen or when read and written to a file.
 Document	<code>doc_</code>	The most universally used data object that represents a flat structure of data types. The documents can be mapped to database tables as well as to text and binary files. They can also be used as input and output parameters for external interface calls or database stored procedure calls. Documents have a rich set of properties that define how they are displayed and processed by the browser.
 Data Structure	<code>dstruct_</code>	A hierarchical structure of data types, documents and Data Structures. The Data Structure can model any hierarchical structure (that is, it can represent any XML message).
 Conversion Map	<code>cmap_</code>	Velocity Studio maintains Conversion Map objects that define the conversion of the content of a Data Structure, Document or XSD Schema to another Data Structure, Document or XSD Schema.

### Finders

Metadata Object Type	Filename Prefix in Source	Description
 Document Finder	<code>findDoc_</code>	Allow searching, reading and modifying data from different data sources including databases, external system interfaces and user scripts. Finders have properties that define how they work in the browser environment - the way they communicate with the user and present the result set of objects.
 Order Finder	<code>findOrder_</code>	
 SQL Finder	<code>findSql_</code>	
 Script Finder	<code>findScript_</code>	
 Interface Finder	<code>findIFace_</code>	
 Rule Finder	<code>findRule_</code>	

### Orders

Metadata	Filename	Description

Object Type	Prefix in Source	
 Order	order_	A hierarchical structure of documents that is used to model order data. Like documents, orders have a set of properties that control the way they are displayed on the browser screen and interact with the user.
 Life Cycle	lcycle_	Define sets of allowed states and transitions between them. Lifecycle diagrams are useful when the processing of an object, such as an order, that cannot be easily described through an ordered tree of tasks (that is, as a business process) but can be defined as reactions to different events (or messages) which change states. The lifecycle diagrams are not another type of business processes. They usually are used with a generic business process that drives the state machine. In their most trivial form, the lifecycle diagrams are used to define which state transitions are valid for a given order type.

## Business Processes

Allow defining the workflow of the business processes and their participants. Participants include the groups of people for manual tasks and interface operations for automatic tasks. Velocity Studio internally uses the BPML process model to define the workflow. The BPML model is functionally equivalent to the BPEL model. It defines the process as a tree of control operations that match the operations defined in the structured programming model which includes sequence, if-then-else or switch and loop. Based on this model, processes can describe any workflow, from the simplest one to workflows with arbitrary complexity. Velocity Studio allows defining handlers that catch the exception in the process flow and run compensation flows for them.

The runtime environment maintains two types of processes, local processes that work with a particular data object, usually order, and complete when the data object reaches its final stage and global processes (also called listeners) that usually process a queue of messages or run periodic database queries. The global processes implement endless loops - they never complete.

Metadata Object Type	Filename Prefix in Source	Description
 Process	proc_	Define the workflow and the optional data objects attached to the process instance at runtime. These data objects are the process document that holds instance specific data and the process order that contains the business data for the process.
 Process Revision	pprev_	An archive copy of old Process metadata. Read-only. A Process may have many Process Revisions. They are created and retained in Velocity Studio to ensure existing process instances at runtime can continue to be executed upon Process metadata change (that is, "grand-parenting").
 Decision Tree	dtree_	Define decision flow graphically, as a tree of decision making units, which encourages a piecewise, divide-and-conquer approach to defining a decision logic. While similar to Processes, Decision Trees are to be completed instantaneously at runtime.
 Participant	part_	Group of users that have certain privileges or external system interfaces. The notion of a participant allows replacing manual tasks with automatic tasks (and vice versa) without changing the flow (business logic) of the processes. Manual participant objects have properties that control the way the tasks are distributed between the users.
 Exception	excp_	Named objects that, when thrown by the process, cause immediate transfer of the control to a place in the process where the exceptions are processed. The mechanism is equivalent to the try - catch blocks that exist in the modern programming languages.
 Signal	signal_	Named objects that allow synchronization between the runtime process instances. A process instance can send a signal to another process instance or can wait for a signal to continue its execution.
 Alert	alert_	System generated messages that are sent to designated users when a process instance, at runtime, encounters some predefined condition. The user can receive the alerts in many different ways - in the worklist, as an e-mail or as a pager call. Alerts can also be sent to an external system interface.
 Rule	rule_	Powerful set of business rules expressed in specialized rule language that can be used as finders, for data object initialization or execution of some logic. Scripts are built by a graphical point-and-click editor, not by the text editor that is used to enter and modify the scripts. Each rule may consist of many conditional instances that are performed or skipped depending on the

			content of the documents they work on.
	<b>Script</b>	<i>script_</i>	Global JavaScript functions that may access and manipulate the internal object model. Almost all metadata objects allow some scripting (that is, the behaviour of the object can be customized by short scripts). These scripts, called object scripts, may call the logic defined in the metadata objects of global script. Global scripts are reusable algorithms while the local object scripts are tightly attached to a given metadata object.
	<b>Permission</b>	<i>permMethod_</i>	Specialized Script metadata object that evaluates what actions are allowed or disallowed for a given user or group of users. Permissions are mainly used to modify the user interface. Velocity Studio allows building sophisticated permission models that depend on the user privileges, the content of the data objects and the current tasks of the business processes.

## External Services

Metadata Object Type	Filename Prefix in Source	Description
	<b>Interface</b>	<i>iface_</i> Define external system interfaces as set of operations. Each operation receives a message (document or data structure) as an input and generates a message as an output. The operations can generate an error message that indicates that the requested operation can not be performed or there have been exceptions during the execution.
	<b>External Service</b>	<i>service_</i> Define a collection of endpoints (ports) that deliver services.
	<b>Binding</b>	<i>bind_</i> Link between service ports, technology (service providers) and interfaces.

## Presentation

Define the communication between the user and a browser. Some of the already described metadata objects, such as documents, finders, or orders, have user interface objects underneath them such that they may render themselves in the browser screen and communicate with the user in a controlled way.

Metadata Object Type	Filename Prefix in Source	Description
	<b>User Interface</b>	<i>ui_</i> The User Interface controller in MVC architecture pattern that links the Model (as variables) and View (as Forms); contains Forms underneath it to present the desired view of the user.
	<b>Navigation Tree</b>	<i>tree_</i> Specialized User Interface Controller that allows the modeling and presentation of tree structure, where each node in the Tree may be bound to a metadata object such as Document or Finder to represent/display its Form.
	<b>Navigation Bar</b>	<i>navbar_</i> Specialized User Interface Controller that models and displays a list of items in a vertical or horizontal bar, where each item may be clicked to enable a user action such as showing a Form. This may be used to create multi-step wizard-like page, where the user has visualization and may traverse steps in the "wizard" via navigation bar.

## Testing

Allow developers to write tests to assert intended functionality of metadata, whether it is for the purpose of unit-testing or automating manual user-interface interactions.

Metadata Object Type	Filename Prefix in Source	Description
	<b>Test Case</b>	<i>tcase_</i> Store test functions, and corresponding setup and teardown activities. They are typically invoked with a menu action script.
	<b>Test Suite</b>	<i>tsuite_</i> Container to assemble test cases for execution.



## Understanding Your Metadata Files

---

Developing or customizing complex metadata requires the efforts of more than one developer, and thus would require merging of metadata among developers. A third-party tool, typically from revision control system, is expected to merge conflicting changes. The following explains the structure of your project directory and also the format of the contained metadata files.

File or Folder	Description
<b>metadata</b>	XML files to persist application metadata, in sub-folders of which the files' namespaces of the same names are contained. See <a href="#">Metadata Objects Overview</a> on how XML files of application metadata objects are named in this folder, and <a href="#">Navigation Pane - Metadata</a> on how actions in Velocity Studio impact files and folders in this directory.  <b>Note:</b> You must not move, copy or rename metadata files within project directory directly in file system. Use the rename, cut/copy and paste function instead in Velocity Studio. See <a href="#">Navigation Pane - Metadata</a> for details.
<b>languages</b>	XML files to persist the <a href="#">Translations imported</a> , which are language translations of visible labels, help, messages etc in runtime application for <a href="#">internationalization</a> purposes.
<b>resources</b>	Resource files stored for the project. These files can be browsed and certain functionalities are provided in the <a href="#">Navigation Pane - Resources</a> . The file <b>SystemImagesSub.txt</b> in this folder stores the list of all <a href="#">image substitutions</a> that is defined in the Resource tab.
<b>templates</b>	Library JAR files to be included into project as libraries. These files can be browsed in the <a href="#">Navigation Pane - Library</a>
<b>header_ConceptWaveMetadata.xml</b>	The XML file that persists the <a href="#">top-level metadata object</a> in Navigation Pane.  <b>Note:</b> The updated timestamp has been removed from the header metadata. When existing metadata is saved using a new build, the <updated> tag is removed from header_ConceptWaveMetadata.xml. The Global.getMetadataTimestamp script method is deprecated and it returns the current timestamp for backward compatibility.
<b>.settings</b>	Settings and preferences in Velocity Studio concerning this project, such as database connections and configuration <i>Node ID</i> to run the metadata within Velocity Studio. They are not related to any metadata. For revision control, you may or may not check-in this file, depending on the similarity (or not) of your team's development environment setup.
<b>.cw.inuse.lock</b>	The presence of this file indicates this project is currently opened by an instance of Velocity Studio, locking other instances of Velocity Studio from opening this project. The file contains the timestamp of the locking start. For revision control, <b>do not check-in this file</b> . If, for example, your Velocity Studio crashes, and no longer able to open the project upon restarting Velocity Studio, delete this file to unlock the problem.

## Command Line Options of Velocity Studio

---

In addition to running the Velocity Studio program, you can command Java to run *Designer.jar* to perform other actions based on parameters in command line. This table lists the capabilities and the parameters that invoke them.

Action	Parameter	Description
<a href="#">Analyze Processes</a>	<code>-analyseProcesses</code>	Compare local process metadata with deployed process metadata.
<a href="#">Start Velocity Studio</a>	(no parameter)	Start the Velocity Studio program.
<a href="#">Build Deployment JAR</a>	<code>-buildDeployment</code>	Creates a deployment JAR file that contains project metadata and required library JAR files.
<a href="#">Deploy Application</a>	<code>-deploy</code>	Deploy an application to a database.
<a href="#">Export Code Tables</a>	<code>-runscript</code>	Export the code tables without Catalog application.
<a href="#">Export Configuration</a>	<code>-configExport</code>	Export the configuration; generates an XML file representing the system configuration, without logging into the System Configuration application.
<a href="#">Export Rule Instances</a>	(no parameter)	Export rule instances to an XML file.
<a href="#">Import rule instances</a>	(no parameter)	Import rule instances from an XML file.
<a href="#">Import Configuration</a>	<code>-configImport</code>	Import the configuration including flags and settings required for the full deployment.
<a href="#">Import Code Tables</a>	<code>--runscript</code>	Import the code tables without Catalog application.
<a href="#">Migrate Application</a>	<code>-migrate5config</code>	When migrating an application, this command allows you to migrate metadata, database, and configuration settings.
<a href="#">Migrate rule instances</a>	(no parameter)	Migrate rules instances from version 4.2 to version 5.x metadata, including templates other than CW system templates, and export legacy rule instances to an XML file.
<a href="#">Package metadata</a>	<code>-package</code>	Package metadata, which is equivalent to <a href="#">Runtime &gt; Build Deployment WAR</a> command in Velocity Studio.
<a href="#">Validate Metadata</a>	<code>-runValidate</code>	Validate the metadata without running or activating.
<a href="#">Run Configuration application only</a>	<code>-runconfig</code>	Starts Configuration application without running Velocity Studio, and without any metadata.
<a href="#">Sign Licence</a>	<code>-signLicense</code>	Signs your licence file, which is used with Sentinel RMS.
<a href="#">Upgrade System</a>	(no parameter)	Generates the Update System output using a command line.

For all command lines, the following points hold true:

- Any logging messages, warnings, or errors appear in the log file in Velocity Studio (that is, `<installation_folder>\designer\env\log`).
- The `logFile` parameter, which applies to most command, allows you to specify the log file for a command line. It is an optional parameter. However, when used, it must be the first parameter.
- You can run these commands simultaneously (for example, the deploy, upgrade system, and build deployment JAR commands).
- An exit code appears after you have issued a command, indicating whether running the command was successful. The following exit codes are available:
  - 0, meaning that the command processed successfully
  - 1, meaning that the wrong command was processed

### Start Velocity Studio by Command Line

The `<installation_folder>\designer\env\startDesigner.cmd` that is invoked in Windows Start menu is of the following:

```

set CLASSPATH=C:\Program Files\CWOrderCare\designer\Designer.jar;C:\Program
Files\CWOrderCare\lib\oracle\ojdbc6.jar;C:\Program
Files\CWOrderCare\lib\oracle\orai18n.jar;C:\Program Files\CWOrderCare\lib\oracle\orai18n-
mapping.jar;C:\Program Files\CWOrderCare\lib\ilog\sdworkflowmodeler.deployed.jar;C:\Program
Files\CWOrderCare\lib\jdic\jdic.jar;
start "Service Designer" /B "C:\java\jdk1.6.0_12\bin\javaw" -Xms1024m -Xmx1024m -cp "%CLASSPATH%" -
Dcom.sun.management.jmxremote.port=10008 -Dcom.sun.management.jmxremote.authenticate=false -
Dcom.sun.management.jmxremote.ssl=false com.conceptwave.servicedesigner.ServiceDesigner

```

## JVM parameters

Parameter	Description
-Xms<number>m & -Xmx<number>m	Mandatory. Specify the amount of memory that will be used. Any metadata that is run within Velocity Studio (that is, using <b>Runtime &gt; Run</b> ) will use this memory as well.
-Dcom.conceptwave.licenseDir=<licenseDir>	Optional. Specify the directory of the license file(s). If this argument is not specified, the default is <b>\CWOrderCare\designer\env</b> .
-Dcom.sun.management.jmxremote.port=<port number> -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.password.file=<file location>	Optional. Enable JMX on the Virtual Machine. See section <a href="#">Enabling JMX</a> for details.
-Dcom.conceptwave.verbose=true	Optional. Enable any warnings during activation.
-Dcom.conceptwave.info.printActivitiesOfProcess.name=NAMESPACE:processName	Optional. When runtime starts, the system creates a file in the working directory with name NAMESPACE_processName (replaces the colon (:)) with an underscore (_)) and prints a list of all activities for this process with indexes as they appear in the database within the CWPACTIVITY table.
-Dcom.conceptwave.info.printActivitiesOfProcess.fullNames=1	Optional. When fullNames=1, full name of the activity is printed, including all parent activities. Otherwise, only the short name (that is, the activity itself) is printed.
-DstandByPE=true	Optional. Specify running a standby process engine.
-Dcom.conceptwave.verbose.file=path_name parameter	Optional. Specify the <a href="#">log file for a parameter</a> in JConsole.
-Djava.awt.headless=true	Mandatory for all commands, except for startDesigner.cmd. This option allows you to run the process engine in headless mode to perform Abstract Windowing Toolkit (AWT) operations. See <a href="#">Configuring PDF Generation</a> -

	<a href="#">Memory Considerations</a> for details.
-Djava.net.preferIPv6Addresses=true	Optional. Support for Internet Protocol Version 6 (IPv6).
-Djava.net.preferIPv4Stack=true	Optional. If AVM is installed on JBoss on Windows operating system, this parameter being set to true in standalone.conf.bat file results in a <i>No valid license</i> error if the license is based on the MAC address and the MAC address is disconnected from the LAN. To avoid the error, this parameter should be removed from the standalone.conf.bat file on Windows.

The JVM arguments must appear before the class (for example, `com.conceptwave.servicedesigner.ServiceDesigner`) in the command.

#### Designer.jar parameters

Parameter	Description
<code>&lt;metadata dir&gt;</code>	Optional. If supplied, Velocity Studio is started with the project opened as located in <code>&lt;metadata dir&gt;</code> .

The Designer arguments must appear after the class (for example, `com.conceptwave.servicedesigner.ServiceDesigner`) in the command.

## Validation of Metadata in the Command-Line Interface

Validation of metadata can be done, without running or activating, in the command-line interface. This feature is useful, where metadata requires validation in development environment (for example, before checking or merging the code to the main project branch), or as a part of the development process (for example, after code review).

**Note:** Running the *runValidate* command from a directory other than designer\env will fail.

To validate metadata, in Windows, use *runValidate.cmd*. A shortcut command can be found in **<installation\_folder>\designer\env\runValidate.cmd**. The following describes the syntax of the command-line interface and the available arguments.

```
@echo off
if ""%1""=="""" goto echoSyntax

set CWOC_CLASSPATH=C:\6\264
set CLASSPATH=%CWOC_CLASSPATH%\designer\Designer.jar;^
%CWOC_CLASSPATH%\lib\oracle\ojdbc6.jar;^
%CWOC_CLASSPATH%\lib\oracle\orai18n.jar;^
%CWOC_CLASSPATH%\lib\oracle\orai18n-mapping.jar;^
%CWOC_CLASSPATH%\lib\ilog\sdworkflowmodeler.deployed.jar;^
%CWOC_CLASSPATH%\lib\jdic\JDICplus.jar;^
%CWOC_CLASSPATH%\lib\axis2\addressing-1.41.mar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-1.5.mar;^
%CWOC_CLASSPATH%\lib\axis2\mex-1.4.1.mar;^
%CWOC_CLASSPATH%\lib\axis2\wstx-asl-3.2.4.jar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-core-1.5.jar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-policy-1.5.jar;

set JAVA_OPTS=-Xms1024m -Xmx1024m -XX:MaxPermSize=128m

"C:\java\jdk1.6.0_16\bin\java" %JAVA_OPTS% -cp "%CLASSPATH%" com.conceptwave.servicedesigner.ServiceDesigner -validate %

goto end
:echoSyntax
echo -----
echo Syntax: runValidate [-logFile logfile] metadata_path
echo Example: runValidate c:\metadata_path
echo -----
:end
```

To validate metadata, in Unix or Linux, use *runValidate.sh*. A shortcut command can be found in **<installation\_folder>\designer\env\runValidate.sh**. The following describes the syntax of the command-line interface and the available arguments.

```
#!/bin/sh
if [ $DOLLAR$# -lt 1 ] || [ $DOLLAR$# -gt 10 ]
then
echo "Syntax: ./runValidate.sh metadata_path"
echo "Example: ./runValidate.sh /home/metadata/"
exit 1
fi

"$JAVA_HOME$/bin/java" -Xms1024m -Xmx1024m -XX:MaxPermSize=128m -cp
"$USER_INSTALL_DIR$/designer/Designer.jar:$USER_INSTALL_DIR$/lib/oracle/ojdbc6.jar:$USER_INSTALL_DIR$/lib/oracle/
mapping.jar:$USER_INSTALL_DIR$/lib/ilog\sdworkflowmodeler.deployed.jar:$USER_INSTALL_DIR$/lib/jdic\JDICplus.jar:1.
41.mar:$USER_INSTALL_DIR$/lib/axis2/rampart-1.5.mar:$USER_INSTALL_DIR$/lib/axis2/mex-1.4.1.mar:$USER_
INSTALL_DIR$/lib/axis2/jaxb-xjc-2.1.6.jar:$USER_INSTALL_DIR$/lib/axis2/wstx-asl-3.2.4.jar:$USER_
INSTALL_DIR$/lib/axis2/rampart-core-1.5.jar:$USER_INSTALL_DIR$/lib/axis2/rampart-policy-1.5.jar" com.conceptwave.servicedesigner -validate $DOLLAR$*
```

The syntax used in the command line is as follows:

```
runValidate metadata_path
```

Where `metadata_path` is the file path to a metadata project folder. If a given metadata is valid, the command exits normally with no errors. Otherwise, if a given metadata or any included library is not valid, the validation errors are sent to the console and the command exits with an error status of 1.

## Open Ports

---

The following are open ports in the framework:

- Catalog server port
- JMX port
- SSL port
- JDBC port
- HTTP port

**Note:** The SSL port is the secure port for HTTP.

The following table lists the default value for each port and how the default value can be changed:

Port	Default Value	How to Change the Default Value
<a href="#">Catalog server port</a>	If this port's value is not specified in the Configuration application, it is JMX+1.	The default value can be changed in the Configuration application.
JMX port	The default values for this port are as follows: <ul style="list-style-type: none"><li>• No default value for container deployment</li><li>• The default value for Velocity Studio is 10008</li><li>• The default value for runConfig is 10020</li><li>• The default value for runActivate is 10009</li></ul>	You can change the default value by changing the <a href="#">command lines</a> .
<a href="#">SSL port</a>	This port has a default value of 443.	The default value can be changed in the Configuration application.
<a href="#">JDBC port</a>	There is no default value. The port is from the Oracle server, which is normally 1521.	
<a href="#">HTTP port</a>	This port has a default value of 8080.	The default value can be changed in the Configuration application.

## System Limits

The following table lists the system limit descriptions, current and recommended limit values, and possible error or warning log messages:

Constructor	Description	Current limit value	Recommended limit value if increased	Error/Warning log message for exceeding the limit
CwpProcessManager, constant CwpMAX_PROC_LIMIT	Maximum number of processes.	5 000 000	No set limit (up to 2 G). Limited by the available memory. Large values may result in reduced speed.	DE0131 in CwpGlobalProcesses Constructor and initializePE in CwpProcessManager.
CwfMetadataElement. ORA_IDENTIFIER_MAX_LEN	Maximum string length for table names.	30	Limit value cannot be increased.	DE4200 in CwfLeaf validate() && DE4201 in CwfDBColumnBasedItem validate().
CwfConstants.CWF_CT_REF_LEVEL_LIMIT	Number of levels to which code tables can refer to other code tables.	3	Unknown.	DE6421 logged from CwfCTStringArray.readCT() method.
CwfEventHandler. MAX_EVENT_ID_LENGTH	Number of characters in the Event ID.	128	Limit value can be increased to accomodate operation for large clients or shared tenant/managed services.	No error or warning message. It only limits the results to 2500. CwfSQLFilter.runSelect() uses this limit to fetch 2500 records each time until it gets all records. Also, the CwfDataObjectList.isResultComplete method checks if the result is complete or partial.
CwpGlobalProcesses. GP_DEFINITIONS_LIMIT	Maximum number of Global Process definitions.	1000	Limit value can be increased but is not recommended.	DE0079 in CwpGlobalProcesses constructor.
CwfMetadataRTRule. executeRule() method	Maximum number of results returned by RTRules.	2500	Limit value can be increased but is not recommended.	No error or warning message. It merely limits the results to 2500. CwfSQLFilter.runSelect() uses this limit to fetch 2500 records each time until it gets all records. Also the CwfDataObjectList.isResultComplete method checks if the result is complete or partial.
CwcInfoTable.setCell() method	Maximum number of rows in InfoTable.	32 766	Limit value cannot be increased. Values above 10,000 are not recommended.	DE0906 in CwcInfoTable.setCell()
CwfMetadataFinder. MAX_ROWS_LIMIT	Maximum number of Finder rows.	9999	Limit value can be increased but is not recommended.	DE4176 in CwfFinder.validate()
CwfMetadataFinder. MIN_ROWS_LIMIT	Minimum number for maximum number of Finder rows.	100	Unknown.	DE4176 in CwfFinder.validate()
CwfMetadataFinder. CWF_VOLATILITY_MAX	Maximum volatility for Finders, in minutes.	9999	Limit value can be increased but is not recommended.	DE4177 in CwfFinder.validate()
CwfProcessBase.MAX_ACTIVITIES_IN_PROCESS	Maximum number of activities in a process definition.	4000	Limit value can be increased but is not recommended.	DE0096 in CwfProcessBase.validate()
ProviderFactory Constant MAX_CACHE_SIZE	Service provider cache limit.  Single cache holds client session related objects. For example, JMS session object or HTTP cookie.	10 000	Unknown.	No error or warning message. If full, objects are not placed in cache.  Existing objects are cleaned if unused for the cache expiry period.  ProviderFactory.getObject() is cleaning unused cache.
Defined in the configuration model (com.conceptwave.config.model.defaults) for each type of port but not exposed in the UI	Service provider cache expiry.	1 hour	Limit value can be decreased.	No error or warning message. Cache objects are removed.  (ProviderFactory.getObject() is cleaning unused cache.)
ListenerHandler.RECONNECT_ATTEMPT_PERIOD	Queue listener reconnect period.  Time after which a JMS/MQ listener thread attempts to reconnect to a queue after failing to connect.	10 seconds	Unknown	N/A
ListenerHandler.DISTRIBUTED_RECONNECT_PERIOD	Queue Listener Distributed Reconnect Period.  Time after which an existing listener queue connection is closed. Used in the distributed queue environment.	5 minutes	Unknown	N/A
ProviderFactory.invoke()	Service Provider Cache Based Retry Limit.  Number of times the interface invocation tries if an exception occurs and the provider used an object from the cache.	1	Limit can be exposed in the configuration.	Throws original exception from the invoke method.

# HTML and JavaScript Injection

HTML and JavaScript injection can occur when the resulting HTML page is modified without authorization due to insufficient validation of user-inputted data. The application should filter the user-inputted data and not blindly re-display data back to the browser. However, the user interface that is displayed by the framework is specifically defined by the application and the framework has no knowledge of what it should and should not filter.

When the application displays any kind of information such as labels or messages back to the user, the application should filter the data either during the time when the input data arrives, or when the data is output to the client. Data validation depends on the type of data that is involved.

# Metadata Development Coding Standards

This document provides a set of standards for metadata development . The following standards are covered:

- [Metadata Objects Naming](#)
- [Database \(DB\) Table Naming](#)
- [JavaScript Coding](#)
- [Other Standards](#)

**Note:** Some of the content is taken from the links provided in the references section.

## References

<http://java.sun.com/docs/codeconv/>  
<http://javascript.crockford.com/code.html>  
<http://code.google.com/p/jsdoc-toolkit/w/list>  
[http://www.dba-oracle.com/standards\\_schema\\_object\\_names.htm](http://www.dba-oracle.com/standards_schema_object_names.htm)

## Metadata Objects Naming Conventions

---

Naming conventions make application components more understandable, which reduces maintenance effort. The following table describes the naming suffix used for metadata objects:

Metadata Object		Suffix
Data Dictionary	Data/Element Type	None
	Document	Doc
	Data Structure	DS
	Conversion Map	CM
External Services	Service Provider	SP
	Interface	Int
	Binding	Bind
	Service	Serv
Business Processes	Exception	Proc
	Signal	Sig
	Participant	Part
	Alert	Alert
	Process	Proc
	Script	None (name should start with lowercase and match script function name)
	Rule	Rule
Orders and Finders	Life Cycle	LC
	Order	Order
	Finder	Fnd
User Interface	Menu	Menu
	Navigation Tree	NT
	Dashboard	DB
	Decision Tree	DT
Security	Permissions	Perm

## Namespaces

---

All application namespaces can have a prefix such as <http://www.company.com/app>. For example:

- **Common:** Common and reusable components.

<http://www.mycompany.com/myapp/common>

- **Order:** Order and related documents, mapping, finders, and UI elements.

<http://www.mycompany.com/myapp/order>

- **Processes:** Workflows and related elements including participant interfaces, participants, processes, and javascripts.

<http://www.mycompany.com/myapp/process>

## CW Artifact Names

---

When producing metadata deliverables, apply naming conventions as follows:

<b>Filename</b>	<b>Example</b>
Metadata filename	COMPANY_PROJECT.xml
Configuration metadata filename	COMPANY_PROJECT_config.xml
User profile administrator metadata filename	COMPANY_PROJECT_upadmin.xml
Upgrade SQL script file	COMPANY_PROJECT_UpgradeSQL.sql
Active metadata named	PROJECT M.m.b
Where	
M : Major release number	
m : Minor release number	
b : Build release number ( use SVN rn )	
Alternative : PROJECT M.m YYYY-MM-DD	

## Database (DB) Tables Naming Conventions

---

The following are description of standard naming conventions for Oracle schema objects including table, index and column naming standards.

### Note:

If a table name, column name or an index name exceeds 30 characters, reduce the size of the table name in this order:

1. From the left of the table name, remove vowels from each word in the table name except for the first vowel of each word.
2. If the table name is still greater than 30 characters, use standardized shorthand indicators. Record this standard for consistent use.

## Table Naming

---

- Use full table names whenever possible.
- If table name consists of two or more words; do not separate them with underscore (Oracle naming conventions point the opposite, but all existing tables used by Order Care product are without underscores).
- When document mapped to the DB is part of an order, use document name as prefix in DB table name.

For example:

**VTOOrder**

**Summary**

**Offer**

**TV Configuration**

    ...

DB tables should be named:

**VTO\_Summary**

**VTO\_Offer**

**VTO\_TV Configuration**

    ...

## Column Naming

---

- Spell out column names whenever possible.
- If column name consists of two or more words; do not separate them with underscore (Oracle naming conventions point the opposite, but all existing tables used by Order Care product are without underscores).

## Index Naming Convention

---

For index names follow this standard:

**IDX\_tttt\_nn**

Where

**IDX** Index

**tttt** Table name the index is built on.

**nn** Numeric value that makes each table index unique.

# Javascript Functions

---

This page provides the information about the Javascript Functions:

## Function Name

Use the exact name of the function itself for the JavaScript function name. The function label includes the function parameters as well. In addition, functions starts with comment block, describing the functionality, the input parameters, their type, short description, and the return value.

Example for function: `myFunction()`

Metadata Name: `myFunction`

Metadata Label : `myFunction(param1,param2,param3)`

## Parameters

All parameters correspond to **Parameters** section.

Define parameters type as specific as possible. If there is a data type specified for the parameter, then it is used rather than a type **string**.

Name	Type	Mandatory
ord	object	✓
accessId	Document ID (cwf system)	✓
scope	Offer Scope (VT PC Extension)	✓

## Function Body

Start each function beginning with comments, with information about the function. Organize the comments with following tags, so a complete set of documentation can be generated using a tool like JSDOC.(<http://code.google.com/p/jsdoc-toolkit/w/list>)

```
/** * @description * @param {paramType} paramName This is a string parameter * @throws ExceptionType This is the label text * @returns {returnType} Description * @author authorName * @since date of first release of MD version first release * @version release version and date for modifications */
```

## Line Length

Avoid lines longer than 80 characters. With current monitor resolutions, this requirement is not that strict.

## Wrapping Lines

If an expression dose not fit on a single line, break it according to these general principles:

- Break after a comma.
- Break before an operator.
- Prefer higher-level breaks to lower-level breaks.
- Align the new line with the beginning of the expression at the same level on the previous line.
- If the above rules lead to confusing code or to code that is squished up against the right margin, just indent 8 spaces instead.

Here are some examples of breaking method calls:

```
function(longExpression1, longExpression2,  
        longExpression4, longExpression5);  
var = function1(longExpression1,  
               function2(longExpression2,  
                         longExpression3));
```

The following are two examples of breaking an arithmetic expression. The first is preferred, since the break occurs outside the parenthesized expression, which is at a higher level.

```
longName1 = longName2 * (longName3 + longName4 - longName5)  
           + 4 * longname6; // PREFER  
  
longName1 = longName2 * (longName3 + longName4  
           - longName5) + 4 * longname6; // AVOID
```

The following are examples of line wrapping for **if** statements:

```
//DON'T USE THIS INDENTATION  
if ((condition1 && condition2)  
    || (condition3 && condition4)  
    || !(condition5 && condition6)) { //BAD WRAPS  
    doSomethingAboutIt(); //MAKE THIS LINE EASY TO MISS  
}  
  
//USE THIS INDENTATION INSTEAD  
if ((condition1 && condition2)  
    || (condition3 && condition4)  
    || !(condition5 && condition6)) {  
    doSomethingAboutIt();  
}  
  
//OR USE THIS  
if ((condition1 && condition2) || (condition3 && condition4)  
    || !(condition5 && condition6)) {  
    doSomethingAboutIt();  
}
```

Here are three acceptable ways to format ternary expressions:

```
alpha = (aLongBooleanExpression) ? beta : gamma;  
alpha = (aLongBooleanExpression) ? beta  
      : gamma;  
alpha = (aLongBooleanExpression)  
      ? beta  
      : gamma;
```

## Comments

Be generous with comments. It is useful to leave information that can be read at a later time by people who need to understand what you have done. The comments needs to be well-written and clear, just like the code they are annotating. It is important to keep the comments up-to-date. Erroneous comments can make programs even harder to read and understand. Make comments meaningful. Focus on what is not immediately visible. Do not waste the reader's time with a comment like:

```
i = 0; // Set i to zero.
```

Generally use line comments. Save block comments for formal documentation and for commenting out.

## Variable Declarations

Declare all variables before use, although JavaScript does not require the declaration. Preferably, give each variable its own line and

comment. List them in alphabetical order.

```
var currentEntry; // currently selected table entry
var level;        // indentation level
var size;         // size of table
```

Put declarations only at the beginning of blocks. A block is any code surrounded by curly braces { and }. Do not wait to declare variables until their first use. It can confuse the programmer and hamper code portability within the scope.

```
void MyMethod() {
    int int1; // beginning of method block
    if (condition) {
        int int2; // beginning of "if" block
        ...
    }
}
```

The one exception to the rule is indexes of **for** loops, which in Java can be declared in the **for** statement:

```
for (int i = 0; i < maxLoops; i++) { ... }
```

Avoid local declarations that hide declarations at higher levels. For example, do not declare the same variable name in an inner block:

```
int count;
...
func() {
    if (condition) {
        int count; // AVOID!
        ...
    }
}
```

Try to initialize local variables where they are declared. The only reason not to initialize a variable where it is declared is if the initial value depends on some computation occurring first.

## Simple Statements

Each line contains at most one statement. Put a ; (semicolon) at the end of every simple statement.

## Compound Statement

Compound statements are statements that contain lists of statements enclosed in {} (curly braces).

- The enclosed statements are indented four more spaces.
- The { (left curly brace) remains at the end of the line that begins the compound statement.
- The } (right curly brace) starts a line and is indented to align with the beginning of the line containing the matching { (left curly brace).
- Braces are used around all statements, even single statements, when they are part of a control structure, such as an **if** or **for** statement. This makes it easier to add statements without accidentally introducing bugs.

## Return Statement

Avoid use of parentheses in a **return** statement with a value unless they make the return value more obvious in some way.

```
return;
return myDisk.size();
return (size ? size : defaultSize);
```

## For Statement

A **for** class of statements can have the following form:

```
for (initialization; condition; update) {  
    statements  
}  
  
for (variable in object) {  
    if (filter) {  
        statements  
    }  
}
```

The first form is used with arrays and with loops of a pre-determinable number of iterations.

The second form is used with objects. Be aware that members that are added to the prototype of the *object* are included in the enumeration. It is wise to program defensively by using the *hasOwnProperty* method to distinguish the true members of the *object*.

```
for (variable in object) {  
    if (object.hasOwnProperty(variable)) {  
        statements  
    }  
}
```

## If, If-Else, If-Else If Statements

The **if** class of statements can have the following form:

```
if (condition) {  
    statements  
}  
  
if (condition) {  
    statements  
} else {  
    statements  
}  
  
if (condition) {  
    statements  
} else if (condition) {  
    statements  
} else {  
    statements  
}
```

## While Statement

A **while** statement can have the following form:

```
while (condition) {  
    statements  
}
```

## Do Statement

A **do** statement can have the following form:

```
do {
```

```
    statement  
} while (condition);
```

Unlike the other compound statements, the do statement always ends with a ; (semicolon).

## Switch Statement

A **switch** statement can have the following form:

```
switch (expression) {  
    case expression:  
        statements  
    default:  
        statements  
}
```

Each **case** is aligned with the **switch**. Switch avoids over-indentation.

Each group of statements, except the default, ends with **break**, **return**, or **throw**. Do not fall through.

## Try-Catch Statement

The **try** class of statements can have the following form:

```
try {  
    statements  
} catch (variable) {  
    statements  
}  
  
try {  
    statements  
} catch (variable) {  
    statements  
} finally {  
    statements  
}
```

## Whitespaces

Blank lines improve readability by setting off sections of code that are logically related.

Use blank spaces in the following circumstances:

- Separate the keyword followed by ( (left parenthesis) by a space.
- Use a blank space between a function value and its ( (left parenthesis). A blank space helps to distinguish between keywords and function invocations.
- Separate all binary operators except . (period) and ( (left parenthesis) and [ (left bracket) from their operands by a space.
- Do not use a space to separate a unary operator and its operand except when the operator is a word such as **typeof**.
- Use a space after each ; (semicolon) in the control part of a for statement.
- Ensure that every , (comma) is followed by a whitespace.

## Metadata Development Coding Other Standards

---

The following describes other standards related to metadata development coding.

### Order Status Codes (in OrderInstance Document)

Order Status field represents a hierarchical structure. The structure makes the status flow clear and searching for orders faster. It is much more convenient and easy to implement. Each project has its own requirements for order status. Follow the example when creating the order status codes:

Code	Hierarchy	UI Description
O_NR_NEW	open.notrunning.new	New
O_R_CON	open.running.configuration	Configuration
O_R_PRO	open.running.provisioning	Provisioning
O_NR_ERR	open.notrunning.error	Error
C_C	closed.completed	Completed
C_X	closed.cancelled	Cancelled
...	...	...

### Global Configuration Variables (in Configuration Tool)

Configuration variables are grouped by their type, subtype, etc. Configuration names are in mixed case, with first letter lowercase. Use a period (.) to separate types and subtypes. For example:

Key	Value
host.name.smtpServer	smtp.conceptwave.com
host.name.pop3Server	pop3.conceptwave.com
host.port.smtpServer	25
host.port.pop3Server	110
url.conceptwave	<a href="http://www.conceptwave.com">http://www.conceptwave.com</a>
time.archiveOrders	01:00
time.reloadCache	03:30
path.interfacesRoot	D:\interfaces
num.maxResults	1000
....	....

### Password Configuration (in Configuration Tool)

Define passwords using the Passwords tab. Instead of defining configuration variables with clear text, follow metadata naming conventions. For example:

Password parameter name	Password value
smtp.provisioning.password	*****

### Privileges for Manual Participants (in User Profile Administrator Tool)

Privileges needed to define manual participants can have a suffix such as **PartPriv** to separate them from the rest of the privileges. For example:

Participant	Participant Privilege
-------------	-----------------------

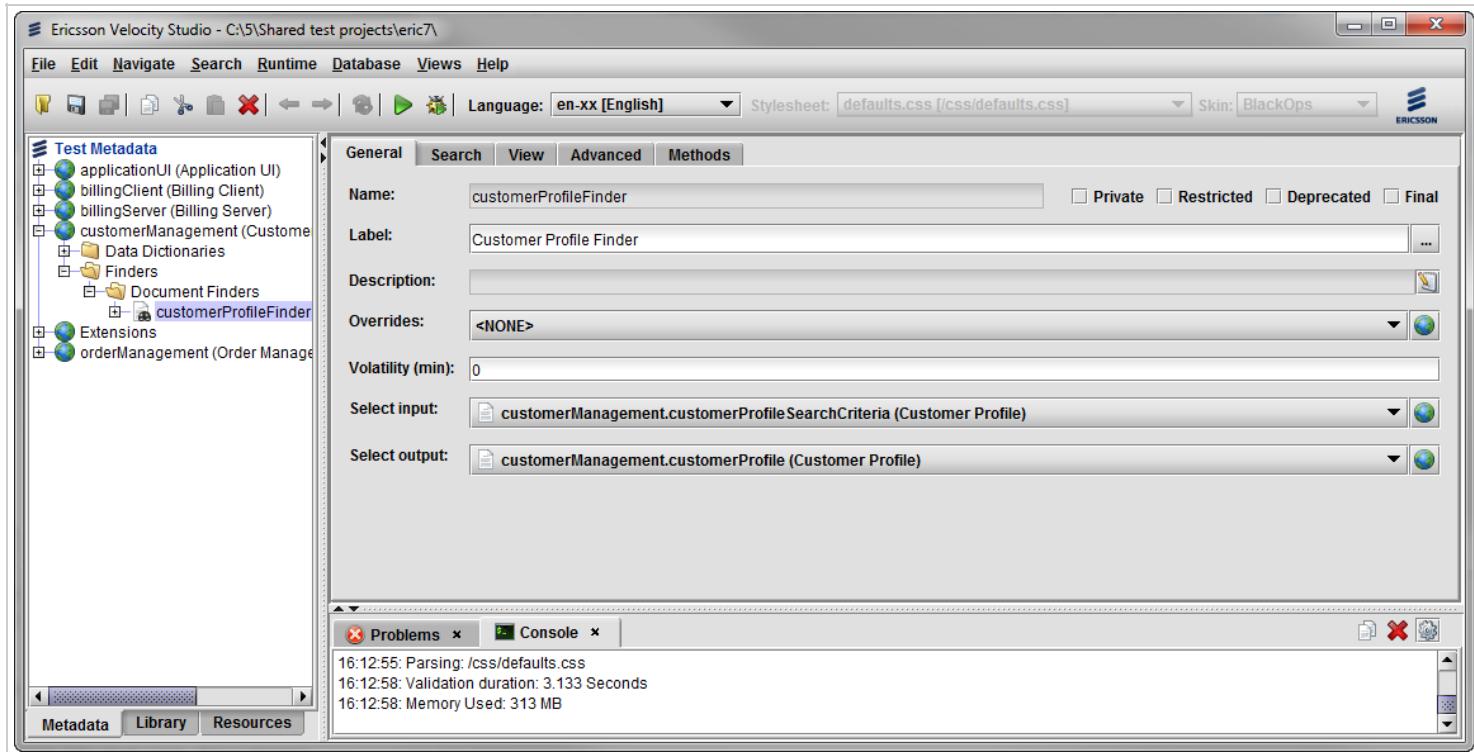
serviceActivationPart	serviceActivationPartPriv
translationsPart	translationsPartPriv

# Velocity Studio User Interface

To launch the Velocity Studio user interface, complete one of the following:

- In **Windows**, open the Start Menu, and select **CWOrderCare > Designer**
- Double-click `<installation_folder>\designer\env\startDesigner.cmd`

You may also launch Velocity Studio with the [command line](#), where you may also change optional parameters as necessary.



The Velocity Studio main window consists of these components:

## Window title bar

If a project is loaded, the project location appears in the title bar.

## Menu bar

The menu bar appears at the top of the window.

## Toolbar

The toolbar appears under the menu bar.

## Navigation pane

This pane appears under the toolbar on the left and serves to control navigation through:

- [Metadata](#) tab - shows the namespaces from the application metadata.
- [Library](#) tab - shows the metadata namespaces that come from external includes and from the product (refer to section on Namespace Types). Metadata in the **Library** tab is read-only but can be used by application metadata. See [Product Metadata](#) on how to leverage libraries to accelerate your application development.
- [Resources](#) tab - shows the resources of the project, such as images.

## Detail pane

The Detail pane appears under the toolbar on the right and displays the details of the currently selected metadata object in the **Navigation** pane.

## Notifications pane

This pane is located under the Detail pane, which displays:

- [Problems](#) tab - Shows all errors of the project.
- [Search](#) tab - Displays search results.
- [Console](#) tab - Shows console output of the project.
- [DB Mapping](#) tab - Displays results of `check DB mapping` command.
- [Breakpoints](#) tab - Provides a list of all scripts in Velocity Studio that have breakpoints set.

You can also perform a number of [common actions outside Velocity Studio](#) and [minimize the preview area](#).

## Toolbar

The toolbar appears below the menu bar and contains icons for frequently used menu items. The Refresh button, Stylesheet and Skin are only applicable (and enabled) when you are managing a [Form](#) in Velocity Studio.

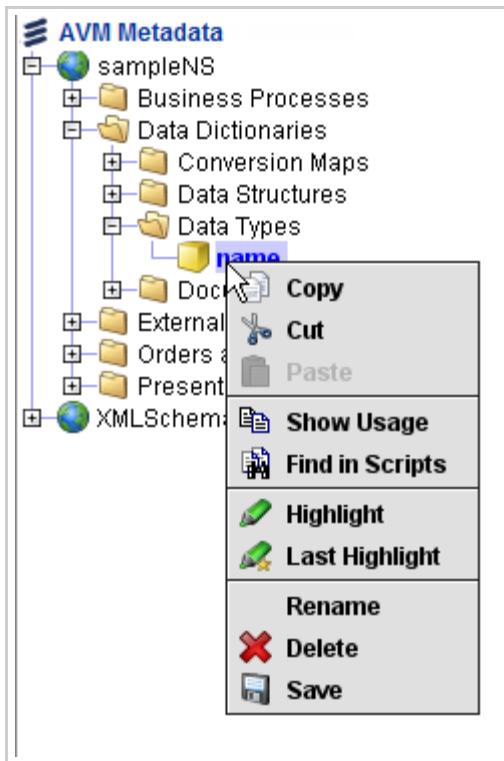


The table below describes the functions of each button in the toolbar.

Button	Function
	Open an existing metadata version.
	Save the selected metadata object.
	Save all changed metadata objects.
	Copy the selected metadata object.
	Cut the selected metadata object.
	Paste the cut or copied metadata object.
	Delete the selected metadata object.
	Navigate to the previous object in metadata object selection history.  <b>Note:</b> If you are editing a script directly from the <b>Script</b> field and use this button to go back to your previous metadata object, Velocity Studio automatically saves any changes you have made to your script. You can also press the <b>F3</b> key to save your changes in your script while you edit it.
	Navigate to the next object in metadata object selection history.
	When managing a Form, it can refresh the Preview Pane.
	Click this icon to start runtime. It is a shortcut to clicking <a href="#">Runtime &gt; Run</a> from the menu bar.
	Click this icon to stop runtime. It is a shortcut to clicking <a href="#">Runtime &gt; Stop</a> from the menu bar.
	Click this icon to start runtime in debug mode. It is a shortcut to clicking <a href="#">Runtime &gt; Debug</a> from the menu bar.
	Click this icon to stop runtime in debug mode. It is a shortcut to clicking <a href="#">Runtime &gt; Stop</a> from the menu bar.
<b>Language:</b> <input type="button" value="en-xx [English] ▾"/>	When managing a Form, selects the language to browse by in Preview Pane. See <a href="#">Form</a> for more information about configuring languages inside of form elements.
<b>Stylesheet:</b> <input type="button" value="cwf.css [/cwf/css/cwf.css] ▾"/>	When managing a Form, selects the stylesheet to render by in Preview Pane.
<b>Skin:</b> <input type="button" value="Enterprise ▾"/>	When managing a Form, selects the skin to render by in Preview Pane.

## Navigation Pane - Metadata

The **Metadata** tab of **Navigation** pane contains a hierarchy of the metadata objects. These objects are of application metadata. (Product metadata, also known as libraries, are in **Library** tab.) The metadata root object is at the top of the structure. Multiple [namespaces](#) can be created under this root object. Each namespace has a predefined structure of type folders (for example, **Data Types**) in which new metadata objects can be created (for example, **Postal Code - USA**). The figure below shows the Metadata tab of Navigation pane with a right-click pop-up menu.



When editing an existing metadata object in the Details Pane and then click away to another node in the Navigation Pane, a pencil appears at the top-left of the icon of the metadata object's node in the Navigation Pane, to signify it is now "dirty". For example, a dirty Document has this icon: (compared to ). This signifies that the metedata must be saved or the changes will be lost. Use the **Save** or **Save All** command to save the changes to the metadata object's XML file.

If the **Label** of a metadata object is different from the **Name** of the metadata object, by default, the corresponding node in the pane shows the name, and then appended by its label in brackets. The nodes' display format in the pane can be configured in [preferences](#).



The Metadata objects that you manage here impact the **Metadata** folder of your Project (source) directory in file system. Overall, metadata creation and deletion impacts the source directory immediately, whereas editing metadata do not impact the source directory until they are saved. The following describes the details of each action:

Action	Impact on Source Directory
Create Namespace	A folder by the Namespace's name is created, and an XML file with filename <b>nspage_</b> + Namespace's name is created in the folder to store the Namespace's properties. These are created immediately upon Namespace creation; no saving is necessary.
Create All Other	An XML file is created in its Namespace folder, with filename by the Metadata object's name prefixed with its object type, as tabularized in <a href="#">Metadata Objects Overview</a> . For example, if a Data Type object of name "postalCode" is

<b>Metadata Objects</b>	created, the XML file would be "dtype_postalCode.xml". The XML file is created immediately upon metadata object creation; no saving is necessary.
<b>Edit Metadata Object</b>	An existing metadata object can be edited in the Details Pane, which would be shown dirty by the mark of a pencil at the top-left of the icon . No source is changed without saving. When save command is issued, Velocity Studio writes the update to the metadata object's XML file. <b>Note:</b> The name of the metadata object is not editable in the Details Pane; use the Rename command in right-click pop-up menu instead.
<b>Rename Namespace</b>	A folder by the Namespace's new name is created, and all metadata object files in the original Namespace folder is moved to this folder. The original Namespace folder is then removed if it is empty. The Namespace XML file ( <b>nspace_</b> + Namespace's name) is updated and renamed to the new name. These are effective immediately upon rename at Velocity Studio; no saving is necessary.
<b>Rename All Other Metadata Objects</b>	The metadata object's XML file is updated with the new name, and the filename of the XML is also renamed according to the new name. This is effective immediately upon rename at Velocity Studio; no saving is necessary. <b>Note:</b> When renaming a metadata object, and the label and its name are identical, the object's label is automatically renamed.
<b>Delete Namespace</b>	Removes the Namespace folder in source, and all metadata objects' XML files within the Namespace folder. This is effective immediately upon deletion; no saving is necessary.
<b>Delete All Other Metadata Objects</b>	Deletes the metadata object's XML file. This is effective immediately upon deletion; no saving is necessary.
<b>Cut Metadata Object, and Paste</b>	The behaviour is as if the originating Metadata Object is deleted and the pasted Metadata Object is created.
<b>Copy Metadata Object, and Paste</b>	The behaviour is as if the pasted Metadata Object is created.
<b>Save</b>	Writes the metadata object into its XML file. If saving at Metadata Root (that is, top-level), it only saves its own metadata object, but not any metadata objects contained beneath it. If saving at Namespace level, all metadata objects underneath the Namespace are saved.
<b>Highlight (or Last Highlight)</b>	Highlight an existing metadata object is similar to editing the metadata object -- that is, the pencil would appear on top of the icon, marking it dirty. No source is changed without saving. When save command is issued, Velocity Studio writes the update to the metadata object's XML file.
<b>Refresh</b>	Reloads the metadata object from source directory. All unsaved changes of the metadata object that are in the Velocity Studio are lost. No impact to source directory. If refreshing at Metadata Root (that is, top-level) or Namespace level, all metadata objects underneath are refreshed as well.

**Note: You cannot move, copy or rename metadata files within project directory (for example, want a Document "B" from Document "A") directly in file system.** In addition to a GUID, there exists a hashvalue that hashes, and thus protects, the relative filepath (to the project directory), including the filename. Use the rename, cut/copy and paste function instead in Velocity Studio.

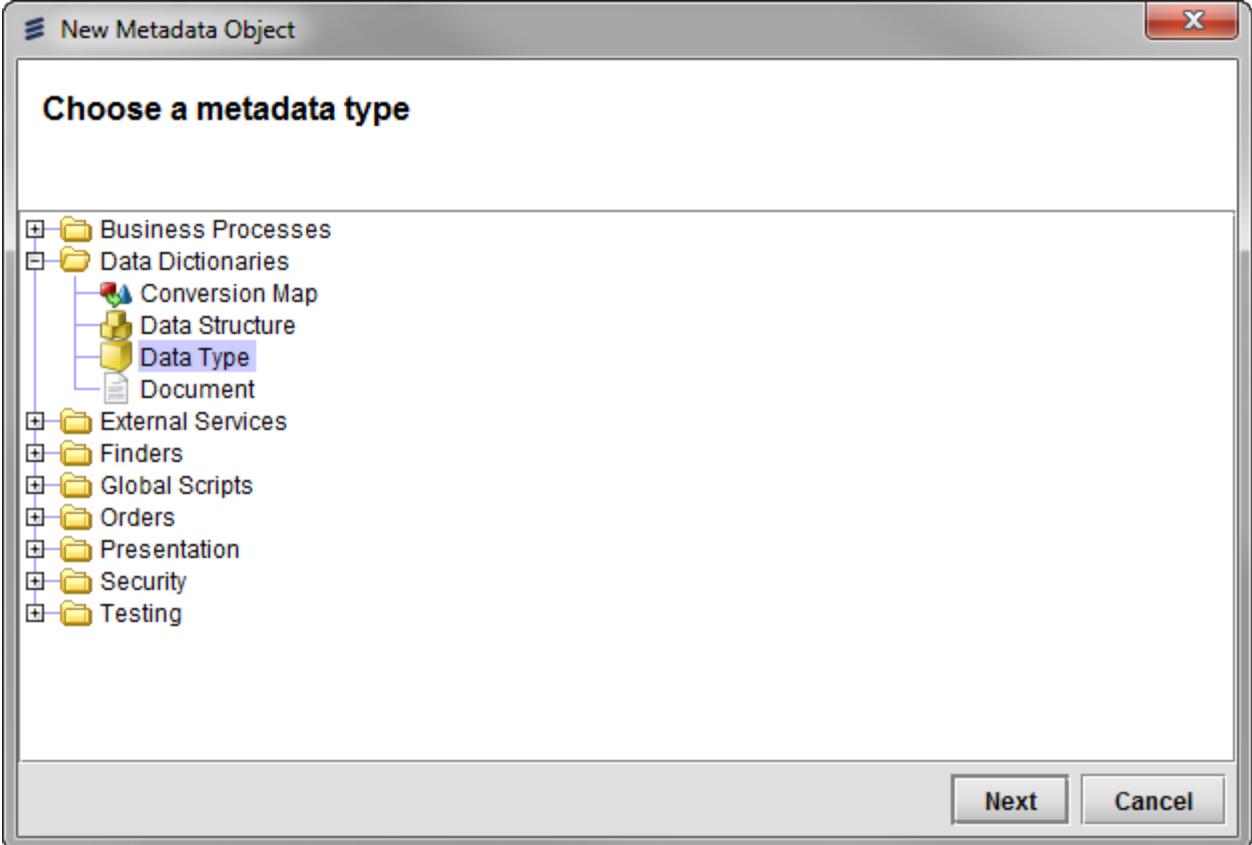
## Right-click Popup Menu

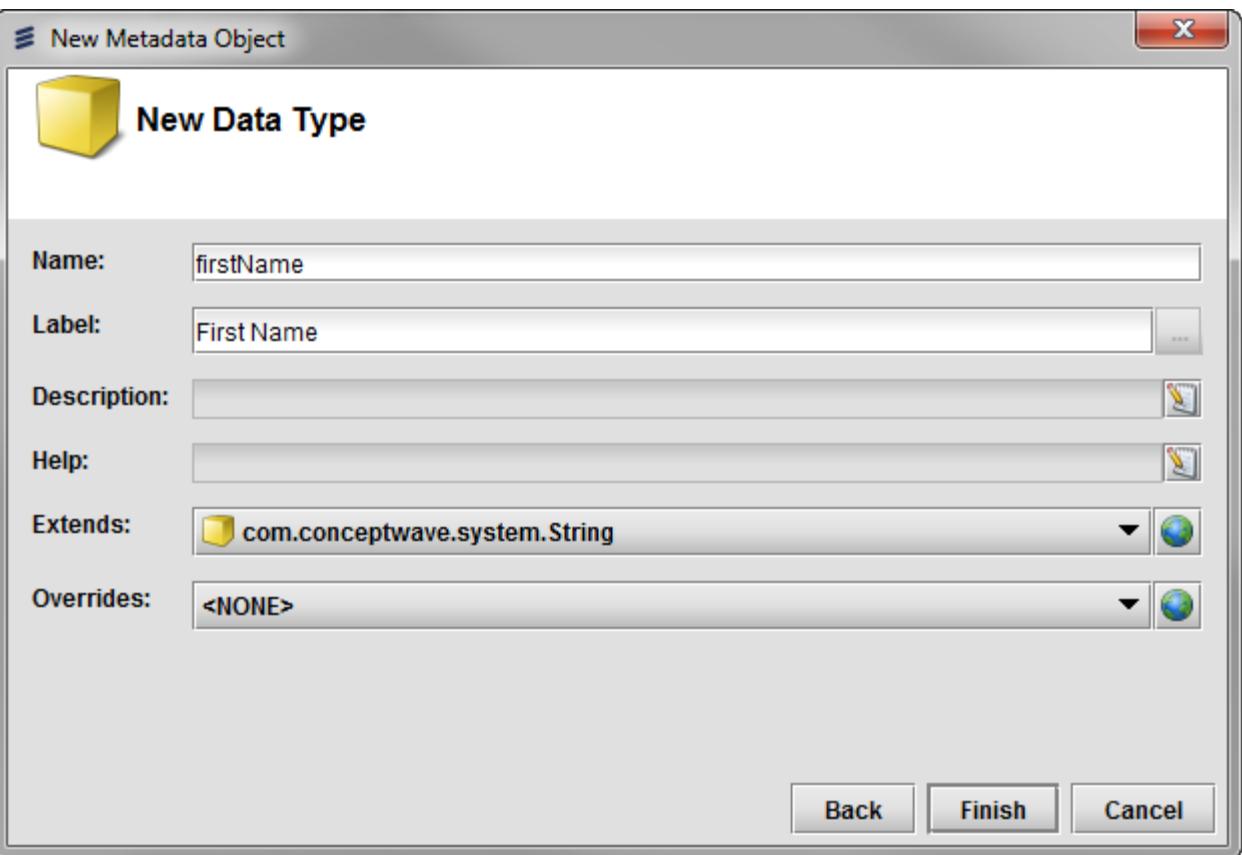
The popup menu is used to manipulate the metadata objects by right-clicking an object in the **Navigation** pane. The menu items available through the right-click popup menu allow a metadata object to be created, deleted, copied, or pasted or other metadata-specific operations to be performed.

The figure above shows the common right-click popup menu items available for all metadata objects in a namespace. These menu items are defined in the table below.

Multiple selections are allowed in the object tree (by holding **Ctrl**-key while clicking) for the **Copy** menu item. After the copy action, the **Paste** menu item will be available in the namespace right-click popup menu, as well as on the folder level if all selected objects have the same metadata type.

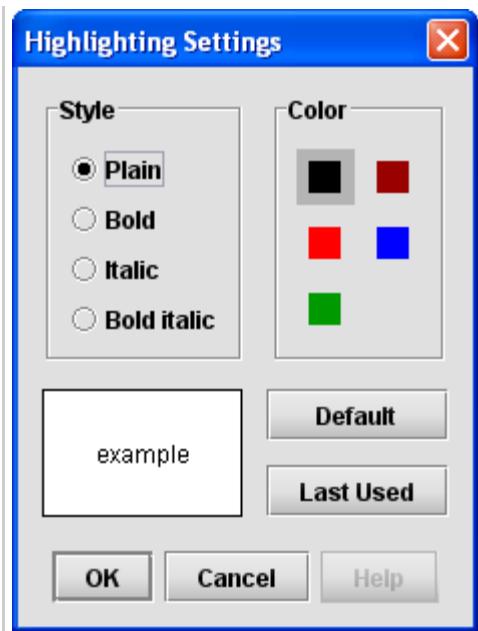
## Description of common popup menu items

Menu Item	Description
New ...	Appears for Namespace level only. Creates a metadata object of a type that can be chosen in the opened <b>New Metadata Object</b> wizard:   <p>Select the desired metadata type and click the <b>Next</b> button. All fields of that metadata type that must be initialized before object creation, such as <b>Name</b>, are prompted:</p>



Fill in the fields, then click the **Finish** button to create the metadata object. See references of each metadata object type in this Velocity Studio User Guide for instructions on each specific object type.

<b>New &lt;Metadata Object Type&gt;</b>	Appears for Namespace or type folders only. Creates a metadata object of this type. The second step of the same wizard in the above <b>New ...</b> command is prompted to fill in mandatory initialization fields before object creation.
<b>Copy</b>	Copies the selected metadata object to the clipboard.
<b>Cut</b>	Same as copy but, in addition, the original object will be removed after pasting.
<b>Paste</b>	Pastes a copy of the metadata object currently in the clipboard.
<b>Show Usage</b>	<p>Opens the <b>Dependent Objects</b> dialog that shows all metadata objects which have a reference to the selected object. You can also find variable instances that point to an object's user interface and user interface form. This function finds references in the following:</p> <ul style="list-style-type: none"> <li>• Form elements that reference the form</li> <li>• Pop-up actions or methods that reference the form</li> <li>• Finders that reference the form</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• The <b>Show Usage</b> function cannot find a method call in a script. This function is used for dependent metadata only.</li> <li>• The list of elements returned depends on the given object's usage. The elements in the list are not necessarily directly dependent on the given object, but can be indirectly dependent on it.</li> </ul>
<b>Find in Scripts</b>	Searches for metadata objects that use the selected object's name in their scripts (for example, a method call in a script).
<b>Highlight</b>	Highlights the selected object with a specified color and font style chosen from the <b>Highlighting Settings</b> dialog:



<b>Last Highlight</b>	Highlights the selected object using the last specified highlight color and font style.
<b>Rename</b>	Rename the selected metadata object to a different name.
<b>Delete</b>	Deletes the selected metadata object. If deleting at Namespace level, it also deletes all metadata objects underneath it.
<b>Save</b>	Saves the selected metadata object. If saving at Metadata Root, it only saves its own metadata object, but not any metadata objects contained beneath it. But if saving at Namespace level, it also saves all metadata objects underneath it.
<b>Refresh</b>	Reloads the selected metadata object from source directory. All unsaved changes of the metadata object that are in the Velocity Studio are lost. Refreshing at the Namespace or Metadata Root level will refresh all metadata objects underneath it.

## View Inherited Leaves from Base Documents

You can view inherited leaves ([variables](#)) from base documents through the Metadata tab's navigation tree. In the JavaScript editor, you can either click the **Select tree** button or use the **Ctrl + T** key combination to select your top-level metadata and their variable scripts. However, the JavaScript editor only shows leaves defined in the document, but not those inherited.

## Navigation Pane - Library

---

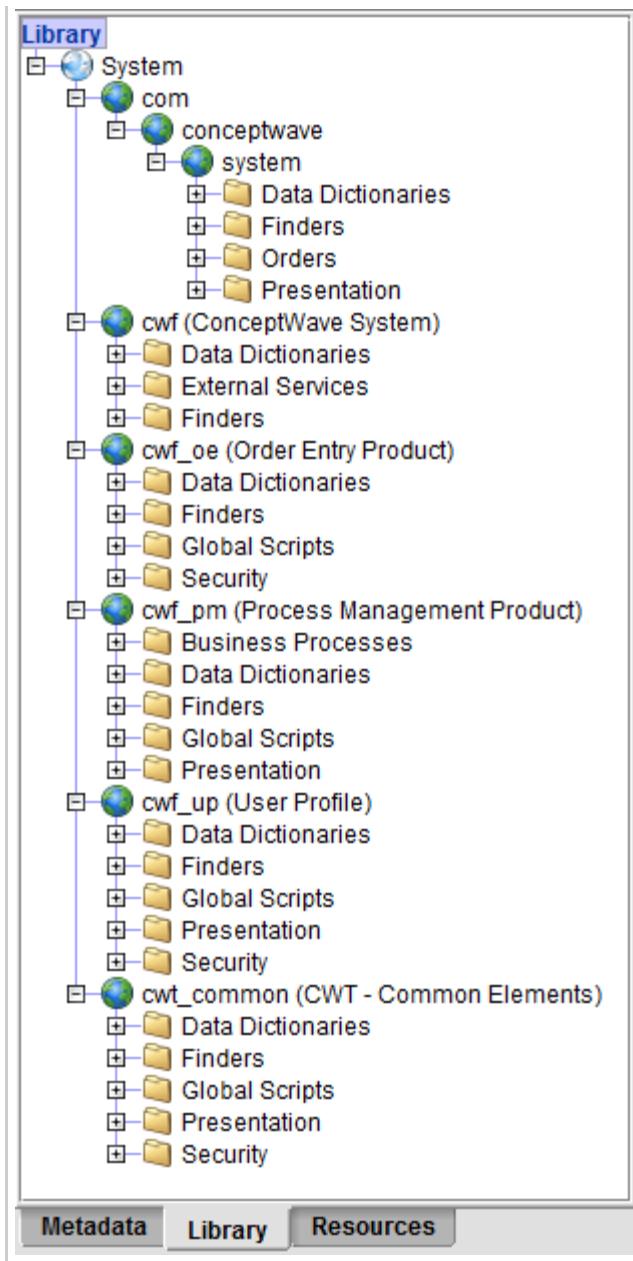
The **Library** tab of **Navigation** pane contains all libraries available for your project. It consists of the following:

- All core product metadata that are exposed to all projects in Velocity Studio
- All project-specific libraries that are loaded from **Templates** folder of your project root directory. These may be accelerators, such as *Product Catalog* and *Customer library*, or it may be other libraries provided by partners or your project team.

**Note:** To add libraries to your project, go to **Metadata** tab of **Navigation pane**, and use [Library tab](#) of top-level metadata object to add JAR files into your project root directory's Template folder. A project reload is necessary to have the added libraries shown in the **Library** tab of the **Navigation Pane**.

The following is a snapshot of the core product metadata available for your application development:

- **com.conceptwave.system** - base metadata objects of various types exposed by core. Most application metadata objects are derived from these base objects.
- **cwf** - Framework objects, which are core system metadata that implements many of the product's "off-the-shelf" functionality. Only relevant metadata of the Framework is exposed here, others remain hidden.
- **cwf\_oe** - Framework objects for Order Entry component. Notably, the following Documents may be overridden to provide customization to your application:
  - *cwf\_oe:login* - User login
  - *cwf\_oe:SelectApp* - Application selection
  - *cwf\_oe:OptionalItemSelect* - Add optional Order Item selection
- **cwf\_pm** - Framework objects for Process Management
- **cwf\_up** - Framework objects for User Profile management
- **cwt\_common** - Framework objects for Common Elements



This **Library** tab is a means to browse and access the library metadata -- to reference, copy, and override them, as explained in section [Managing Metadata](#). Library metadata cannot be changed. The right-click popup menus for library metadata objects do not contain actions that incur changes to metadata. The common menu items are as follows:

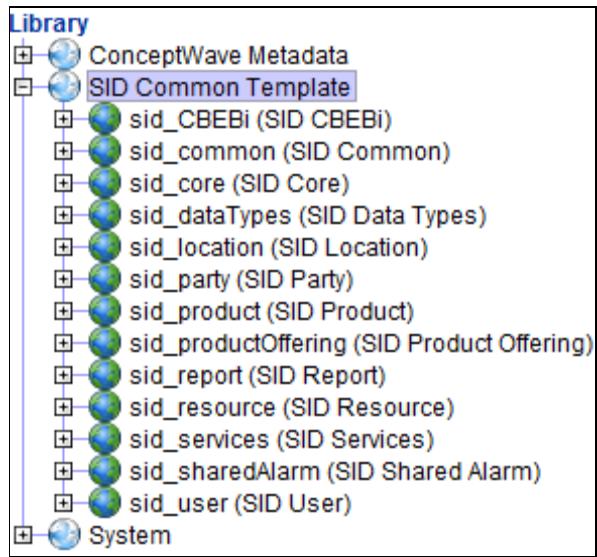
- **Copy** - Copies the selected metadata object to the clipboard.
- **Show Usage** - Opens the **Dependent Objects** dialog that shows all metadata objects which have a reference to the selected object. You can also find variable instances that point to an object's user interface and user interface form. This function finds references in the following:
  - Form elements that reference the form
  - Pop-up actions or methods that reference the form
  - Finders that reference the form

#### Notes:

- The **Show Usage** function cannot find a method call in a script. This function is used for dependent metadata only.
- The list of elements returned depends on the given object's usage. The elements in the list are not necessarily directly dependent on the given object, but can be indirectly dependent on it.
- **Find in Scripts** - Searches for metadata objects that use the selected object's name in their scripts.

When you add a library (that is, a .jar file) to your project, Velocity Studio allows you to display the header file label in the JAR file as

the library's root node, instead of the JAR file.



If the header file's label is either empty or null, the name appears instead.

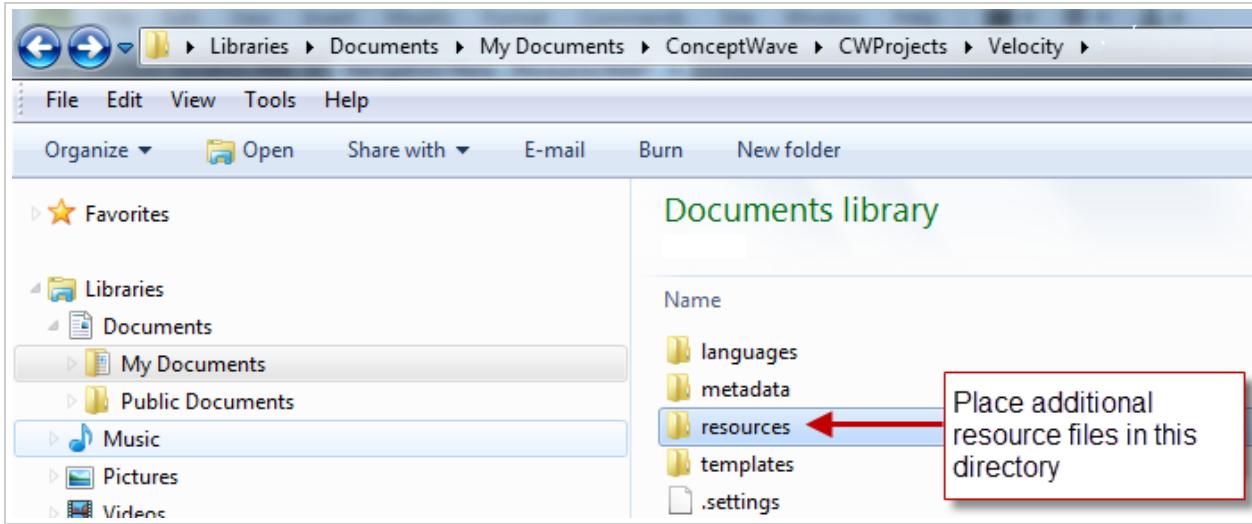
## Navigation Pane - Resource

The **Resources** tab of **Navigation** pane contains all resources available for your project, which consists of:

- all core product resources that are exposed to all projects in Velocity Studio
- all project-specific resources loaded from **resources** folder of your project directory.

### Adding Resources

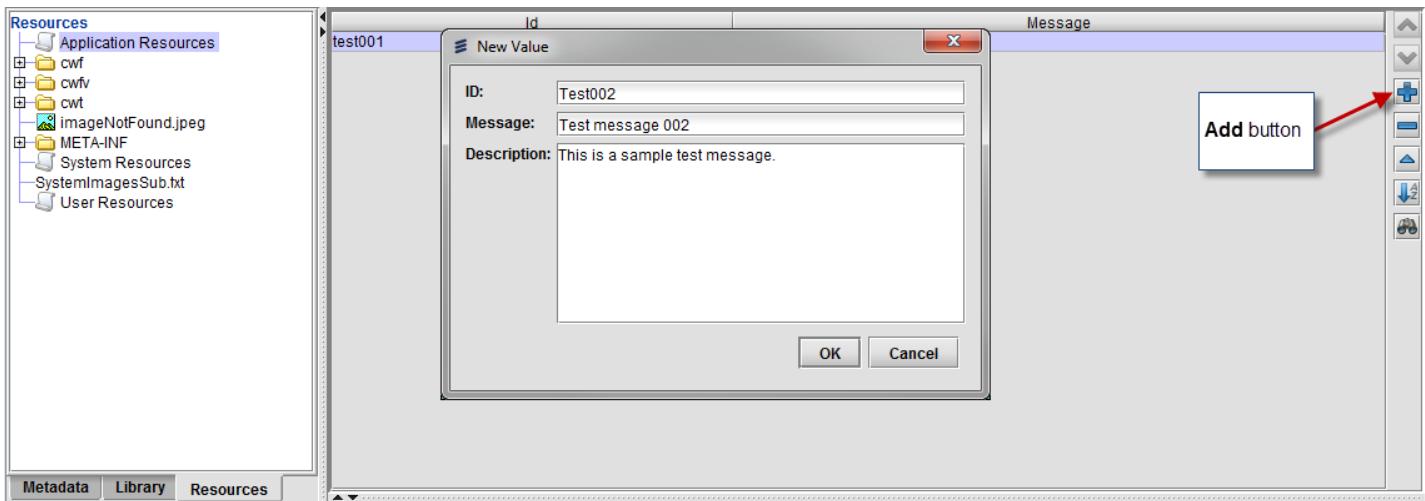
Additional Resource files can be added to the metadata by navigating to the metadata Resources folder and simply adding the file to the folder. Once the resource file is added to the Resource folder of the metadata, you will need to re-open the project to see the changes.



You can also add a resource by doing the following:

1. Click the **Add** button to launch the New Value dialog and fill in the following fields:

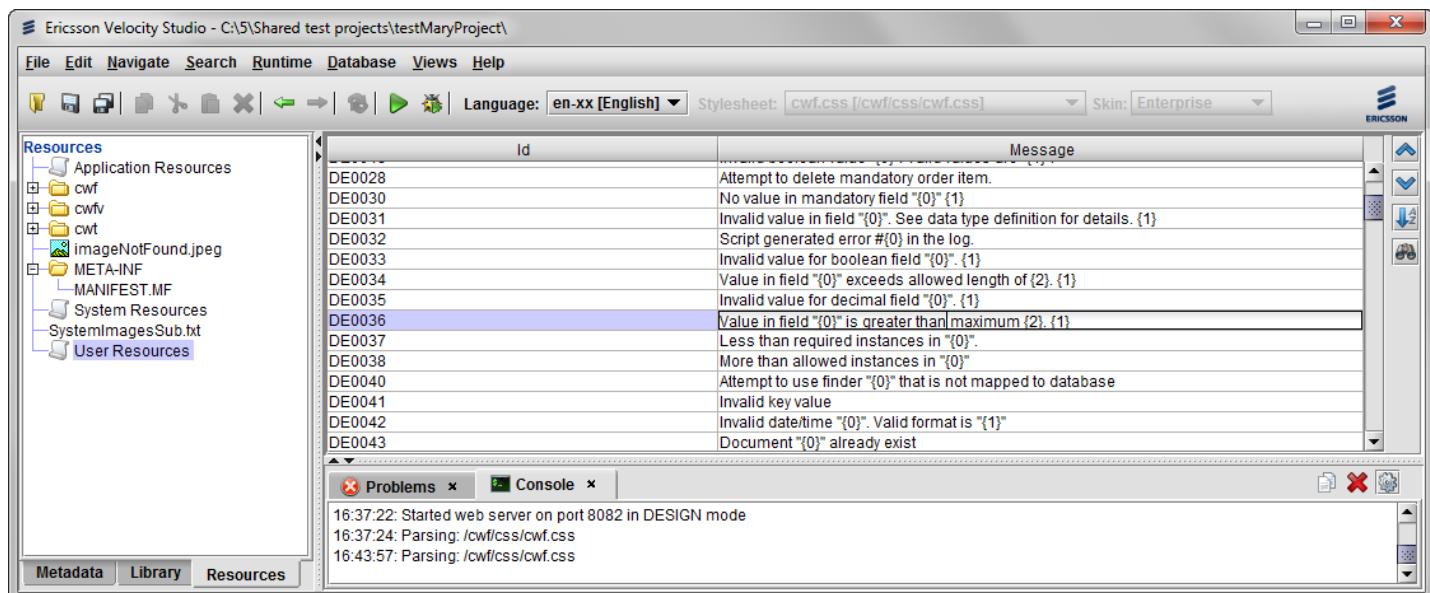
- A message **ID**
- A **Message**
- A **Description** about the message



2. Click the **OK** button to save your information.

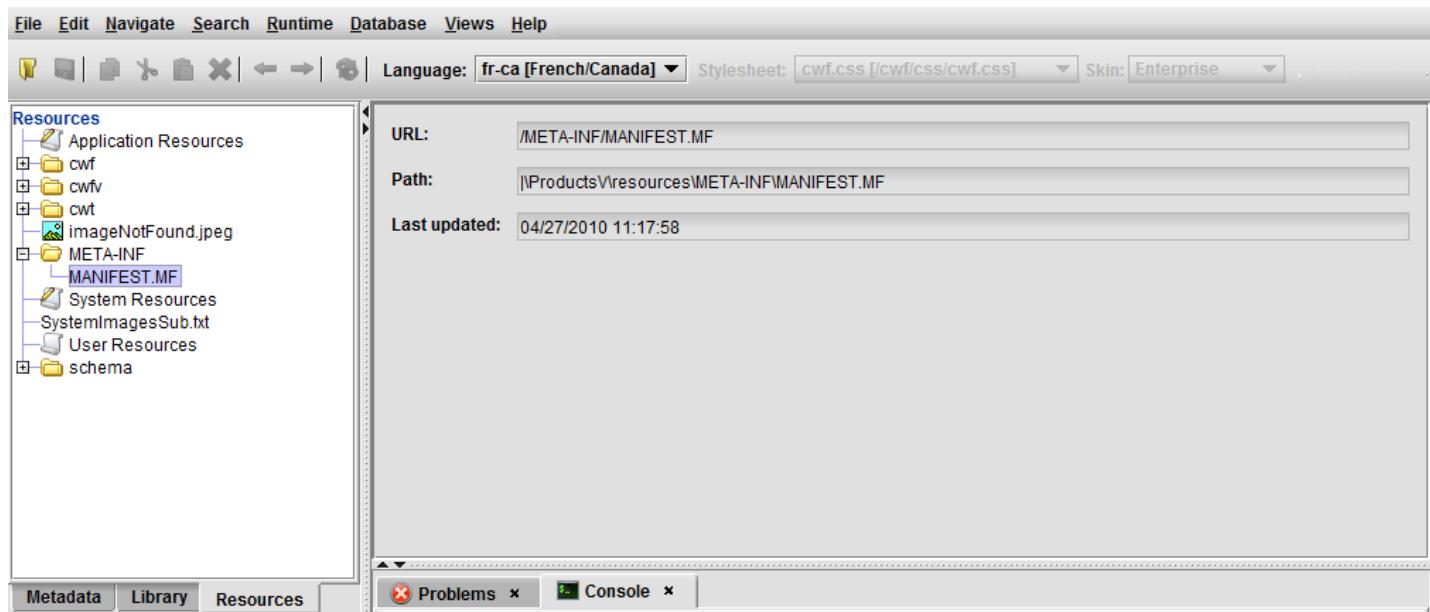
### Edit Resources

You can edit existing resources by double-clicking either an ID's **Message** or Description field an **ID**, and then editing the information. The ID field cannot be changed. Proceed to save your metadata project.



## Resources Property

When a resource is selected in the Resources tab of Velocity Studio, some properties of the resource are displayed:



Field	Description
URL	Use for metadata referencing.
Path	The file path of the resource.
Last updated	When the file is last updated.

Table below summarizes the resource types that are recognized by Velocity Studio. More description of individual resource types can be found in the sections that follow.

Resource Type	File Extensions	Import Technique
<a href="#">Image files</a>	.gif, .jpeg, .jpg, .png.	Place image files directly into project directory's <b>resource</b> folder and refresh resources tab.  The icons in the <i>cwf\resources\cwtl\images</i> folder is replaced with a newer version of icon images. If you want to continue using the former set of icons, you need to extract these images from the build (under <i>old_resources, cwt.jar</i> ) into your resources folder under <i>resources\cwtl\images</i> .
<a href="#">XSD files</a>	.xsd	Import XML Schema Definition (XSD) files with <a href="#">File &gt; Import &gt; XSD</a> menu command. The files are then stored in <b>schema</b> subfolder of <b>resources</b> folder in project directory.

<b>WSDL files</b>	.wsdl	Import Web Service Description Language (WSDL) files with <a href="#">File &gt; Import &gt; WSDL</a> menu command. The files are then stored in <b>schema</b> subfolder of <b>resources</b> folder in project directory.
<b>Stylesheets</b>	.css	Place .css files directly into project directory's <b>resource</b> folder and refresh resources tab.
<b>Archive files</b>	.jar	Place JAR files directly into project directory's <b>resource</b> folder and refresh resources tab.
<b>System Image Substitution List file</b>	SystemImagesSub.txt	Always at the root <b>resource</b> folder of project's directory.
<b>Message Resources</b>	xml	These are system generated notes that contain the message resource files for User, Application and System.
<b>All Other File Types</b>	<Any>	Simply create subfolders and place files in project directory's <b>resource</b> folder and refresh resources tab.

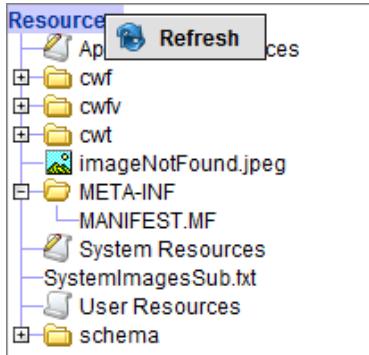
Different resources may collide on the same URL, because:

- project resources may be located at the same relative path as product resources
- project resources may be archive files, where these files may contain arbitrary folders and files in them.

The following is the order of precedence in resource resolution of the same URL:

1. project resource
2. project resource in archive files; if resources of the same URL among multiple archives files exist, alphabetical order of the archive filename is used to resolve order (for example, *a.jar* and *b.jar* both has *logo.gif*, then *a.jar*'s image file is used)
3. product resource

Changes are effective immediately when managing resources in the **Resources** tab. The Save command in Velocity Studio does not affect resources. If you have modified/added/deleted files in the project directory's **resource** folder, you can right-click the top-level **Resource** node in the tab and click **Refresh** to reload the updated resources.



**Note:** Unlike 4.x and previous predecessors, "Template Resources" no longer exist. Libraries are to be included per project, in **Navigation Pane's Metadata** tab, top-level metadata's **Library** tab.

### Image Files

When an image file is selected, the image is shown in image preview.

<b>URL:</b>	/nR/HDW19592001-83-87.gif
<b>Path:</b>	C:\Documents and Settings\danielho\My Documents\metadata6\resources\nR\HDW19592001-83-87.gif
<b>Last updated:</b>	02/12/2009 12:04:27
<b>Substitution Path:</b>	C:\Documents and Settings\danielho\My Documents\metadata6\resources\nR\logo.png
<b>Substitution URL:</b>	

**Select Substitution** **Remove Substitution**

**Image preview:** original      substitute



83x87

In addition, you can substitute the original image with another resource image (it is typically done for product images to alter images rendered in framework UI) by selecting the **Select Substitution** button, and select the replacement image in the **Image Resource** dialog. The original image is thus substituted as specified, and the image node in the resources tab becomes blue to indicate substituted.

<b>URL:</b>	/nR/HDW19592001-83-87.gif
<b>Path:</b>	C:\Documents and Settings\danielho\My Documents\metadata6\resources\nR\HDW19592001-83-87.gif
<b>Last updated:</b>	02/12/2009 12:04:27
<b>Substitution Path:</b>	C:\Documents and Settings\danielho\My Documents\metadata6\resources\nR\logo.png
<b>Substitution URL:</b>	/nR/logo.png

**Select Substitution** **Remove Substitution**

**Image preview:** original      substitute



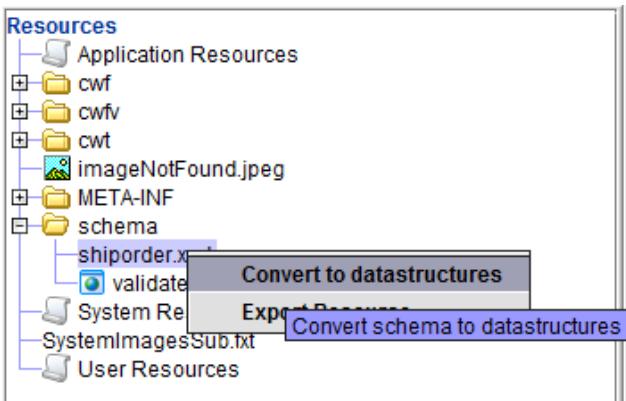

83x87      59x59

You can clear the substitution by clicking the **Remove Substitution** button.

**Note:** Be aware that all images are cached in client Web browsers. Changes to images made here will only be reflected after subsequent metadata activation OR if the client browser cache is explicitly cleared. For example, in Internet Explorer 8, click **Tools > Internet Options** from the menu bar, click **Delete...** in Browsing history, and then select the **Temporary Internet files** to delete.

## XSD Files

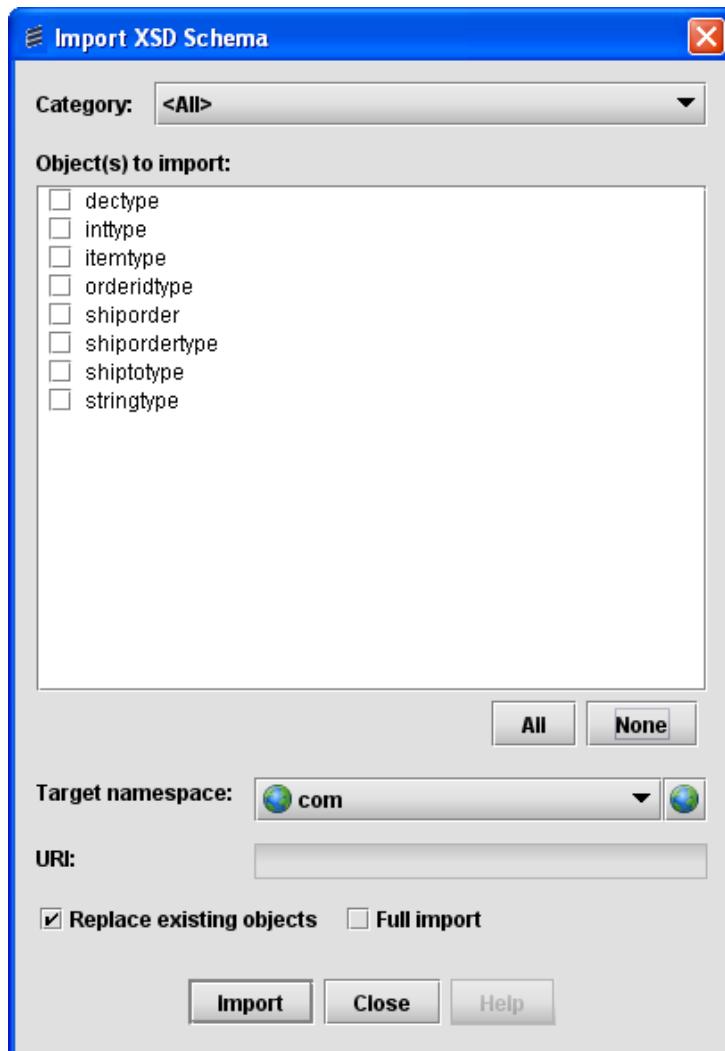
You can create Data Types and Data Structures based on an XSD resource file into application metadata, by right-clicking the resource node and select **Convert to datastructures** to the menu pop-up.



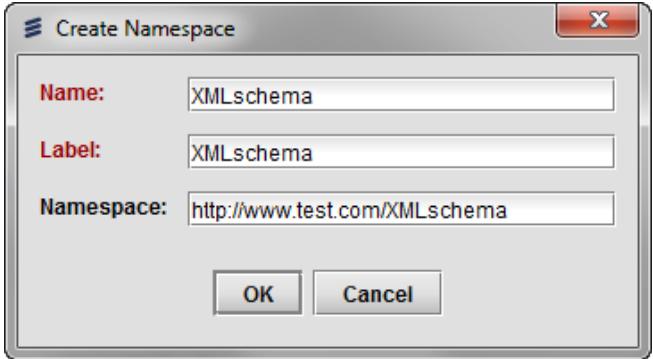
**Note:** This command is a helper function to accelerate development. Instead of manually creating the corresponding objects, the metadata developer can automatically generate them. When this command is used, certain assumptions are made that may not be valid in all cases. It is recommended that the developer carefully examines the generated objects and does appropriate changes where needed.

Velocity Studio always attempts to use the existing metadata objects before generating new ones. In some cases, Velocity Studio may decide to modify an existing metadata object to fit the required properties, but this will only be done if the modified object is backward compatible with the original object. If a metadata object with the same name already exists in the specified namespace and it cannot be reused or modified, Velocity Studio will generate a new object that has the name of the existing object with a numeric suffix added for uniqueness.

Once the command is selected, The **Import XSD Schema** dialog appears.

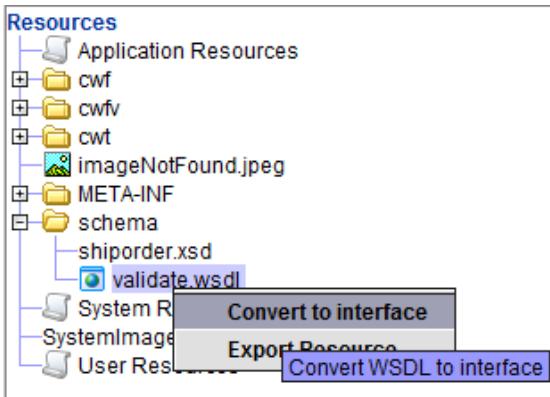


Field	Description
<b>Category</b>	One of <i>&lt;All&gt;</i> , <i>Data Structures</i> or <i>Data Types</i> . Allows the user to select what type of objects will be imported.
<b>Object(s)</b>	Allows the user to select objects that will be imported within the selected <b>Category</b> . If the selected object depends on other objects they

<b>to import</b>	will also be imported, regardless of whether they are selected or not.
<b>All</b>	When clicked, all the objects in the <b>Object(s) to import</b> list will be selected for import.
<b>None</b>	When clicked, all the objects in the <b>Object(s) to import</b> list will be deselected.
<b>Target namespace</b>	The namespace into which the objects will be imported. If <new> is selected then the <b>Create Namespace</b> dialog will open when the <b>Import</b> button is clicked.
	
<p>You can define the <b>Name</b>, <b>Label</b> and <b>Namespace</b> properties of the new namespace. When the <b>OK</b> button is clicked, the new namespace will be created in the metadata along with the corresponding objects. <b>Note:</b> If the importing XSD contains a <i>Target namespace</i> element, then the corresponding Namespace must be selected here. If the <i>Target namespace</i> element exists and a corresponding metadata Namespace does not, then the &lt;New Namespace&gt; will appear regardless of what is selected here.</p>	
<b>URI</b>	Shows the target namespace URI if it is defined in the XSD schema. In this case, if a namespace object with this URI already exists in the metadata, it will be pre-selected in the <b>Target namespace</b> field.
<b>Replace existing objects</b>	If checked, existing objects with the same name and type in the namespace will be replaced with the new one. This is useful when the same file is imported more than one time with different objects selected.
<b>Full Import</b>	If checked, imports all included XSD schemas (for example, external XSD schemas which are referenced will also be imported).
<b>Import</b>	When clicked, creates the selected objects in the metadata.
<b>Close</b>	When clicked, closes the dialog without any action.
<b>Help</b>	When clicked, opens a window with application help.

## WSDL Files

You can create SOAP interface metadata -- Data Structures (messages), interfaces with operations, bindings and external services with ports -- based on WSDL resource file, by right-clicking the resource node and select **Convert to interface**.



The XSD schema definitions inside the WSDL file will also be processed. The schemas must contain correct namespace definitions and should not refer to any outer schemas except the standard XSD Schema. If the objects defined by the XSD schema belong to namespace(s) that are different from the WSDL target namespace, the user will be prompted for the XSD namespace(s) name and label.

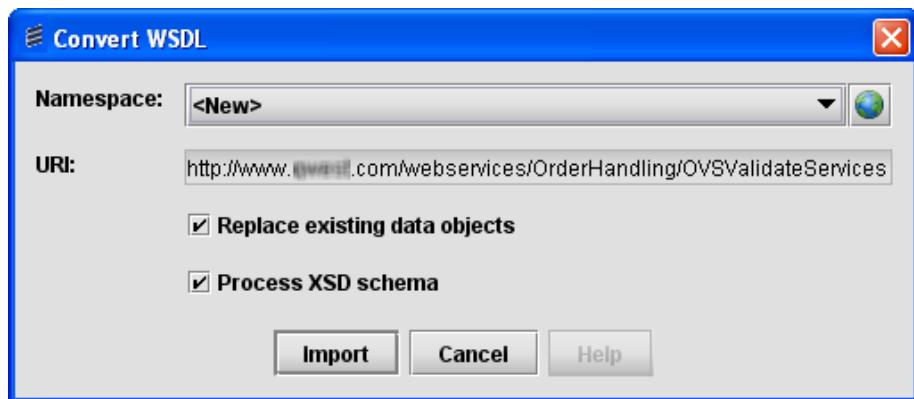
**Note:** This command is a helper function to accelerate development. Instead of manually creating the corresponding objects, the metadata developer can automatically generate them. When this command is used, certain assumptions are made that may not be valid in all cases. It is recommended that the developer carefully examines the generated objects and does appropriate changes where needed.

Velocity Studio always attempts to use the existing metadata objects before generating new ones. In some cases, Velocity Studio may decide to modify an existing metadata object to fit the new requirements, but this will only be done if the modified object is backward compatible with the original object. If a metadata object with the same name already exists in the specified namespace and it cannot be reused or modified, Velocity Studio will generate a new

object that has the name of the existing object with a numeric suffix added for uniqueness.

This command does not store location (address) information for **Port** metadata, which is stored in Configuration application. To do so, go to [Services](#) page in Configuration application to save the location information in the *Port* node's **Configuration** section.

Once the command is selected, The **Convert WSDL** dialog appears.



Field	Description
Namespace	The target namespace where the objects will be imported.
URI	Shows the target namespace URI if it is defined in the WSDL document. In this case, if a namespace object with this URI already exists in the metadata, it will be pre-selected in the <b>Namespace</b> field.
Replace existing data objects	When checked, existing objects with the same name and type in the namespace will be replaced with the new one. Useful when the same file is imported more than one time with different objects selected.
Process XSD schema	If checked, the XSD schema definitions from the WSDL file will be processed (the data types, object types and Data Structures will be created). If unchecked, they will be ignored.
Import	When clicked, creates selected objects in the metadata.
Cancel	When clicked, closes the dialog without any action.
Help	When clicked, opens a window with application help.

## Stylesheets

Velocity Studio recognizes the stylesheet file in the resource repository, and make the CSS available in **Stylesheet** drop-down list in managing Forms and Pages, as well as in **Style** property in Form Elements.

**Note:** If you are running Velocity Studio while placing a CSS file into the **resources** folder of your project directory, the project must be re-opened to load the CSS file.

## Archive Files

Runtime can read into content files in the JAR file. For example, if */dirA/sample.jar* contains resource with relative path */dirJAR/a.gif*, then this *a.gif* resource has the path */dirA/dirJAR/a.gif* relative to **Resource** folder of the project directory.

The **Resources** tab does not support browsing through JAR files, however.

This JAR file may also include a text file which predefines substitute association. This method is provided as a short-cut to manual substitution. This file is optional, and must meet all conditions below. The file **SystemImagesSub.txt** at project directory's resource folder is one such example.

- Must be placed in the root of the archive, and be a .txt file.
- Each line must indicate a URL, and URLs will be substituted in pairs: the first in a pair will indicate the original, and second will indicate its substitute.
- Commented lines starting with // will be ignored.

## System Image Substitution List

The file **SystemImagesSub.txt** stores the list of all image substitutions that is defined in the **Resource** tab, which is a text file with a pair of URLs in each line: the URL of the original, and then the URL of the substitute.

## Message Resources

Throughout Velocity Studio, messages are used to provide notifications or error messages. These messages are organized by Application, System, and User and can be viewed in the **Resources** folder of Velocity Studio. For each language implemented, there is an associated message resource node file. However, the default messages are written in the English language. The table below contains the default file names for the English language (en-xx[English]) and an associated description of the Message Resource Nodes:

File Name	Message Resource Node	Description
appRC_en-xx.xml	Application	Messages defined in user applications for use in scripting.
sysRC_en-xx.xml	System	Messages defined by the system for logging purposes only. These are rarely translated since only administrators will see these messages.
userRC_en-xx.xml	User	Messages defined by the system that are presented to the user.

## Message Translation

For each Metadata Root Language implemented within Velocity Studio, there is an associated Application, System, and User file created within the metadata **languages** folder. These files can be viewed within the **Resources** folder of Velocity Studio. For only the English (en-xx) language files, can the **Message** be modified. Once the English language System and User files are modified, the file will be stored in the **languages** folder of the metadata. For only the Application English language file (appRC\_en-xx.xml) can additional message records be added or deleted by clicking the associated buttons on the side bar as shown below:

Id	Message
CWTPC009	Code table must be selected.aaaaaaaa
CWTPC007	Can not exclude the same tax more than once.
CWTPC006	Can not activate the item, please activate all the dependent items first.
CWTPC005	Please add an attachment before save.
CWTPC004	No model updates required.
CWTPC003	Invalid Attribute Restriction field.
CWTPC012	Only active item(s) can be archived
CWTPC013	Group "(2)" requires {0} {1} selection(s)
CWTPC010	Please select the Attribute Value Type.
CWTPC011	Not eligible for product "(0)".
export/images	Images
CWTPC019	Can not apply tax on top of itself.
CWTPC018	Can not delete {0}. Object is active, archived or referenced from another object.
export/signature	Verify Signature
CWTPC015	Do you want to re-price this product?
CWTPC014	No pricing data found for "(0)"
CWTPC017	Can not delete the component group that has components
CWTPC016	Can not delete active or archived price version(s)
CWTPC021	Invalid date. start date must be prior to end date.

When a Metadata Root Language is changed from the default language (en-xx [English]) to another language, it is possible to translate the default English message to another language. By simply selecting the message node file from the Resources Tree directory, the associated language file appears and the user is then able to translate a message by double-clicking the **Message** column of the file. For the example shown below, the Application messages are being translated for the French/Canada language. The system continues to display the associated **Message ID** and its English translation in the **Default Message** column, while the translated message appears in the **Message** column.

It is not possible to add Application messages for any other language other than English. In this case, the user would have to switch the language back to English, add a new Application message record, and then modify the **Message** with the associated language file. The English language message appears as the Default Message.

Change Language → Language: fr-ca [French/Canada] Stylesheet: cwf.css [/cwf/css/cwf.css] Skin: Enterprise

ConceptWave

Id	Message	Default Message
CWTPC009		Code table must be selected.aaaaaaaa
CWTPC007		Can not exclude the same tax more than once.
CWTPC006		Can not activate the item, please activate all the dependent item(s).
<b>CWTPC005</b>	<b>Language translated message here...</b>	Please add an attachment before save.
CWTPC004		No model updates required.
CWTPC003		Invalid Attribute Restriction field.
CWTPC012		Only active item(s) can be archived.
CWTPC013		Group "{2}" requires {0} {1} selection(s).
CWTPC010		Please select the Attribute Value Type.
CWTPC011		Not eligible for product "{0}".
export/images		Images
CWTPC019		Can not apply tax on top of itself.
CWTPC018		Can not delete {0}. Object is active, archived or referenced from ...
export/signature		Verify Signature
CWTPC015		Do you want to re-price this product?
CWTPC014		No pricing data found for "{0}"
CWTPC017		Can not delete the component group that has components
CWTPC016		Can not delete active or archived price version(s)
CWTPC021		Invalid date_start date must be prior to end date.

Resources

- Application Resources
- cwf
- cwfv
- cwt
- imageNotFound.jpeg
- META-INF
- System Resources
- SystemImagesSub.txt
- User Resources

Metadata Library Resources

## Metadata Object Detail Pane

Once selected in the **Navigation** pane, the metadata objects' properties will appear in the **Detail** pane where they may be updated. Objects from the **Library** tab are read-only and cannot be changed.

The general layout of the **Detail** pane for all metadata objects is very similar. The **Detail** pane for a Data Type object is shown here.

Screenshot of the Metadata Object Detail Pane for a Data Type object. The pane is divided into several sections:

- General Tab:** Contains fields for Name (testDT), Label (testDT), Description, Help, Extends (com.conceptwave.system.String), Overrides (<NONE>), Length (<Unlimited> (<Inherited>)), Default value, XML name, Text format, and Element properties (<NONE>). It also includes checkboxes for Private, Restricted, Deprecated, Final, Nullable, and Encrypt.
- Properties Table:** A table showing element properties with columns for Name and Value.

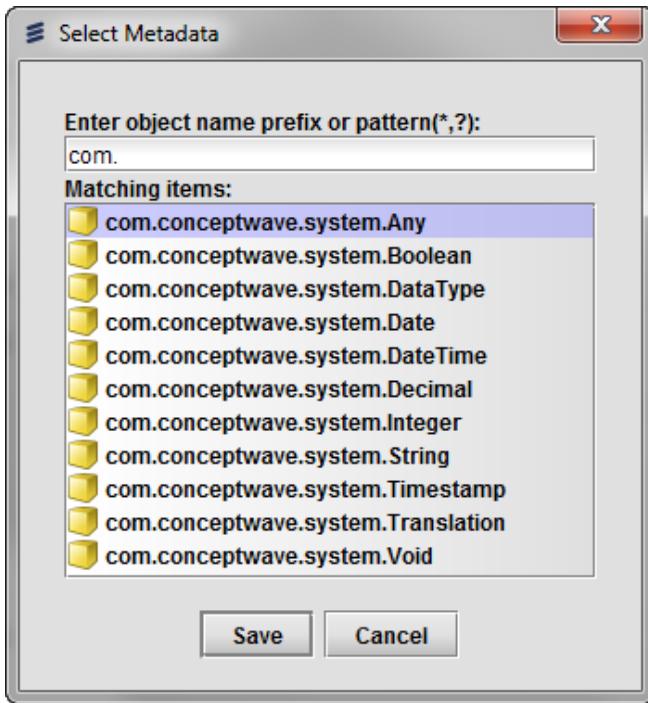
The following sections highlight some common notable features found in object pane.

### Property Tabs

The tabs at the top of the **Detail** pane allow navigation to the various groups of object properties. Clicking a tab displays the associated properties. Property tabs are different depending on the metadata type of the object.

### Metadata Lookup

The metadata lookup icon  is located at the end of any Metadata Object Selection field. If the field is editable, a Metadata Lookup dialog box appears when the icon is clicked:



Only those relevant to the field (that is, the field is allowed to reference such metadata objects) are listed in the **Matching items** list. The dialog box allows you to search for the metadata object, in various ways:

- Scroll up/down the list of matching metadata objects in the **Matching items** listbox, which is all the metadata objects qualified for selection based on this selection field's criteria. The full pathname (that is, prefixed with namespace path) of each qualified metadata object is shown. Select from this list and click the **Save** button to select the desired metadata object.
- Use the search field to key in refined matching criteria. With each keystroke, the **Matching items** listbox is updated by filtering pathname of metadata objects using the search key.
  - the search key is prefix-matching on pathname and name of metadata objects. For example, if search key is *sample*, then metadata objects such as *sampleNamespace.abc.documentA* and *anyNamespace.def.sampleDoc* would be matched.
  - the asterisk (\*) character can be used to wildcard match any character strings of any length. For example, *\*.system* would be to match metadata objects with any character strings followed by *.system* and followed by any character strings (because the search key is prefix matching), such as *com.conceptwave.system.anyObject*.
  - the question mark (?) character can be used to wildcard match any character. For example, *da?e* would be to prefix match metadata objects such as *date*, *datetime*, *dayeffective*.

Type in a fragment to shorten the matching list, such as namespace prefix, before finding your desired metadata object in the listbox.

If the Metadata Object Selection field is read-only, the properties of the metadata object in selection appears in pop-up when the metadata lookup icon is clicked. The pop-up is read-only.

**Properties**

**General**   **Methods**   **Enumeration**

Name:	String	<input type="checkbox"/> Private	<input type="checkbox"/> Restricted	<input type="checkbox"/> Deprecated				
Label:	String							
Description:	<input type="button" value="Edit"/>							
Help:	<input type="button" value="Edit"/>							
Extends:	com.conceptwave.system.DataType	<input style="width: 20px; height: 20px; vertical-align: middle;" type="button" value="..."/>	<input style="width: 20px; height: 20px; vertical-align: middle;" type="button" value="..."/>	<input style="width: 20px; height: 20px; vertical-align: middle;" type="button" value="..."/>				
Overrides:	<Any>	<input style="width: 20px; height: 20px; vertical-align: middle;" type="button" value="..."/>	<input style="width: 20px; height: 20px; vertical-align: middle;" type="button" value="..."/>	<input style="width: 20px; height: 20px; vertical-align: middle;" type="button" value="..."/>				
Length:	<Unlimited> (<Inherited>)							
Default value:	<input type="text"/>							
XML name:	<input type="text"/>							
Text format:	<input type="text"/>							
Element properties:	<NONE>							
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr><td></td><td></td></tr> </tbody> </table>					Name	Value		
Name	Value							
<input type="button" value="OK"/>								

### Manage Array Elements

Quite often, a metadata object contains an array of elements, and is managed with a tabular widget as shown below.

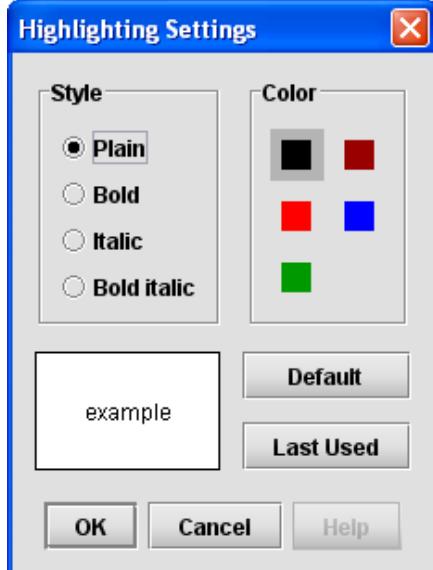
**General**   **Variables**   **Methods**

Name	Type	Methods	Vis	Opt	Edit
detail	User Interface	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
resultHover (Result Hover)	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
detailForm	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
result	Document	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
resultForm	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
finder	Finder	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
search	User Interface	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
searchForm	String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
isSearchFormVisible	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
invokeContext		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
result	Document	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
active	Boolean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
age	Integer, 3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

### Action Icon Toolbar

A column of action icons exists at the right of the table. The following table describes their functionalities. Note that not all tabular widgets have all of these icons. Only the applicable icons appear.

Icon	Description

 <b>Move Up</b>	Move the selected Element up one row in the table.
 <b>Move Down</b>	Move the selected Element down one row in the table.
 <b>Add Element</b>	Adds an Element to the table.
 <b>Remove Element</b>	Remove/delete the selected Elements from the table.
 <b>Sort</b>	Sort the Elements in the table in alphabetical order.
 <b>Highlight</b>	Highlight the selected Elements in another color. The <b>Highlighting Settings</b> Dialog appears, which has the same functionality as the <b>Highlight</b> command in right-click pop-up menu found in Navigation Pane.
 <p>The screenshot shows the 'Highlighting Settings' dialog box. It has two main sections: 'Style' and 'Color'. The 'Style' section contains radio buttons for 'Plain' (selected), 'Bold', 'Italic', and 'Bold italic'. The 'Color' section shows a 2x3 grid of color swatches: top-left is gray, top-right is dark red, middle-left is red, middle-right is blue, bottom-left is green. Below these are 'example', 'Default', and 'Last Used' buttons. At the bottom are 'OK', 'Cancel', and 'Help' buttons.</p>	
 <b>Cut</b>	Cut the selected Element.
 <b>Copy</b>	Make a duplicate copy the selected Elements in the table. The duplicated Elements has all the same properties as the selected Elements, but with the name appended with "1" (for example, <code>dateOfBirth</code> would be duplicated to <code>dateOfBirth1</code> ).
 <b>Paste</b>	Paste a selected Element that was either cut or copied.
 <b>Properties</b>	Shows the properties of the selected Element in a Dialog box.

**Note:** With the **Cut**, **Copy**, and **Paste** icons, you can move or copy object parts to other objects:

- Select variables (that is, leafs) of a document, and either copy-paste or cut-paste them to another document
- Select functions of an object, and either copy-paste or cut-paste them to another similar (document or UI) object
- Select nodes or leafs of a data structure on the same level, and either copy or move them to another data structure

## General Tab - Common Items

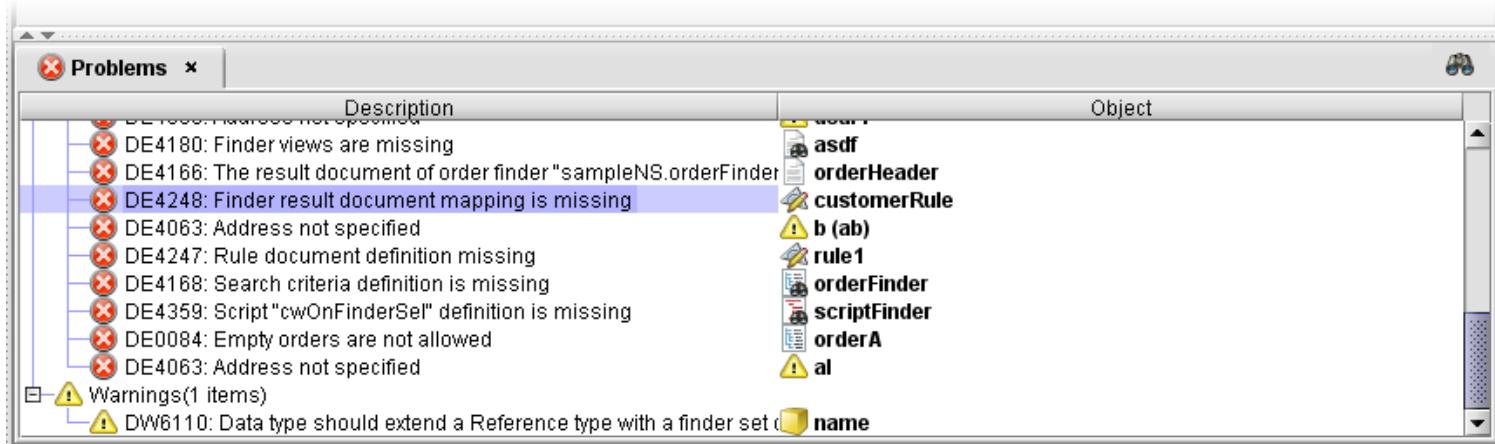
All metadata objects have a **General** tab that contains the common fields described in the table below.

Property	Description
<b>Name</b>	Mandatory. Unique name of the object. The <b>Name</b> must conform to JavaScript naming conventions (that is, begin with a letter and the following characters must be letters, numbers, or the underscore character).  For namespaces, the name must not match another namespace name and cannot be any of the following: Array, boolean, break, case, catch, class, const, continue, Dashboard, DataStructure, debugger, DecisionTree, default, delete, deprecated, do, Document, else, enum, export, extends, false, final, finally, Finder, for, function, Global, if, import, in, instanceof, int, new, null, Number, Object, optional, Order, private, Process, public, readonly, Reference, return, static, String, super, switch, this, throw, true, try, typeof, var, void, while, with, writeonly. For all other objects, the name must be unique within the namespace. Read-only after creation. To

	change it, use the <b>Rename</b> command in the popup menu by right-clicking the metadata object.
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (for example, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (for example, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Final</b>	Metadata elements can be marked as final, which includes any Boolean final property that is available on any metadata element. Additionally, any metadata element that has the <b>Final</b> property set to true can neither be extended, nor overridden.
<b>Label</b>	Mandatory. Visual label of the Metadata object. The label may include the full variety of characters, including punctuation. The label has the same restrictions in duplication as the <b>Name</b> field.
<b>Description</b>	Detailed description of the object used for documentation purposes.

## Problems Pane

The Problems pane is located at the bottom-right of Velocity Studio as the **Problems** tab, which contains all validation errors and warnings of all metadata in the project. You can expand the list of errors/warnings using the expand icon, and then double-click the individual error or warning to jump to the metadata object in question. You can then fix the problem at hand.



The Problems pane contains the following columns:

Column	Description
Description	The description of the validation error
Object	The metadata object that contains the validation error

The Problems pane groups messages by warning or error first, followed by the message code. You can also expand and collapse grouped errors and warnings as shown the previous example.

As metadata objects are created and updated by you, Velocity Studio performs validation checks on these updates and generate validation errors and warnings if any is failed. If there exists validation errors, metadata can continue to be edited and can also be saved. However, metadata cannot be [run](#) or [debugged](#). Other project functions such as to [build deployment WAR](#) or [library JAR](#), [check DB mapping](#), as well as [Upgrade System](#) are disabled. The [Debug Script](#) function in top-level Scripts is disabled as well. Therefore, **you must resolve all validation errors before running your metadata**. Warnings, on the other hand, do not disable any running command of metadata.

If the Problems pane is closed, click **Views > Problems** from the menu bar to open the pane again.

### Filter Results button

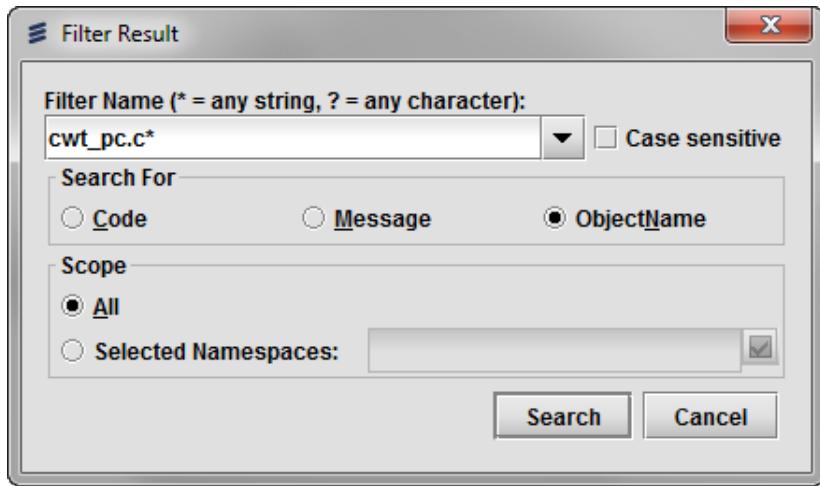
Clicking the **Filter Results** button ( ) allows you to select the following options:

- **Filter Result**, which launches the Filter Results dialog to filter the information in the Problem pane to suit your needs
- **Clear Filter**, which clears all information in the Problem pane

### Filter Result dialog

To use the Filter Result dialog to filter the information in the Problem pane, do the following:

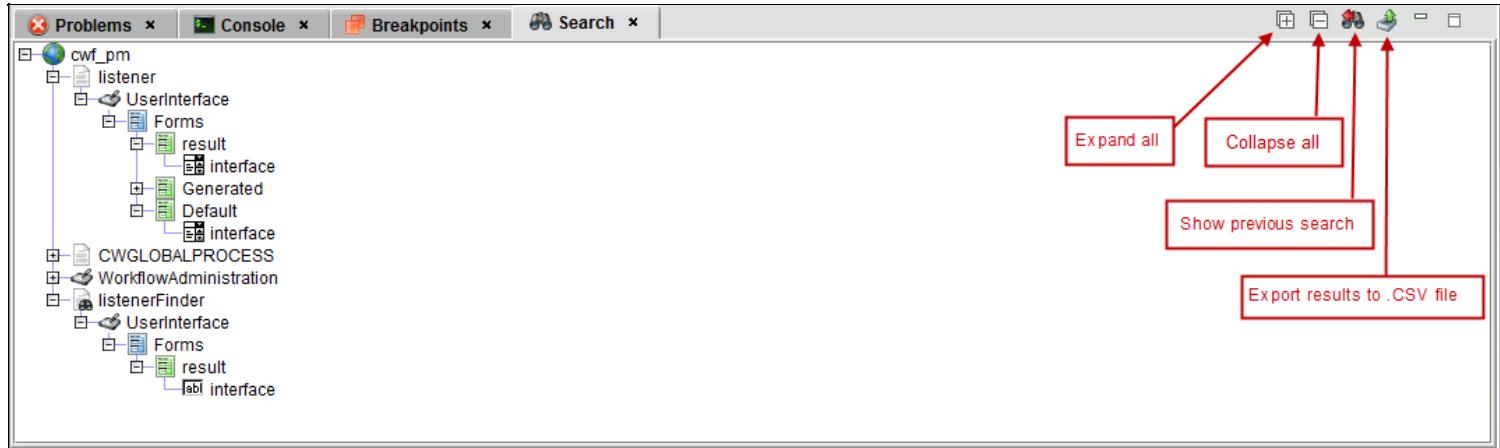
1. In the **Filter Name** field, enter your filter criteria. Wildcard (\*) and any character (?) are acceptable. In this example, filtering is for an object name using a wildcard.



2. In the **Search For** section, **Object Name** has been selected. You can also filter by **Code** or **Message**. If you leave the **Filter Name** field blank, and select either **Code** or **Message**, your results are filtered alphanumerically.
3. For the **Scope** section, the default is **All**. To narrow down your filtering, you can click **Selected Namespaces**:
  - a. Proceed to click the **Checkmark** icon to launch the Select Namespaces dialog and specify the namespaces that you want.
  - b. Click the **Save** button to return to the Filter Result dialog.
4. Click the **Search** button to filter your results, which appear in the Problem pane.

## Search Pane

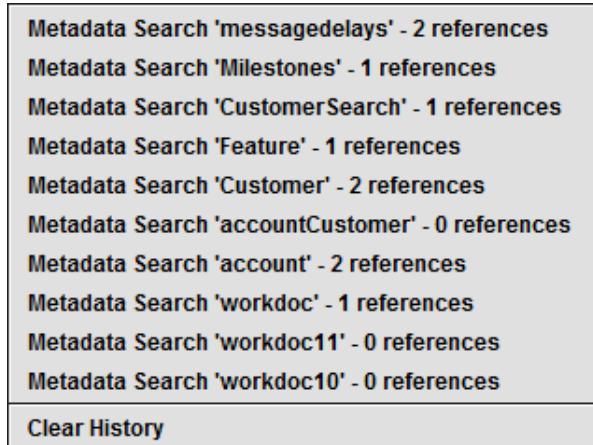
The search pane is located at the bottom-right of the Velocity Studio as the **Search** tab, which contains all matched results of a [search command](#). The search results are metadata objects listed in namespace hierarchy. You can double-click a metadata object to jump to that object in the navigation and detail panes. You can also use the Expand all and Collapse all buttons to expand or collapse the search results.



If the search pane is closed, click **Views > Search** from the menu bar to open it again. Alternatively, click [Search > Search...](#) from the menu bar to perform another search, and to open the search pane.

### Show History of Previous Searches

You can click the **Show Previous Searches** icon to display the last ten searches. The search history also includes the number of successful search references. To clear the search history, click the **Clear History** option.



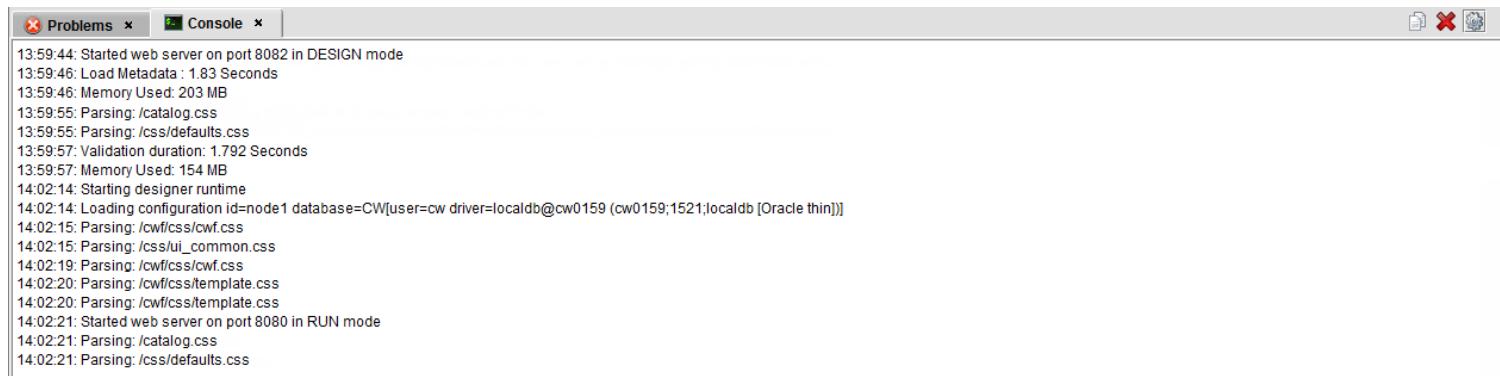
### Export Search Results to the .CSV File

To export the search results as a comma-separated values (CSV) file, do the following:

1. Right-click in the search pane and select the **Export** menu, or click the **Export results to .CSV** icon.
2. From the Export Search Results to File dialog, enter the name of the file.
3. Click the **Save** button.

## Console Pane

The Console pane is located at the bottom-right of Velocity Studio as the **Console** tab, which displays when you have opened a project, or shows system information of issued commands such as [Run](#) and [Debug](#).



A screenshot of the Velocity Studio interface showing the Console pane. The pane displays a log of events with timestamps and descriptions. The log includes:

```
13:59:44: Started web server on port 8082 in DESIGN mode
13:59:46: Load Metadata : 1.83 Seconds
13:59:46: Memory Used: 203 MB
13:59:55: Parsing: /catalog.css
13:59:55: Parsing: /css/default.css
13:59:57: Validation duration: 1.792 Seconds
13:59:57: Memory Used: 154 MB
14:02:14: Starting designer runtime
14:02:14: Loading configuration id=node1 database=CW[user=cw driver=localdb@cw0159 (cw0159;1521;localdb [Oracle thin])]
14:02:15: Parsing: /cwfcss/cwf.css
14:02:15: Parsing: /css/ui_common.css
14:02:19: Parsing: /cwfcss/cwf.css
14:02:20: Parsing: /cwfcss/template.css
14:02:20: Parsing: /cwfcss/template.css
14:02:21: Started web server on port 8080 in RUN mode
14:02:21: Parsing: /catalog.css
14:02:21: Parsing: /css/default.css
```

If a CSS parsing error occurs, Velocity Studio logs the error with its filepath in the Console pane. The Console pane also displays information about the duration of loading, validating, and starting runtime, and memory usage.

Velocity Studio has an embedded Web server to support itself as an Integrated Development Environment (IDE). One of the important system information details displayed in the **Console** is starting and stopping of the Web server in a specific *mode*, and on a particular port, as shown in the previous screenshot. There are several modes that a Web server port can be binded to:

Mode	Description
RUN mode	Indicates the project's metadata is running by AVM at this port. In RUN mode, the <a href="#">Configuration application</a> and <a href="#">System Administration application</a> are available.
CONFIG mode	Indicates the Configuration application is running at this port. System Administration App or project's metadata are not available.
HELP mode	Indicates the <a href="#">product documentation</a> is running on this port.

If the Console pane is closed, use the menu **Views > Console** to reopen the Console pane.

### Console pane buttons

The following table describes each **Console pane** button and its action when you click it:

Button	Description
	Click the <b>Copy</b> button to copy all information in the Console pane. You can then paste the information into an editor, such as Notepad, and save the information as a file.
	Click the <b>Clear All</b> button to clear all information in the Console pane.
	Click the <b>Auto-Scroll</b> button to have the information in the Console pane automatically scroll whenever new information appears.

**Note:** You can also right-click the Console pane area to access these actions. By default, the Auto-Scroll feature is on.

## Database Mapping Pane

The Database Mapping Pane is located at the bottom-right of Velocity Studio as the **DB Mapping** tab, which displays all warnings and errors of [Check DB Mapping command](#). You can expand the list of errors/warnings using the expand icon, and then double-click the individual error or warning to jump to the metadata object in question. You can then fix the DB mapping problem at hand.

The screenshot shows the Velocity Studio interface with the DB Mapping pane open. The pane has three tabs: Problems, Console, and DB Mapping. The DB Mapping tab is selected. It contains a table with two columns: Description and Object. The Description column lists errors and warnings, and the Object column lists the corresponding metadata objects. A specific warning about the 'Name' field being unmapped is highlighted with a blue selection bar.

Description	Object
Errors(1 items) DE0509: Table "CWPUSRPRTCPNTS" does not exist	CWPUSRPRTCPNTS (Participant Users)
Warnings(20 items) DW0515: Database column is larger than field "Event External Cc" DW0513: Field "Group Name" is not mapped to database DW0513: Field "Name" is not mapped to database DW0513: Field "Active" is not mapped to database DW0513: Field "Manager" is not mapped to database DW0513: Field "YEAR" is not mapped to database DW0513: Field "CALENDAR" is not mapped to database DW0513: Field "DAY" is not mapped to database	mappingForProductProperties (Mapping for product properties) user_group (User or Group) user_group (User or Group) user_group (User or Group) user_group (User or Group) calendarYearByType (Calendar Year By Type) calendarYearByType (Calendar Year By Type) calendarYearByType (Calendar Year By Type)

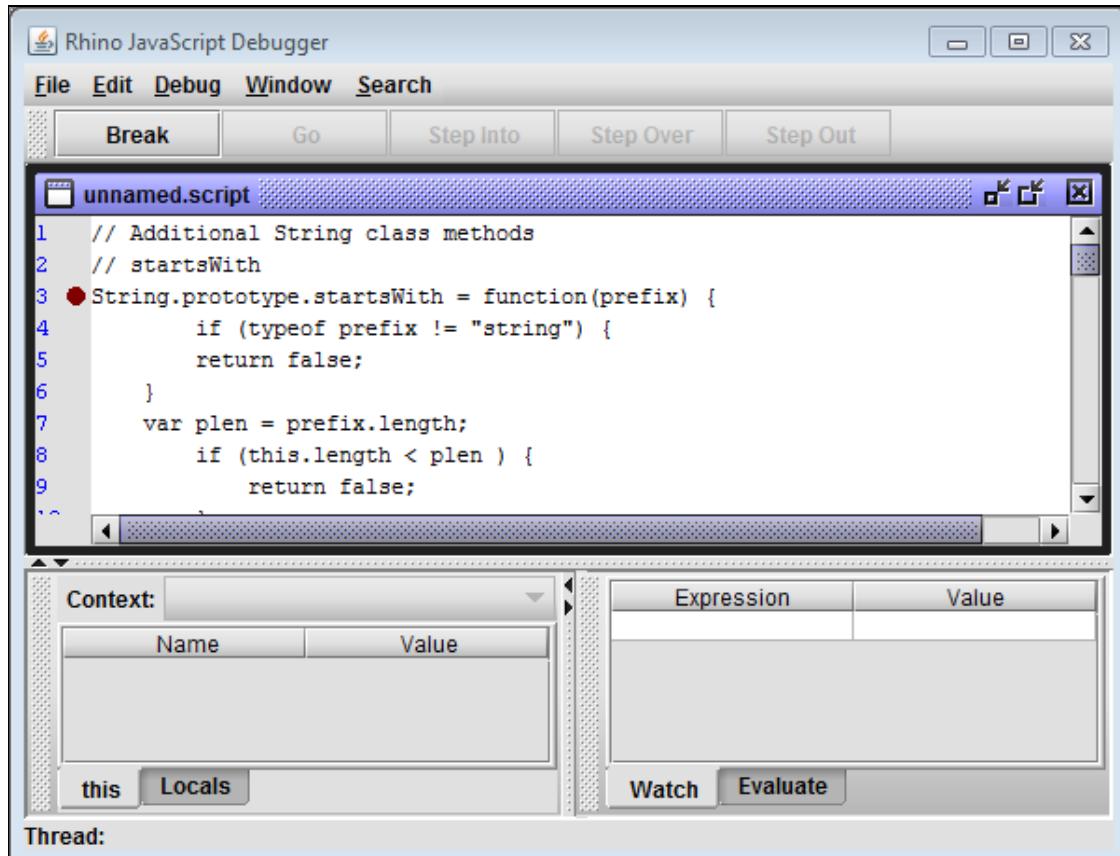
Column	Description
Description	The description of the mapping error
Object	The metadata object that contains the mapping error

If the DB Mapping Pane is closed, use the menu **Views > DB Mapping** to open the Search Pane again. Or, click [Database > Check DB Mapping](#) from the menu bar to perform another database mapping check to open the pane.

## Breakpoints Pane

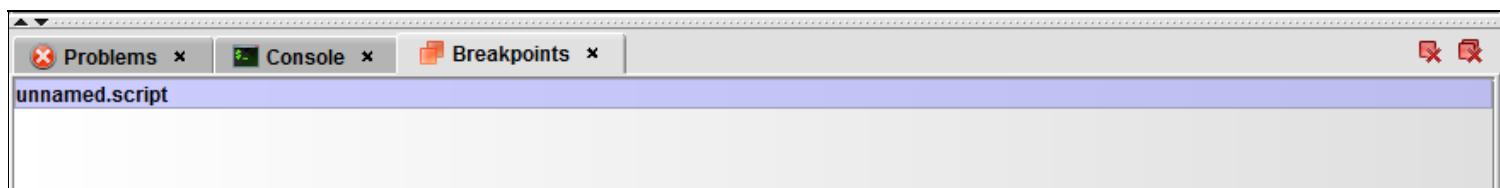
The Breakpoints Pane is located at the bottom-right of Velocity Studio as the **Breakpoints** tab, which provides a list of all scripts in Velocity Studio that have breakpoints set.

In this example, breakpoints have been set in `unnamed.script` using the JavaScript debugger.



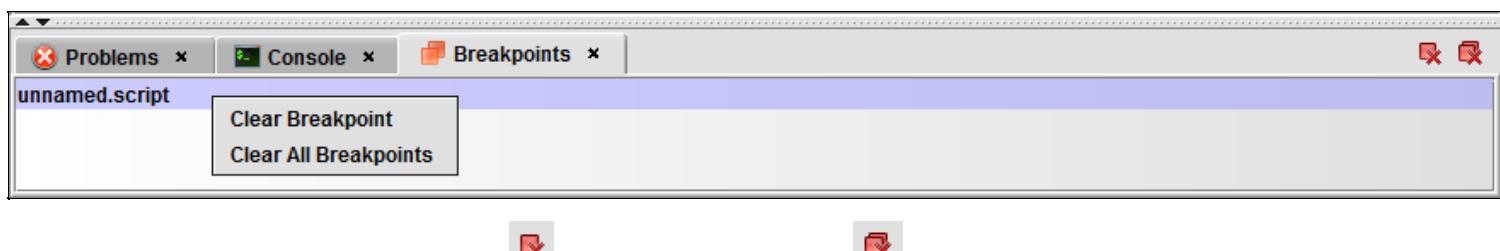
The script name also appears in the Breakpoint pane, listing it as a script that contains at least one breakpoint.

**Note:** You can double-click the script link in the Breakpoints tab, to open the script and set the focus to the corresponding object or function.



When you close the JavaScript debugger, you can clear breakpoints in a script. Right-click the script that you want and select from one of the following options:

- **Clear Breakpoint** to clear a specific breakpoint set in your script.
- **Clear All Breakpoints** to clear all breakpoints set in your script.



Alternatively, you can click the **Clear Breakpoint** ( ) and **Clear All Breakpoints** ( ) buttons.

For more information on using breakpoints and debugging your metadata, see the [Debug](#) section.

## Common Actions Outside Velocity Studio

---

There are various tasks where project manipulation is done outside of Velocity Studio. The following lists and describes the actions necessary to accomplish these tasks.

### Make a Copy of a Project

To make a copy of the project, you can simply copy the Project Directory to another location. Close the project if it is already opened in Velocity Studio, so that the project lock file `.cw.inuse.lock` is not present when copying.

To aid the transfer of the project files (for example, through FTP, e-mail, and so on), compress the Project Directory to a zip file before transferring. The recipient can then unzip before working on the project in Velocity Studio.

On a related note, the Velocity Studio supports [exporting the project as JAR file](#), but only as a library to be included in another project.

### Version Control on a Project

In a typical engineering environment for building enterprise applications, you are expected to be sharing development work in a team, and centralize your project work with a version control system, such as [Subversion](#) (that is, SVN). Simply create version control repository with Project Directory, and check-in / check-out files in the directory for metadata, library, and resource changes.

Refer to [Understanding Your Metadata Files](#) on the nature of the folders and files in the project, and [Metadata Objects Overview](#) on metadata files, which are stored as XML. If multiple team members work on the same metadata object, a conflict may occur in version control. Since product version 5.x, Velocity Studio no longer provides metadata merging tool. Instead, use a third-party tool or manually merge the XML file. Be careful to the integrity of the XML file in this case (for example, matching tags).

Depending on your team's environmental setup (such as local versus centralized database), you may or may not want to include `.settings` file in Project Directory's root folder.

The following are some best practices on managing project files and folders in a version control system:

- **Keep your source files in source control**

It is helpful to check in a copy of each source file that you need to build your project.

- **Ensure that you are working with the latest version of your file or metadata**

Follow the file check-out procedures that your source-control administrator has specified or that your team has established. Those procedures help make sure that you are using the latest copies of each file, meaning that you have the latest changes from other members of your team.

As other team members make changes to files in your project source control repository, the copies of those files that you have in your workspace become outdated and require that you update them. Because source control automatically assumes that the last checked-in version of a file is the newest version, checking in a stale version of a file after somebody else has checked in a newer version creates extra work. It also increases the risk that you will lose some important changes to the file.

- **Check out only the files that you need**

Check out only the files that you plan to change and work on. Do not check out an entire folder unless you plan to change every file in that folder.

- **Check in promptly**

It is not recommended that you leave your files checked out any longer than necessary. Check your files in as soon as you are done changing and testing them, ensuring that your teammates have access to the latest versions of the files.

- **Write good check-in comments**

When a problem occurs, you can use good check-in comments to help identify where your metadata went wrong and how to fix it more quickly. Even if nothing goes wrong, you can quickly identify what changes you made and why you made them.

### Import Metadata

To import parts or all application metadata of one project into another project, you must first [export Metadata as a library JAR](#), and [import it](#) into the target project as a library. Then, copy the desired metadata that is in the **Library** tab of **Navigation Pane** to application metadata *using Velocity Studio's Copy and Paste command* (in right-click pop-up menu).

You shall not arbitrarily copy folders and XML files in Project Directory's metadata folder into another Project Directory via file system. There exists a hashvalue in each metadata XML file that hashes, and thus protects, the relative filepath (to the project directory), including the filename. Thus, you must instead use rename, cut/copy and paste functions in Velocity Studio.

## Manage Resources

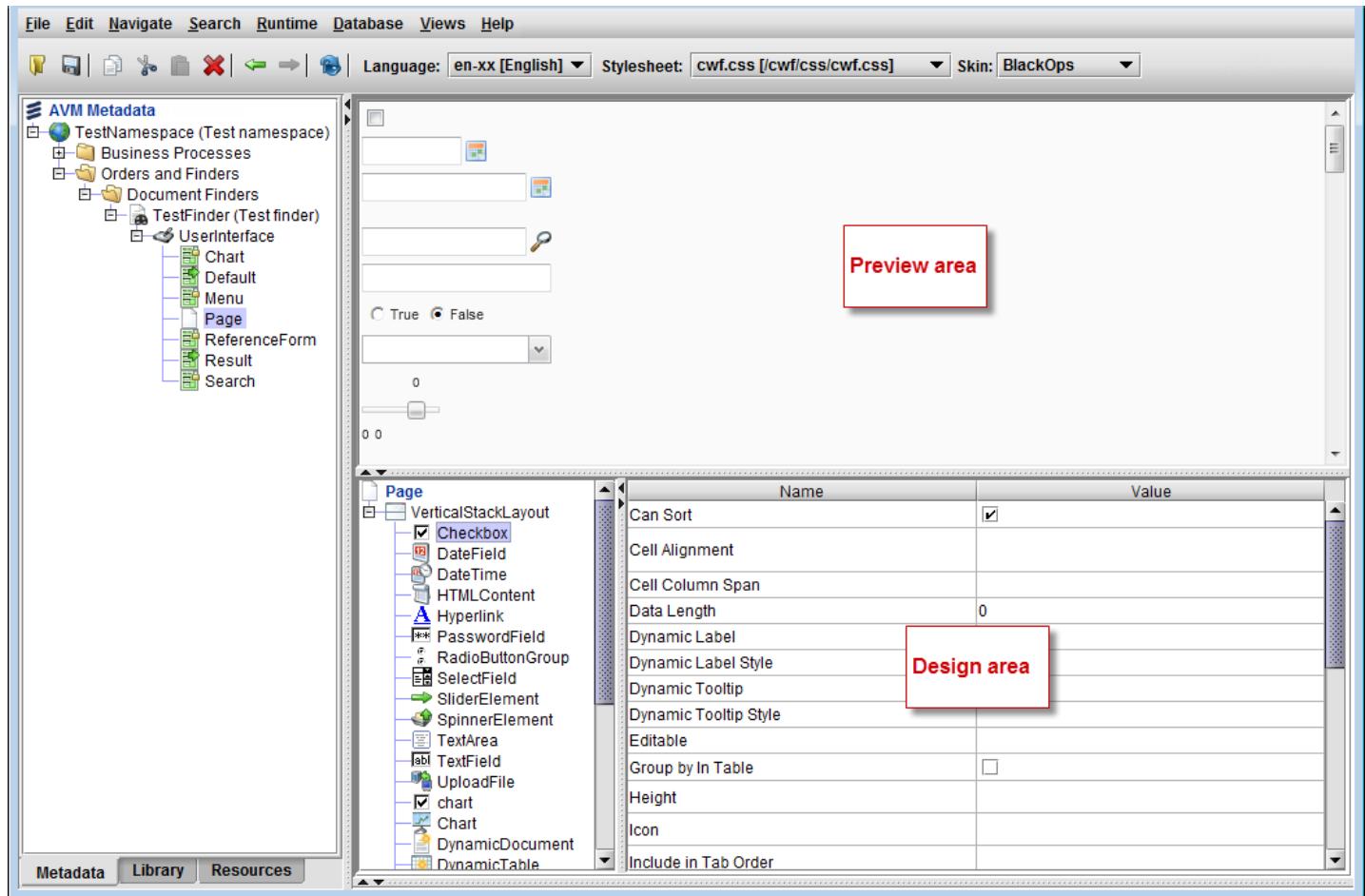
In general, with the exception of [XSD](#) and [WSDL](#) files, resource files are imported into project by simply adding them into Project Directory's **resources** folder through the file system. Similarly, they are removed by simply deleting them in file system.

The **Resources** tab in **Navigation Pane** provides navigational and other functionalities per file type; please see [its documentation](#) for details in supported file types and functionalities, such as Image files, XSD, WSDL, CSS and even JAR files. Indeed, you can package resource files into JAR files and add them into resources folder; Velocity Studio is able to read into content files in the JAR, at both design time and runtime.

For CSS, Velocity Studio does not provide editor to customize styles in the stylesheet. You can text-edit or use a third-party CSS editor for managing .css files in resources.

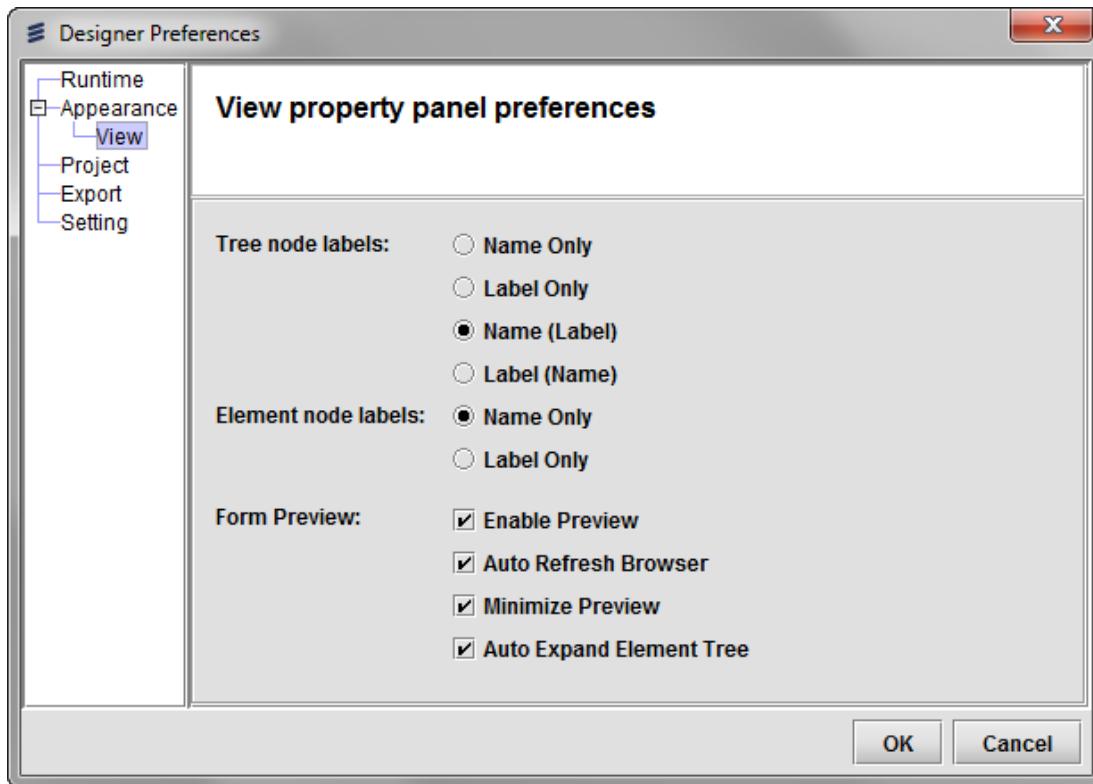
## Minimize Velocity Studio's Preview Area

By default, when you are developing your metadata in the design area, the preview area allows you to see your changes as you go. Velocity Studio allows you to minimize the preview area, providing you with a full design area.



To minimize the preview area, complete these steps:

1. Click **File > Preferences** from Velocity Studio's menu bar.
2. From the left menu, click **Appearance > View** in the Designer Preferences dialog.
3. Select the **Minimize Preview** checkbox and then click the **OK** button to save your change.



- When you resume designing your metadata, only the design area appears.

The screenshot shows the AVM Metadata editor interface. The top menu bar includes File, Edit, Navigate, Search, Runtime, Database, Views, and Help. The toolbar below the menu bar includes standard icons for file operations like Open, Save, Print, and Undo/Redo. The header also displays Language: en-xx [English], Stylesheet: cwf.css [/cwf/css/cwf.css], and Skin: BlackOps.

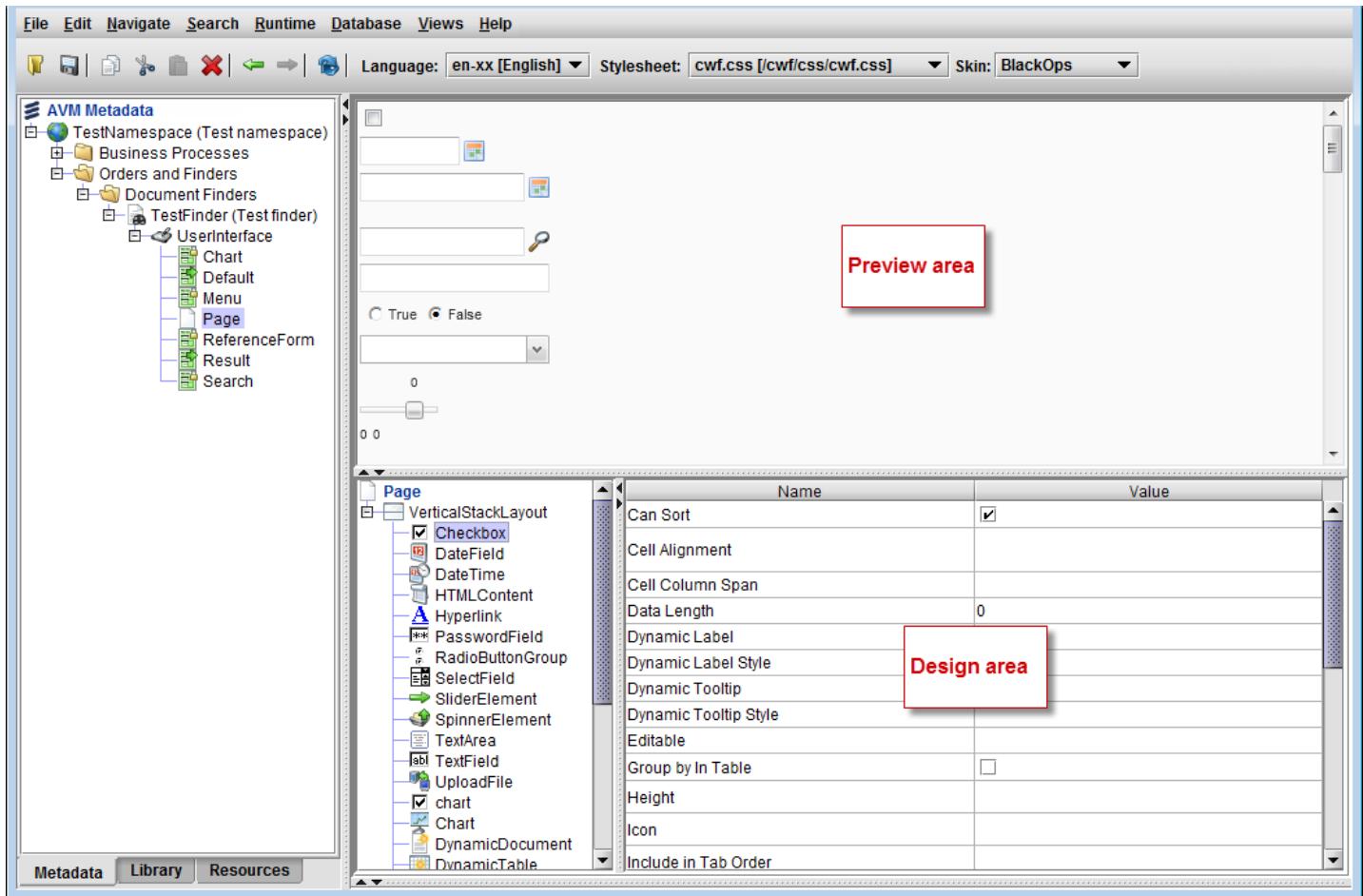
The left side features a tree view under 'AVM Metadata' labeled 'TestNamespace (Test namespace)'. It lists Business Processes, Orders and Finders, Document Finders (with a sub-node 'TestFinder (Test finder)'), and several UI components: Chart, Default, Menu, Page, ReferenceForm, Result, and Search.

The central and right sections of the interface are currently empty, indicating that the design area is active.

You can still access the preview area by dragging the top horizontal separator down and adjusting the preview area to the size that you require.

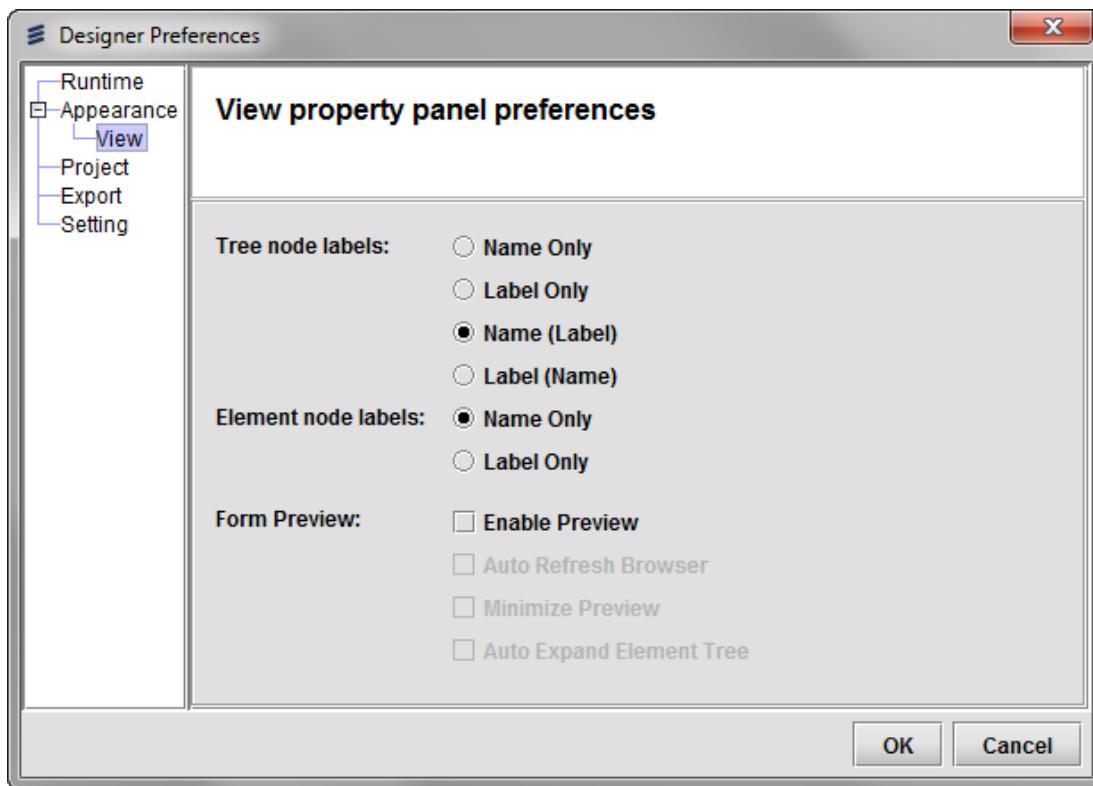
## Disable Velocity Studio's Preview Area

By default, when you are developing your metadata in the design area, the preview area allows you to see your changes as you go. Velocity Studio allows you to disable the preview area, providing you with a full design area.



To disable the preview area, complete these steps:

1. Click **File > Preferences** from Velocity Studio's menu bar.
2. From the left menu, click **Appearance > View** in the Designer Preferences dialog.
3. By default, the **Enable Preview** checkbox is selected. Deselect this checkbox to disable the preview area, and then click the **OK** button to save your change.



**Note:** Deselecting the **Enable Preview** checkbox disables all preferences in the Form Preview section.

- When you resume designing your metadata, only the design area appears.

Unlike having the **Minimize Preview** checkbox selected, which allows you to still access the preview area by dragging the top horizontal separator down and adjusting the preview area to the size that you require, deselecting the **Enable Preview** checkbox does not provide you with this horizontal

separator.

# Velocity Studio Menu Bar

---

The Velocity Studio menu bar appears at the top of the window and contains the following items:

## File

The **File** menu allows you to do the following tasks:

- Create a new project
- Open a project and recently opened projects
- Save a selected metadata object or all changed objects within your metadata
- Import and export files in various formats
- Customize skins
- Change your Velocity Studio preferences
- Exit the Velocity Studio application

## Edit

The **Edit** menu allows you to cut, copy, paste, and delete metadata objects. You can also access the Independent Script Editor from this menu.

## Navigate

The **Navigate** menu item allows you to open metadata objects using a [metadata name search](#) dialog. You can also navigate to the previous or next object in the metadata object selection history.

## Search

The **Search** menu has the following options:

- Perform a search in your project
- Search for an API in your metadata project scripts
- Replace an API with another one in your scripts, except in modules
- Replace object references in your metadata
- Replace deprecated system methods in your project

## Runtime

The **Runtime** menu provides you with the following actions:

- Run, stop, and debug your project's metadata
- Build a deployment WAR file
- Build a library WAR file
- Clean process revisions

## Database

The **Database** menu lets you perform the following actions:

- Connect and disconnect from the database
- Check your database mapping and map documents to the database
- Generate an SQL file to upgrade your system
- Import procedures and database schema

## Views

The **Views** menu allows you to open the following tabs:

- Problems
- Search
- Console
- Database mapping
- Breakpoints

## Help

The Help menu allows you to access the online help documentation, and view product and licence details.



# Velocity Studio - File Menu

---

The Velocity Studio **File** menu contains the following menu items and brief descriptions of each menu item:

## New > New Project

Creates new project

## Open Project

Opens an existing project from file system.

## Compare

Compare two sets of metadata.

## Save

Saves the selected metadata object in file.

## Save All

Saves all changed metadata objects in file.

## **Import**

The Import menu contains the following submenu options:

- Translation  
Import Language Translation resource file in Excel format.
- WSDL  
Import WSDL file.
- XSD  
Import XSD file.
- XML Data  
Creates a metadata Data Structure from an XML file.
- Rule Instance  
Import existing rules instances from an XML file.

## **Export**

The Export menu has the following submenu options:

- Translation  
Export Language Translation resource file in Excel format.
- Documentation  
Exports metadata HTML documentation.
- Reference  
Exports metadata reference HTML documentation.
- Rule Instance  
Export existing rules instances to an XML file.

## Customize Skin

Allows you to copy an existing Skin and rename it to a new name in the application metadata under the *skins/* folder.

## Preferences

Shows dialog to manage Velocity Studio and project preferences.

## [List of Recently Opened Projects]

Opens a project from the list of recently opened.

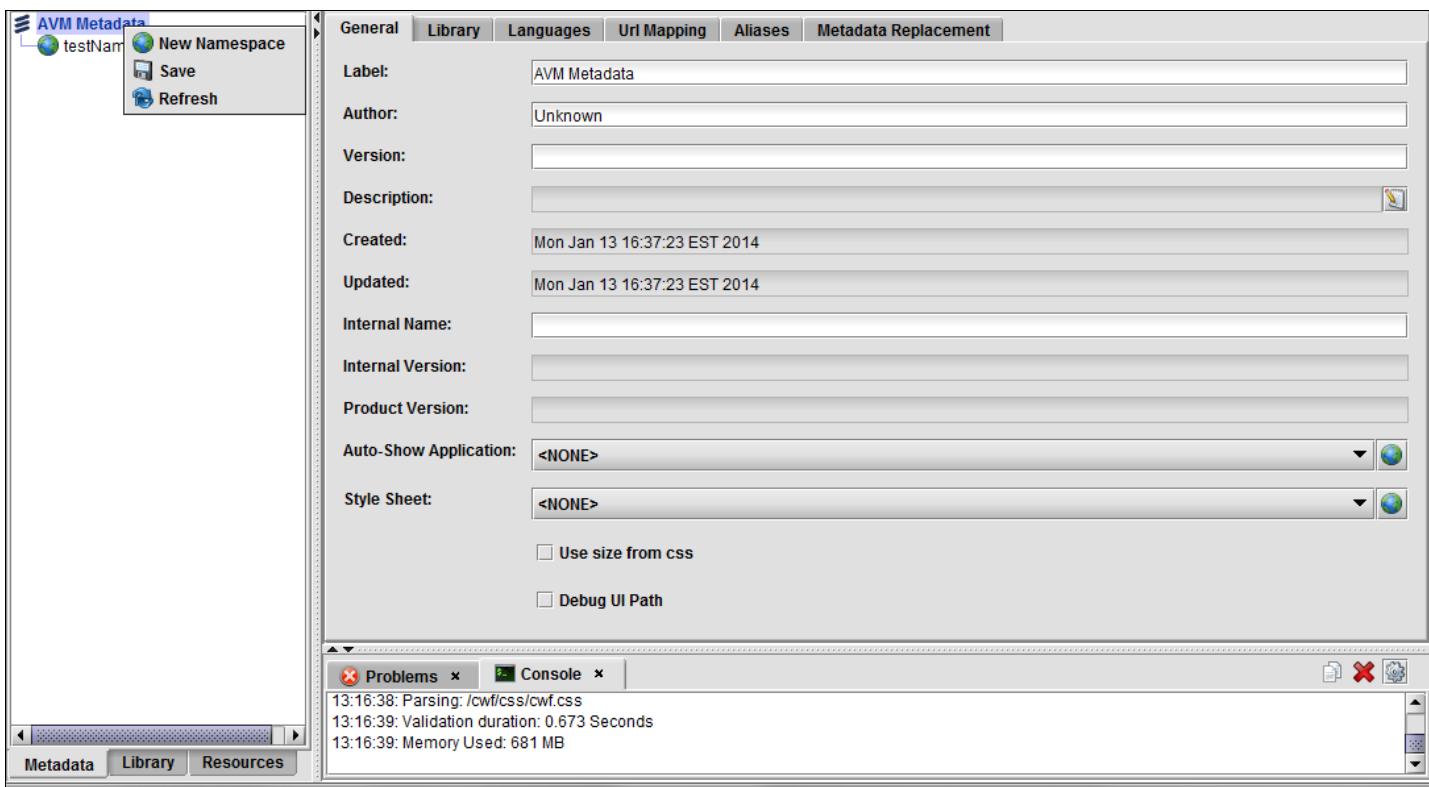
## **Exit**

Exits the Velocity Studio.

## New Project

To create a new project, complete these steps:

1. From the menu bar, click **File > New > New Project**.
  2. The Select an empty directory dialog opens. Select the root project directory and then click the **Save** button.
  3. The directory selected to create the project should be empty. If it is not, a prompt appears, asking whether you want to override the existing contents. Clicking the **Yes** button indicates that any directory content that impedes creating the project directory is overwritten. Otherwise, clicking the **No** button cancels creating a new project.
- In the selected directory, several files and folders are created to set up the project directory. See [Understanding Your Metadata Files](#) for details of these folders and files.
4. After you have created your new project, namespaces and other metadata objects may be added through right-click pop-up menus available in the **Metadata** tab of the navigation pane. For descriptions of the types of metadata objects available and their properties, refer to the [Metadata Objects Overview](#).



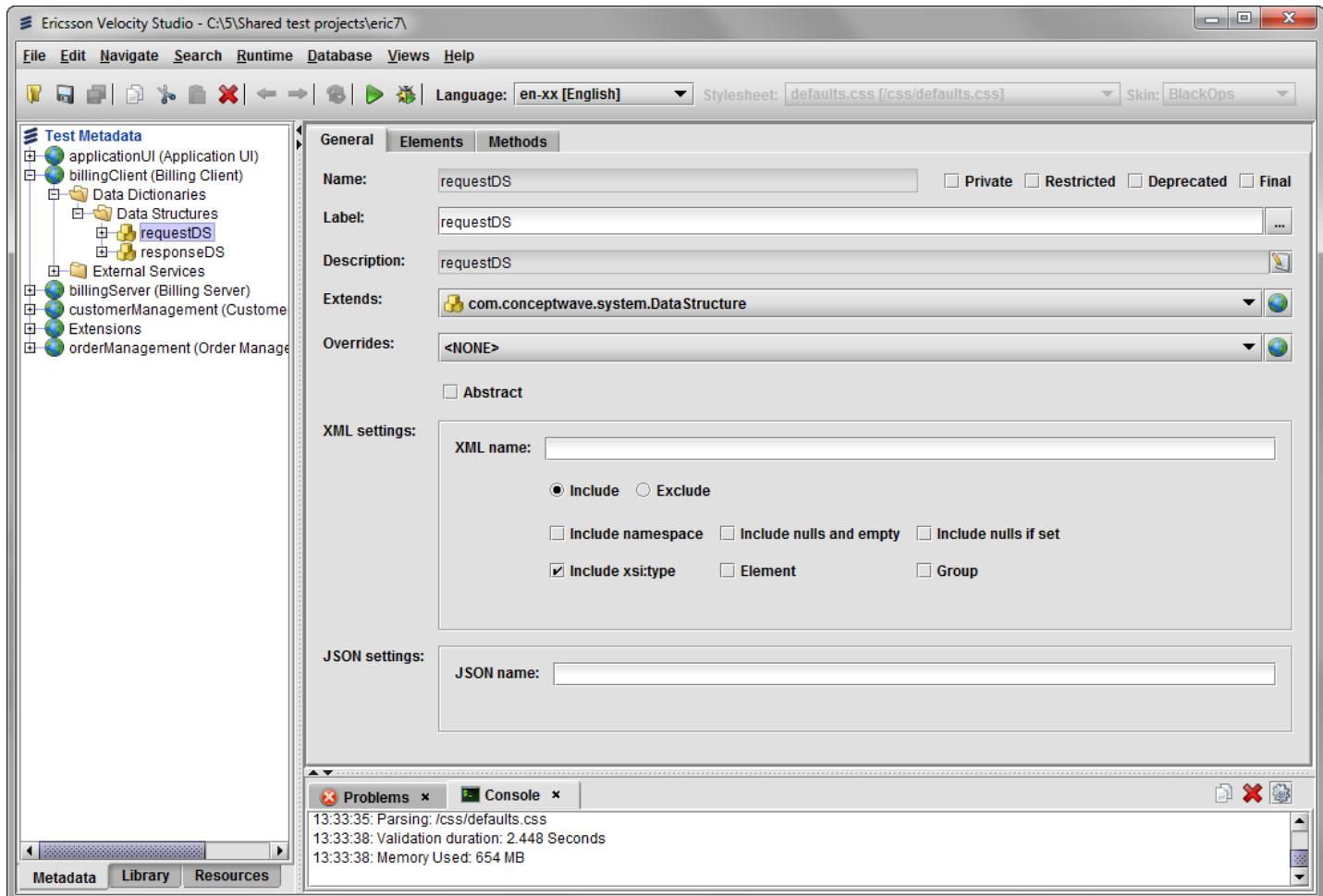
## Open Project

To open an existing project, complete these steps:

1. From the menu bar, click **File > Open Project**.
2. The Open Project Dialog opens. Select the root project directory and then click the **Open** button.

**Note:** If you open a project that does not have a metadata header file, an error appears. Clicking the **OK** button closes the error dialog and allows you to select an existing project from the Select Designer project directory.

3. While the project is opening by Velocity Studio, the **Opening Designer Project** popup appears, and disappears when the operation is finished.

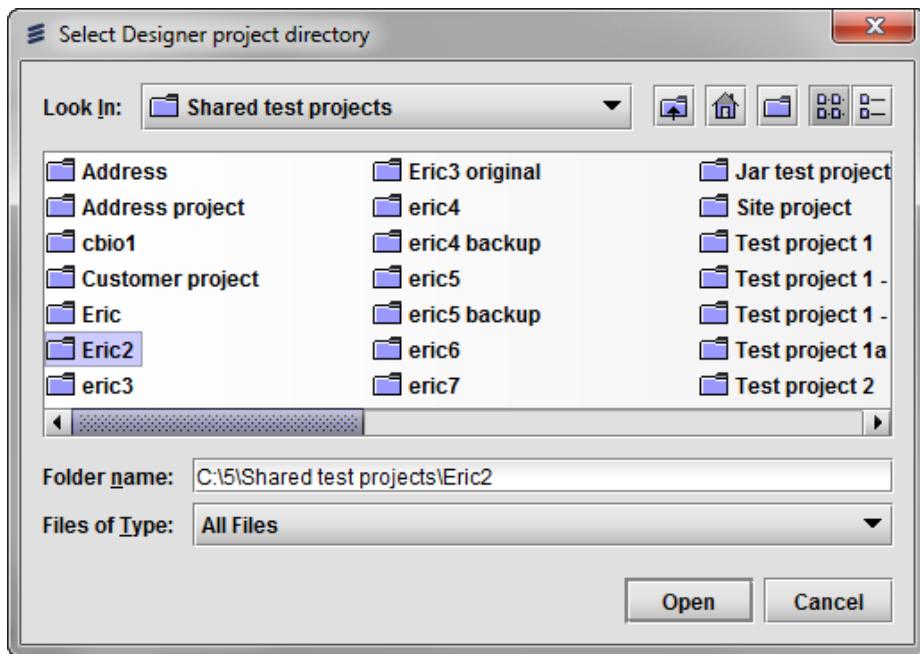


**Note:** You can read the console information that relates to opening the project using the [Console view](#).

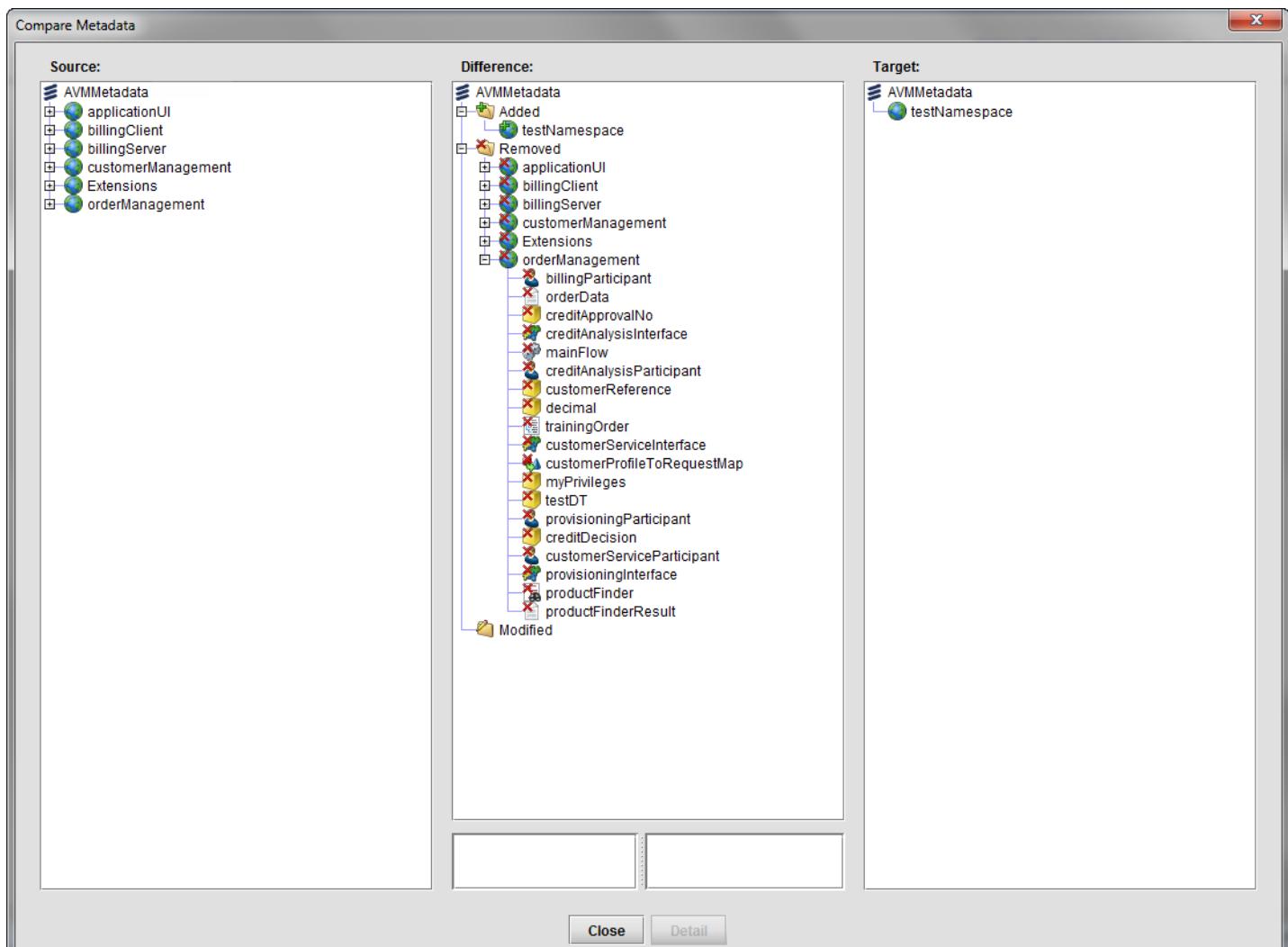
## Compare Metadata

Velocity Studio allows you to view the differences between two metadata projects. To use the metadata comparison tool, complete these steps:

1. In Velocity Studio, ensure that you have [opened a project](#) that will be compared with another project.
2. From the menu bar, click **File > Compare**.
3. A dialog appears, prompting you to select the directory in which your second project is located. Once you have selected your project directory, click the **Open** button to launch the **Compare Metadata** page.



4. The **Compare Metadata** screen appears:



The page contains the following sections:

Pane	Description
Source	This pane shows the current project's metadata tree. You can right-click any metadata object and select <b>Properties</b> to view its properties.
Difference	This pane contains a metadata tree containing three nodes: <ul style="list-style-type: none"> <li>o <b>Added</b> - node contains all top-level metadata that does not exist in the current metadata project.</li> <li>o <b>Deleted</b> - node contains all top-level metadata objects that no longer exist in the secondary metadata project.</li> <li>o <b>Modified</b> - node contains all top-level metadata objects that are different in the two projects. You can expand each top-level metadata object node in the Modified pane to view which of its children are different. Specifically, this pane contains all modified property elements of top-level metadata objects, including children, methods, and the user interface.</li> </ul>
Target	This pane shows the secondary project's metadata tree. You can right-click any metadata object and select <b>Properties</b> to view its properties.

#### Notes:

- It is not possible to have two simultaneous AVMs. The source metadata is located in the first AVM that was created, which refers to objects in the second AVM that was created.
- If no projects have been opened in Velocity Studio, the **Compare** option under the **File** option is disabled.
- When you select a modified property element, the old and new values appear in the detail panel under the **Difference** pane.
- When you select a modified element, the **Detail** button is enabled. Click this button to open the **Detail** panel, showing both the old and new values in a split screen.



## Save

---

Click the **File > Save** menu command to save the currently selected metadata object in **Metadata** tab of **Navigation Pane** (the command is disabled if **Navigation Pane** is not selected at **Metadata** tab).

The metadata object is written into the corresponding XML file in Project Directory. If the metadata object has been edited in Velocity Studio, and thus marked with a *pencil* stamp at the top left of the metadata object's icon, the "dirty" mark is cleared upon saving. For example, a dirty Document that shows becomes upon saving.

This menu command is the same as found in [Toolbar](#) or in [right-click pop-up menu](#) in Navigation Pane.

Note that creating or deleting a metadata object does not require saving; the XML file is added or removed immediately in Project Directory. See [documentation of managing metadata in Navigation tab](#), which lists what actions on metadata require saving, or not.

## Save All

---



Click **File > Save All** from the menu or click the **Save All** button ( ) from the menu bar to save the all application metadata objects in the project. You can also press the *Ctrl + Shift + S* key combination to save all changed metadata objects. This button is disabled when no changes have been made to your metadata objects.

Upon using this command, all metadata objects are written into the corresponding XML files in Project Directory. All metadata objects that have been edited in Velocity Studio, and thus marked with a *pencil* stamp at the top left of the metadata object's icon, the "dirty" marks are cleared upon saving. For example, a dirty Document that shows becomes upon saving.

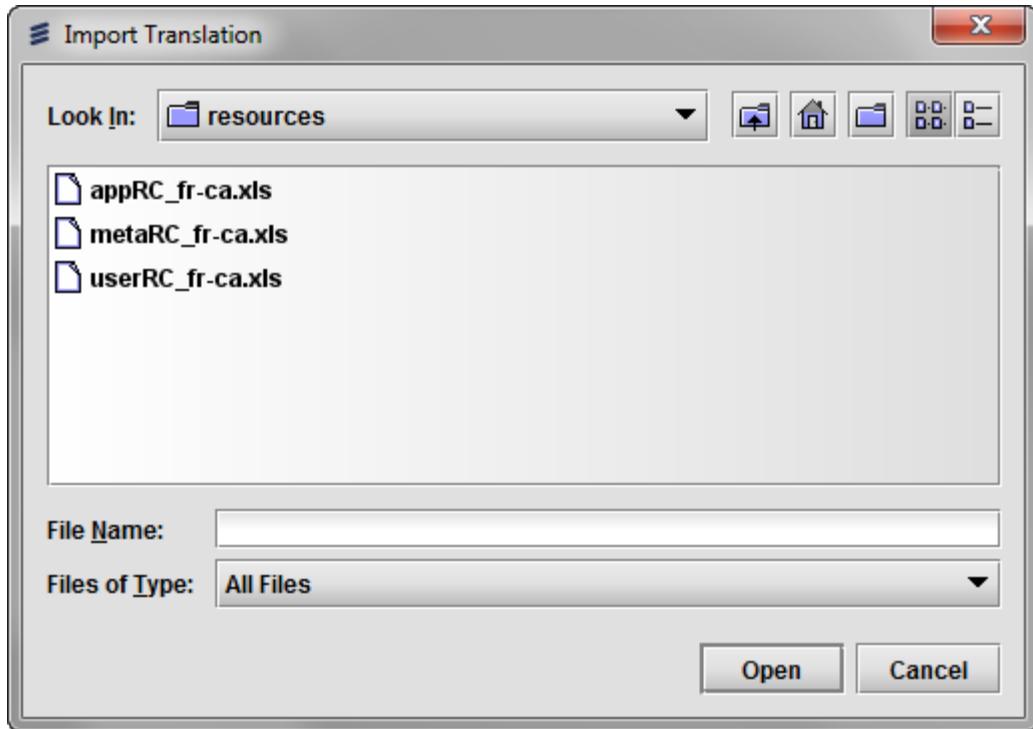
Note that creating or deleting a metadata object does not require saving; the XML file is added or removed immediately in Project Directory. See [managing metadata in Navigation tab](#), which lists what actions on metadata require saving, or not.

## Import Translation

The purpose of [Translation](#) is for defining all fields and labels visible in the runtime application in a specific language. You can import Excel files that contain Translation records of a specified language in the project with the **File/Import/Translation** menu.

With this menu command, you can import Translation that has been previously [exported](#) and subsequently edited using the Excel application.

Upon invoking the menu command, the **Import Translation** dialog is displayed. Select an Excel file to import. For the multiple Excel files that were exported, you need to import each of them separately if they are all modified. The filename and content of the Excel file must follow the format described in [Translation Export](#). The language code in the filename must correspond to one of the available languages in top-level metadata's [Language tab](#).



If there are no errors with the Translation Import, a success prompt appears. Click the **OK** button to continue.

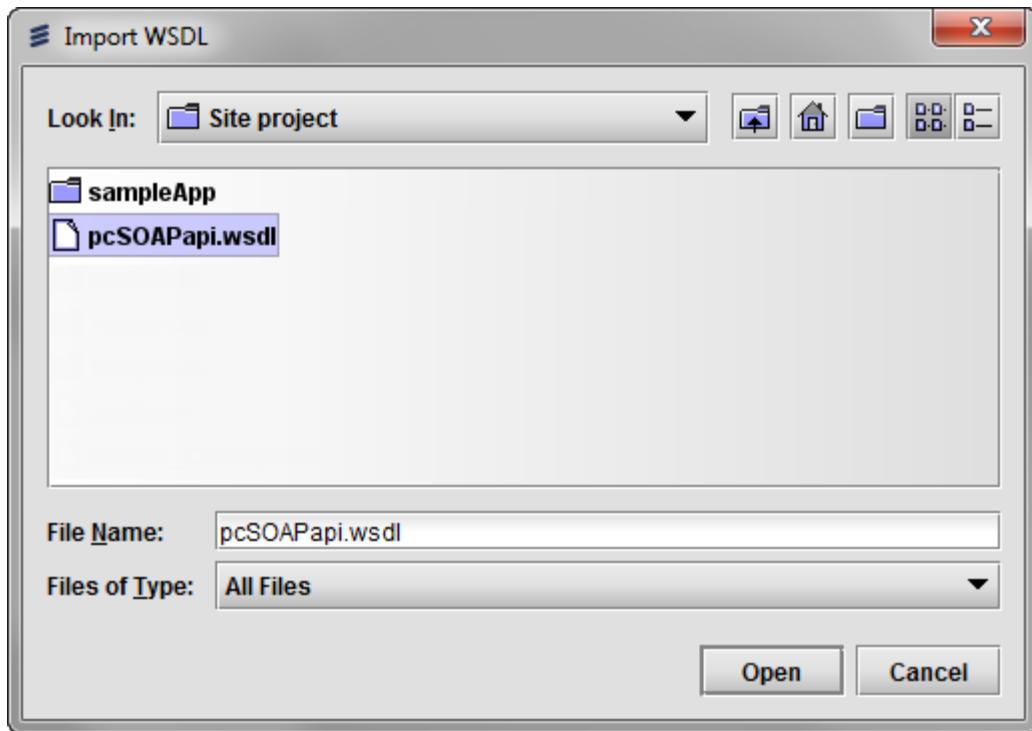
The imported Translations are stored in **languages** folder of Project Directory as XML files.

## Import WSDL

You can import a *Web Service Description Language* (WSDL) file as a resource of the Project using the **File > Import > WSDL** menu command.

Upon invoking the menu command, the **Import WSDL** dialog is displayed. Select a WSDL file to import.

**Note:** Axis2, SOAP encoding, and complex data types are unsupported.



Once imported, the WSDL file is stored in **schema** subfolder of **resources** folder in project directory. In Velocity Studio, the **Resource** tab of **Navigation Pane** is displayed upon successful import.

Note that importing the WSDL file as resource does not immediately generate application metadata based on the WSDL. You can generate interface-related metadata for your application based on the WSDL file by using **Convert to interface** right-click pop-up menu command on the resource file. See [Navigation Pane - Resource](#) for details.

At runtime, XSDs and WSDLs are found if they are in the **Resources\Schema** folder. They can also be in the **Resources\Schema** folder of a template JAR. If they are not found, they are dynamically generated and stored in the system/user temp folder. The available WSDLs for running SOAP listeners are exposed through the services URL. To be found, the WSDL name must match the SOAP service name (that is, metadata interface name) with a .wsdl extension.

**Example:** The **Resources\Schema** folder contains the following:

- MySoapInterfaceName.wsdl
- subfolder/Schema1.xsd

You can access the WSDL externally by using the following URL:

<http://MyHost:8080/cwf/services/MySoapInterfaceName?wsdl>

You can access the corresponding imported XSD as follows:

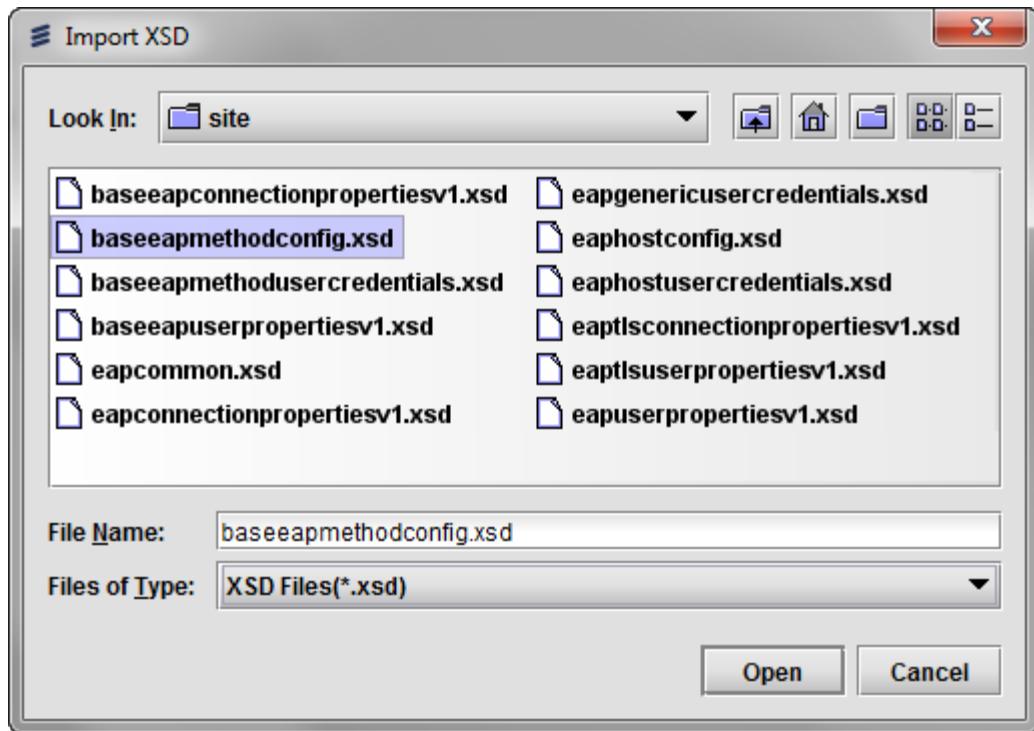
<http://MyHost:8080/cwf/services/MySoapInterfaceName?xsd=subfolder/Schema1.xsd>

## Import XSD

You can import a *XML Schema Definition* (XSD) file as a resource of the Project using the **File > Import > XSD** menu command.

Upon invoking the menu command, the **Import XSD** dialog displays. Select a XSD file to import.

**Note:** Axis2, SOAP encoding, and complex data types are unsupported.



Once imported, the XSD file is stored in the **schema** subfolder of **resources** folder in project directory. In Velocity Studio, the **Resource** tab of **Navigation Pane** is displayed upon successful import.

If the XSD schema in the selected file refers to other files using import statements, the other files will also be imported.

### Notes:

- Importing the XSD file as a resource does not immediately generate application metadata based on the XSD. You can generate Data Types and Data Structures for your application metadata based on the XSD file, by using the **Convert to datastructures** right-click pop-up menu command on the resource file. See [Navigation Pane - Resource](#) for details.
- When importing anonymous data types, they are placed in the parent namespace.
- Name conflicts due to case-insensitive matches in top-level objects are automatically resolved so that all schema objects are imported.

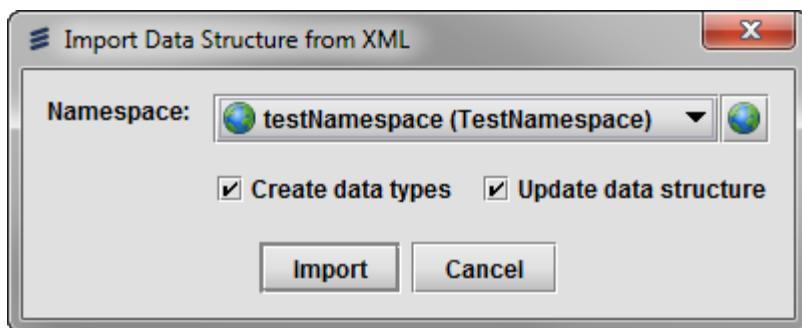
## Import XML Data

A Data Structure can be created based on the structure of a sample XML data file by selecting the **File > Import > XML Data** menu command.

The **XML Data** command is a helper function that is used to reduce the development work. Instead of manually creating the corresponding data type objects and Data Structure objects, the metadata developer can automatically generate them. When this command is used, certain assumptions are made that may not be valid in all cases. It is recommended that the developer carefully examines the generated objects and does appropriate changes where needed.

Velocity Studio always attempts to use the existing metadata objects before generating new ones (unless new objects are explicitly requested). In some cases, Velocity Studio may decide to modify an existing metadata object to fit the required properties, but this will only be done if the modified object is backward compatible with the original object (that is, Velocity Studio will change existing data type objects only if their base type (integer, boolean, string, etc.) is the same and the length and precision of the modified data type object can accommodate the length and precision of the original type). For Data Structures, Velocity Studio will add new nodes if needed but never will remove existing nodes. If a metadata object with the same name already exists in the specified namespace and it cannot be reused or modified, Velocity Studio will generate a new object that has the name of the existing object with a numeric suffix added for uniqueness.

Once the menu command is invoked, the first **Import Data Structure from XML** dialog opens where the XML file to be imported is selected. After file selection, the second **Import Data Structure from XML** dialog opens.



Field	Description
Namespace	Namespace in which to create the new Data Structure object.
Create data types	If checked, data type objects for each new tag name will be created.
Update data structure	If checked, the existing the Data Structure object will be updated/merged with the incoming data. If unchecked, the existing Data Structure object will be replaced by a new Data Structure object. <b>Note:</b> This applies only if a Data Structure object by the same name exists. If an object with the same name does not exist then a new Data Structure object will be created whether checked or unchecked.
Import	When clicked, creates or updates the Data Structure object from the XML file.
Cancel	When clicked, closes the dialog without any action.

**Note:** When creating XML in Velocity Studio, the following properties do not appear:

- filenameHash
- versionHash

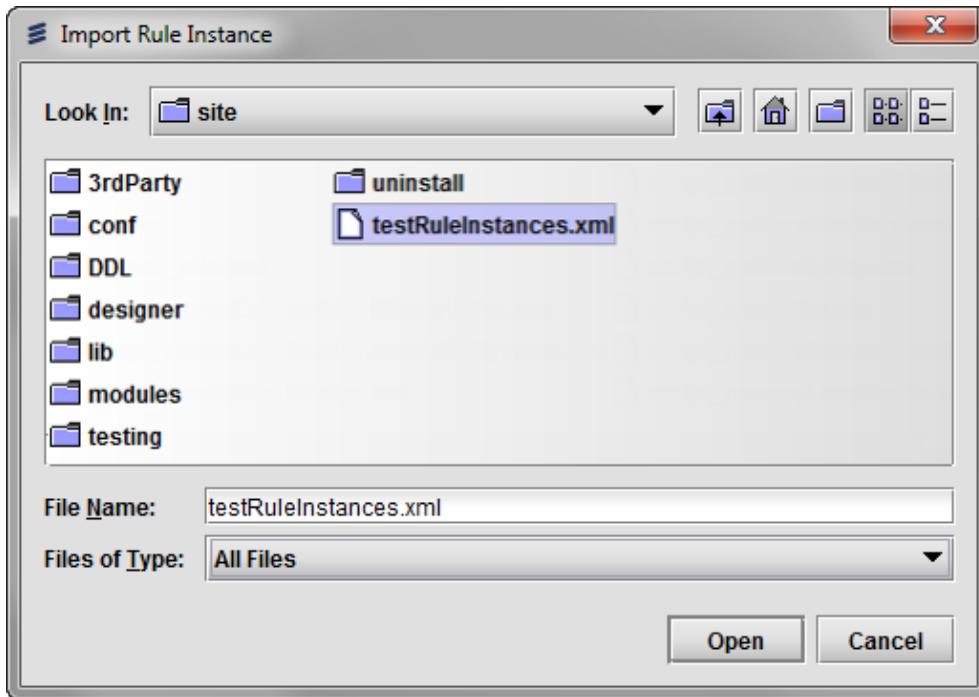
Additionally, validation does not occur for these two properties when you import XML files with these properties.



## Import Rule Instance

Velocity Studio allows you to import existing rules instances from an XML file into your CWDYNAMICMETADATA table by completing these steps:

1. From the Velocity Studio menu bar, click **File > Import > Rule Instance**.
2. From the Import Rule Instance dialog, locate the XML file containing the rules instances that you want to import. Click the **Open** button to continue.



3. If you are not already connected to a database, the Database Login dialog appears, prompting you to log in. Click the **OK** button to continue.
4. Your rule instances are imported, provided that they exist and do not have any validation errors. The number of successfully imported rule instances display in Velocity Studio.

## Import Rule Instances from the Command Line

You can import rule instances from the command line by using the ruleInstancesImport command, which can be found in the `<installation_folder>\designer\env\` folder:

```
ruleInstancesImport JDBC_URL metadata import_file
```

Where:

- *JDBC\_URL* is any JDBC URL or takes the form *user@host:port:sid*,*user@host:port/service\_name*
- *import\_file* is the name of your XML file

Examples:

```
ruleInstancesImport cw@localhost:1521:orcl c:\metadata c:\ruleInstances.xml  
ruleInstancesImport  
"cw@oracle.jdbc.driver.OracleDriver;jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on) (ADDRESS=(PROTOCOL=TCP) (HOST=c:\metadata c:\ruleInstances.xml
```

## Export Translation

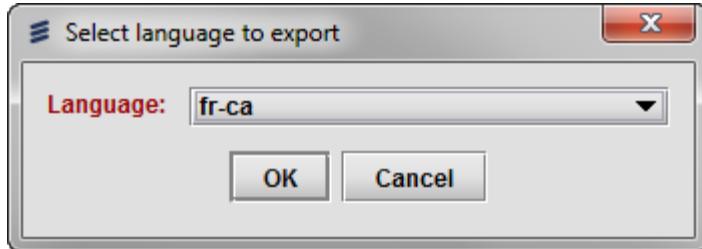
The purpose of [translation](#) is for defining all fields and labels visible in the runtime application in a specific language. You can export a set of Excel files that contain all translation records to a specified language in the project by clicking **File > Export > Translation** from the menu bar.

This function allows you to export translation that has been previously [imported](#). You can also export translation on a language that has not been previously imported, which serves as the Excel spreadsheet template for you to fill in the translation using the Excel application, before you import the translation into the project.

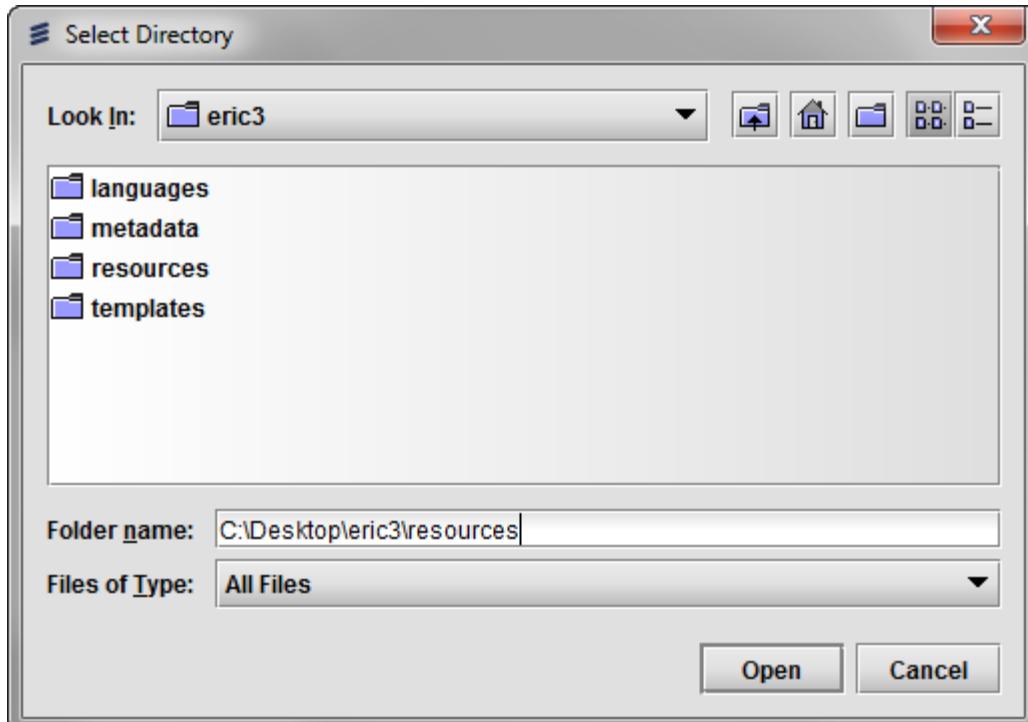
It is recommended that you save your metadata project before invoking this menu command so that latest changes in metadata objects are reflected in the Excel files.

To use the export translation function, perform these steps:

1. From the menu bar, click **File > Export > Translation**.
2. From the **Select language to export** dialog, click the **Language** drop-down menu and select a language from the available list. You are restricted to export from the list defined in top-level metadata's [Language tab](#).



3. By default, the exported resources are sorted by translation value. To have your exported resources sorted by translation key, select the **Sort By Key** checkbox. Click the **OK** button to continue.
4. The **Select Directory** dialog displays. Select a directory where you want the files to be exported. You may save the Excel files under **Resources** folder of the project directory such that these files can be managed as part of the project.



5. Click the **Open** button to export your metadata resource files. These exported files have the following hard-coded names:

- appRC\_//-xx.xls
- metaRC\_//-xx.xls
- sysRC\_//-xx.xls
- userRC\_//-xx.xls

where // is the language of the selected language code and xx is the country or locale (for example, *appRC\_en-us.xls*).

Each file contains an Excel sheet with three columns:

- Resource ID. It is a hidden column for system usage only. Do not change data in this column.
- Resource label on the metadata language. It is a source for translation. Do not change data in this column.
- Resource label on the currently selected language. This column can be empty. A translation of the label from column B should be written here.

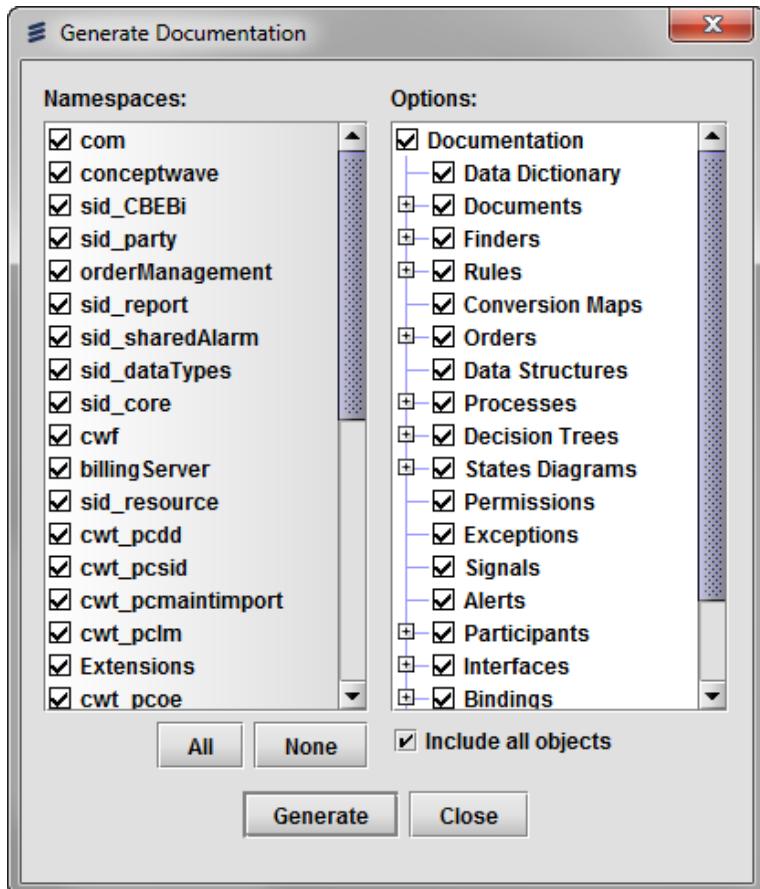
6. A confirmation message appears, indicating that your export was successful. Click the **OK** button.

**Notes:**

- Any row can be deleted from the generated Excel sheet. It only means that translation for this resource is missing when the file will be imported back into the **Resources** tab of your metadata project.
- Do not add new rows in the Excel sheet or, if you do, always leave a cell in the column A empty.
- Do not add new columns between columns A and C.
- The generated part of the resource file name must not be changed. Otherwise, you will not be able to import the file back into the **Resources** tab. Any additions to the file name can be done only between RC and underscore (\_) characters (for example, *metaRCforTranslation\_//-xx.xls*).
- You can export icon tooltips. The icon **Tooltip** property only appears when the UI element's **Icon** property is assigned an image.

## Generate Metadata Documentation

Full HTML design documentation for the current project can be generated by selecting the **File/Export/Documentation** menu command. Once the command is selected, the **Generate Documentation** dialog will open where you can specify the namespaces and metadata objects to be included in the generated documentation. By default, all namespaces and objects are included. The table below describes the fields in the dialog.



Field	Description
<b>Namespaces</b>	Allows the user to select the namespaces in the current metadata that will be documented.
<b>Options</b>	Allows the user to select the object types in the current metadata that will be documented. When the <b>Documentation</b> check box is selected all the object types in the <b>Options</b> list will be selected. When the <b>Documentation</b> check box is deselected all the object types in the <b>Options</b> list will be deselected.
<b>All</b>	When clicked, all the namespaces in the <b>Namespaces</b> list will be selected.
<b>None</b>	When clicked, all the namespaces in the <b>Namespaces</b> list will be deselected.
<b>Include all objects</b>	When clicked, all metadata objects of the types listed in the <b>Options</b> pane will be included in the documentation
<b>Generate</b>	When clicked, opens the <b>Save Documentation</b> dialog where the directory location to export the file too and the file name is specified.
<b>Close</b>	When clicked, closes the dialog without any action.

The documentation that is generated consists of tables names, columns, etc. The example, shows documentation that was generated for a user interface application that contained one variable.

## TABLE OF CONTENT

1.  [OrderEntry](#)
  - 1.1.  [User Interface](#)
    -  [Order Entry](#)

### 1. OrderEntry

Description	
Target Namespace	

#### 1.1. User Interface

 <a href="#">Order Entry (OrderEntry)</a>
--

Description	
Overrides	
Extends	 <a href="#">Application ()</a>

#### Variables

Variable	Type	Audit	Array	Constant	Validation	Init	Calculation	Trigger
OrderNumber	Integer(39)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				

#### Forms

##### Default

Element Name	Element Type	Parent	Binding	Optional	Editable	Visible
VerticalLayout	Vertical Layout					
OrderNumber	Textfield	VerticalLayout	OrderNumber			

If there are multiple namespaces, the namespaces will be sorted and presented as a tree, as shown in the following figure.

## TABLE OF CONTENT

- +1.  [Application UI](#)
- 2.  [Billing Stub](#)
  - +2.1.  [Data Structures](#)
  - +2.2.  [Interfaces](#)
  - +2.3.  [Bindings](#)
  - +2.4.  [External Services](#)
- +3.  [com](#)
- +4.  [conceptwave](#)
- +5.  [ConceptWave System](#)
- +6.  [Customer Management](#)
- +7.  [CWT Catalog Designer](#)
- +8.  [CWT Catalog Designer Data Dictionary](#)
- +9.  [CWT Product Catalog Maintenance API](#)
- +10.  [Order Analytics](#)
- +11.  [Order Management](#)
- +12.  [reports](#)
- +13.  [SID\\_CBEbi](#)
- +14.  [SID Common](#)
- +15.  [SID Party](#)
- +16.  [SID Product](#)

### 1. Application UI

Description	
Target Namespace	

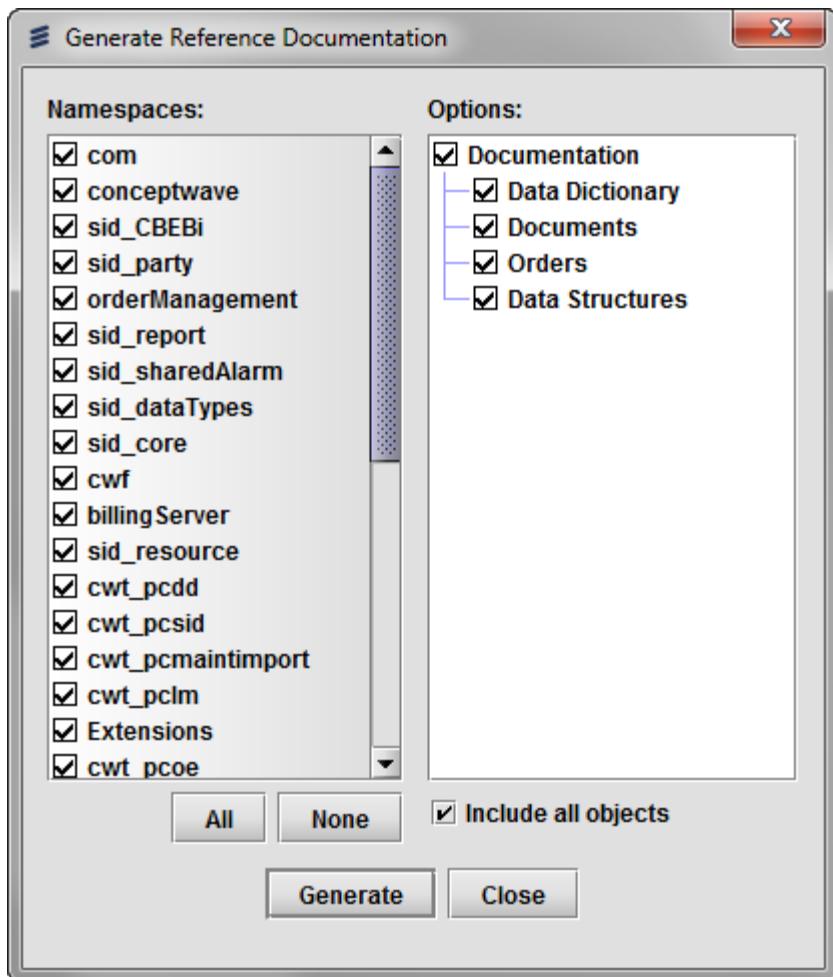


## Generate Metadata Reference Documentation

The [Documentation](#) command from the **File/Export** menu generates a large volume of documentation for all metadata objects of the specified types in the selected namespaces. In many cases, less documentation is adequate, and you would like to decide which objects to be included in documentation.

The **File/Export/Reference** command, comparatively, allows generating less documentation, which includes only the essential properties of the selected metadata objects. The HTML documentation that is generated is referred to as *reference documentation*.

Once the menu command is invoked, the **Generate Reference Documentation** dialog opens where you can specify the namespaces and metadata objects that will be included in the reference documentation. By default, all namespaces and objects are included. The table below describes the fields in the dialog.



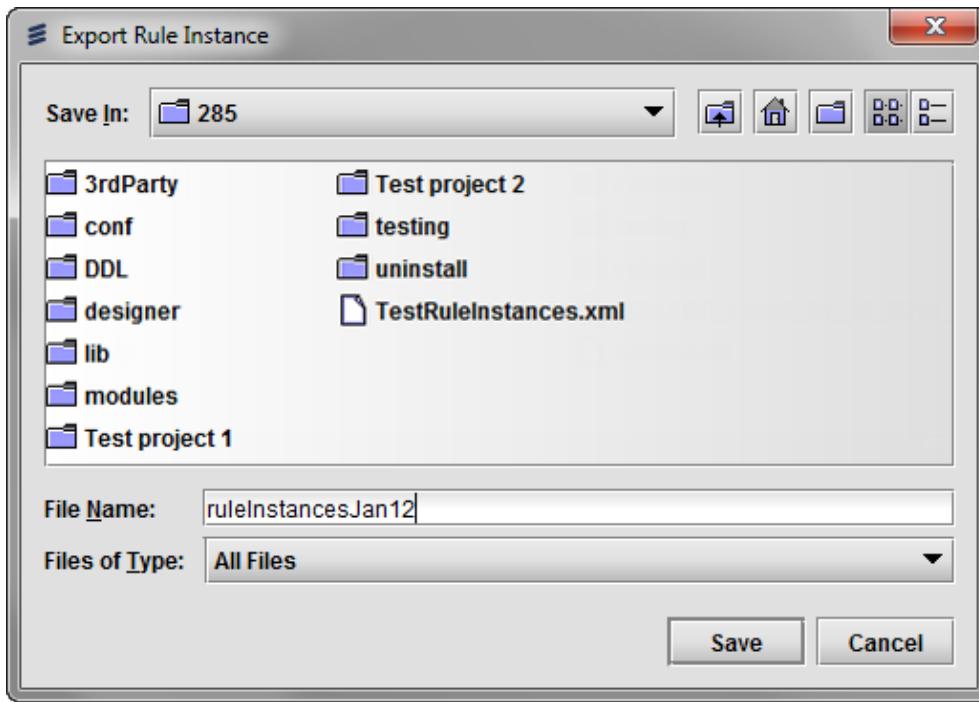
Field	Description
<b>Namespaces</b>	Allows the user to select the namespaces in the current metadata that will be included.
<b>Options</b>	Allows the user to select the object types in the current metadata that will be included. When the <b>Documentation</b> check box is selected all the object types in the <b>Options</b> list will be selected. When the <b>Documentation</b> check box is deselected all the object types in the <b>Options</b> list will be deselected.
<b>All</b>	When clicked, all the namespaces in the <b>Namespaces</b> list will be selected.
<b>None</b>	When clicked, all the namespaces in the <b>Namespaces</b> list will be deselected.
<b>Include all objects</b>	When clicked, all applicable metadata objects of the types listed in the <b>Options</b> pane will be included in the documentation
<b>Generate</b>	When clicked, opens the <b>Save Documentation</b> dialog where the directory location to export the file too and the file

	name is specified.
<b>Close</b>	When clicked, closes the dialog without any action.

## Export Rule Instance

Velocity Studio allows you to export existing rules instances from your CWDYNAMICMETADATA table to an XML file by completing these steps:

1. From the Velocity Studio menu bar, click **File > Export > Rule Instance**.
2. From the Export Rule Instance dialog, enter the **File Name** that you want your rules instances to be exported to. Click the **Open** button to continue.



3. If you are not already connected to a database, the Database Login dialog appears, prompting you to log in. Click the **OK** button to continue.
4. Your rule instances are exported to your specified XML file, provided that they exist and do not have any validation errors.

## Export Rule Instances from the Command Line

You can export rule instances from the command line by using the ruleInstancesExport command, which can be found in the `<installation_folder>\designer\env\` folder:

```
ruleInstancesExport JDBC_URL export_file
```

Where:

- `JDBC_URL` is any JDBC URL or takes the form `user@host:port:sid`, `user@host:port/service_name`
- `export_file` is the name of your XML file

Examples:

```
ruleInstancesExport cw@localhost:1521:orcl c:\ruleInstances.xml
```

```
ruleInstancesExport  
"cw@oracle.jdbc.driver.OracleDriver;jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on)(ADDRESS=(PROTOCOL=TCP)(HOST=c:\ruleInstances.xml
```

## **Customize Skin**

The **Customize Skin** menu item allows you to copy an existing skin and rename it to a new name in the application metadata. For more information, see [Customize Skin](#) in *Modifying URL Mappings, Stylesheets, and Skins*.

## Preferences

The Designer Preferences page allows you to define a number of Velocity Studio settings. Click any of the following sections in the left menu to change your Velocity Studio settings:

- [Runtime](#)
- [Appearance > View](#)
- [Project](#)
- [Export](#)
- [Setting](#)
- [Test Case](#)

### Runtime

The **Runtime** preferences specify how Velocity Studio shall execute at runtime:

Field	Description
HTTP Port	The Port number to use to serve Web application when the project is <a href="#">run within Velocity Studio</a> . Default is 8080.
Help Port	The Port number to use to serve Help pages in Velocity Studio. Default is 8081.
Preview Port	The Port number to use to display the preview pane when <a href="#">managing Forms</a> in Velocity Studio. Default is 8082.

For any Port settings, if the port number is already in use by the machine, the runtime attempts to bind to the next available incremental port number. In this case, this information appears in the [console](#).

### Appearance > View

This section allows you to defines the view preferences within Velocity Studio when navigating metadata:

Field	Description
Tree node labels	Specifies how the text of the nodes in the metadata tree of <a href="#">Navigation Pane</a> is displayed. It can be one of the following: <ul style="list-style-type: none"><li>• <b>Name Only</b> - only name of metadata object is displayed</li><li>• <b>Label Only</b> - only label of metadata object is displayed</li><li>• <b>Name (Label)</b> - name of metadata object is displayed, appended with its label in brackets if label is different from name. This is the default.</li><li>• <b>Label (Name)</b> - label of metadata object is displayed, appended with its name in brackets if name is different from label.</li></ul>
Element node labels	Allows you to specify how the text of element nodes is displayed in the element tree. It can be one of the following options: <ul style="list-style-type: none"><li>• <b>Name Only</b> - only the element's name is displayed</li><li>• <b>Label Only</b> - only the element's label is displayed</li></ul>
Form Preview - Enable Preview	By default, the preview area is enabled. Deselecting this setting disables all settings in the Form Preview section and <a href="#">disables the preview area</a> when you are developing your metadata.
Form Preview - Auto Refresh Browser	Specifies whether to automatically refresh the preview pane when you click a different form element in the <a href="#">Form</a> design in Velocity Studio.
Form Preview - Minimize Preview	Allows you to <a href="#">minimize the preview area</a> when you are developing your metadata in the design area, providing you with a full design area.
Form Preview - Auto Expand Element Tree	Allows you to view multiple levels of children (for example, submenus) on a form. By default, this setting is checked.

### Project

This section allows you to define your project preferences:

Field	Description
<b>Filtered resources</b>	When Velocity Studio manages the Project Directory (that is, source files of the project), any folders and files with the specified name in this field are ignored and left alone in the directory. This setting defaults to .svn to allow version control on Project Directory using <a href="#">subversion</a> .
<b>Filtered in war</b>	This field allows you to specify file names that are separated with a comma (,). When you create a WAR file using metadata, the files specified in this field are ignored. This setting defaults to .bak to filter out all the backup files. The <a href="#">Build Deployment WAR</a> function uses this field.
<b>Webapp library folder</b>	Specify the folder that contains extra JAR files, which will be packaged to <i>WEB-INF/lib</i> folder upon <a href="#">packaging</a> .
<b>Compress SmartClient Resources</b>	By default, this setting is checked. When this setting is unchecked, upon packaging, the generated WAR file contains <i>servlet.compress: false</i> .  <b>Note:</b> See the <a href="#">Disabling SmartClient compression</a> section on how this parameter in your startup script works with the <b>Compress SmartClient Resources</b> checkbox.
<b>SmartClient Resources Expiry Time (hours)</b>	By default, this setting defaults to one hour. This setting allows you to set the expiration time for SmartClient resources.
<b>Folders to look in for template files</b>	This property allows you to specify either a relative or absolute directory where Velocity Studio will search for template JAR files. This search is performed in addition to searching the templates folder of your metadata. To specify more than one relative or absolute directory, separate them with a semicolon (;).  <b>Notes:</b> <ul style="list-style-type: none"> <li>If a folder has the same file as another folder, the template JAR file is read from the templates folder first, and then from left to right of the specified folders.</li> <li>When you add a new template library file to the metadata header, a prompt appears, asking whether you want to copy the file into the templates folder of your metadata. Clicking the <b>Yes</b> button copies the files to the template folder. Otherwise, clicking the <b>No</b> button adds the path to the project settings property, if it does not already exist.</li> </ul>
<b>Runtime Test Features</b>	This field's drop-down menu shows features that are currently defined and activated for testing using the test handler to open any protected template. This field allows you to turn these features on and off.

## Export

This section allows you to define preferences in export commands:

Field	Description
<b>Export Schema inside WSDL</b>	If checked, any Data Types or Data Structures that are referenced by the SOAP elements are included in <i>&lt;schema&gt;</i> element at the top of the WSDL file when Exporting WSDL, akin to the content of an XSD file.
<b>Export Referenced Types Only</b>	For WSDLs, if you select this setting, only types referenced by the services and any types that they depend on are exported.  For XSDs, the following occurs if you select this setting: <ul style="list-style-type: none"> <li>For the selected namespace, all types are exported.</li> <li>For referenced namespaces, only types required by the selected namespace are exported instead of the complete set of types for all referenced namespaces.</li> </ul>
<b>Resolve Override Types</b>	This setting allows you to indicate whether overrides are used in the WSDL (that is, references to a base object are replaced with references to an overriding object). If you want to use overrides, ensure that you make the WSDL or XSD consistent with your business needs, as overriding is not a supported feature in XSD. This setting is off by default, meaning that overrides are not resolved.

<b>Export Template JAR Folder</b>	<p>This field allows you to set the default path for building the template JAR file by clicking the <b>Browse</b> button and selecting the folder that you want. If the path is either empty or does not exist, a popup appears, prompting you to select a file editor. Otherwise, the default path is used to build the template JAR file.</p> <p>This setting is a project preference, and not a Velocity Studio preference, as indicated in the following example:</p> <ol style="list-style-type: none"> <li>1. Open a project (project A) and define the <b>Export Template JAR Folder</b> as Y.</li> <li>2. Proceed to export the file by clicking <b>Runtime &gt; Build Library JAR</b> from the menu bar. Ensure that the JAR has been exported to folder Y.</li> <li>3. Open another project (project B) and define the <b>Export Template JAR Folder</b> as folder X.</li> <li>4. Export the file by clicking <b>Runtime &gt; Build Library JAR</b> from the menu bar. Ensure that the JAR has been exported to folder X.</li> <li>5. Reopen project A. When you click <b>Runtime &gt; Build Library JAR</b> from the menu bar, the JAR is still exported to folder Y.</li> </ol>
-----------------------------------	---

## Setting

This section allows you to define the following setting preferences:

Field	Description
<b>Auto Save Metadata</b>	By default, this field is set to <b>Never</b> . You can specify how often Velocity Studio automatically saves your metadata by clicking the drop-down menu and selecting the number of minutes from the list.
<b>Prompt for Confirmation on Method Deletion</b>	By default, when you delete a method, you do not see a confirmation message. Selecting this field allows you to see this confirmation dialog before you delete any method.
<b>Show Finder Type Folder</b>	By default, this field shows the finder type folder in the Metadata tree.
<b>Show Library Warning</b>	By default, this field shows all library warnings received in Velocity Studio.
<b>Show Metadata Label in recent project list</b>	Select this checkbox to display the metadata label in the recent project list (that is, clicking <b>File</b> from the menu bar) and its file path.
<b>Generate upgrade SQL in one output file</b>	By default, this setting generates one upgrade.sql file when you select <b>Database &gt; Upgrade System</b> from the menu bar. When you uncheck this setting, Velocity Studio generates separate SQL script files for each logical schema.
<b>Generate verbose message in console</b>	When selected, this setting generates verbose messages in console of Velocity Studio. Such as, information about the calls on the system, like notices, warnings, and errors etc.
<b>Prompt to propagate subflow revision</b>	<p>By default, this field is selected, meaning that each time a subflow generates a new revision, a dialog with a list of processing using this subflow appears. You can then select processes that need to have their subflows re-adjusted to use the latest subflow revision. When you click the <b>OK</b> button, the selected process references are automatically re-adjusted.</p> <p>When this field is unchecked, generating a new subflow revision no longer prompts to re-adjust subflow references.</p>

## Test Case

The **Test Case** preference provides the ability to configure predefined scripts for test cases:

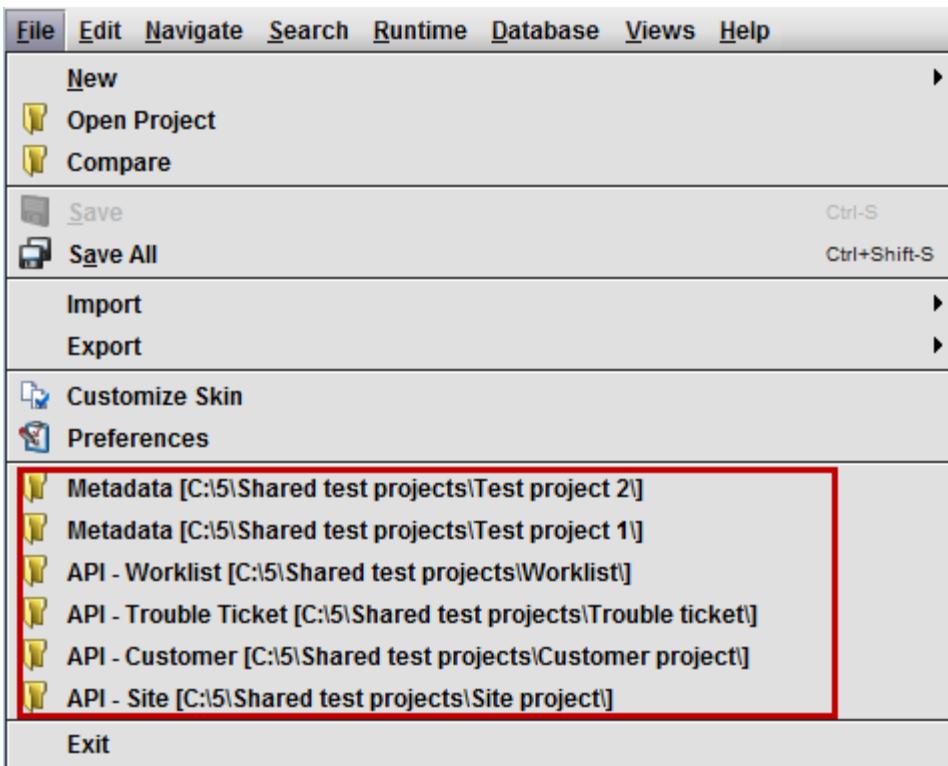
Field	Description
<b>Show TestCase Template</b>	By default, this field is unchecked and all three input fields are display only. Check this field to enable the three input fields and to indicate that the predefined script will be used when creating a new test case or

	test function.
<b>Setup Script, Teardown Script, Script Template</b>	By default, these three input fields are display only unless the <b>Show TestCase Template</b> checkbox field is checked. These fields are text areas that can be used to define a script for a test case and a test function.
<b>Default Script</b>	If modifications have been made to the predefined script via the three input fields, this button provides a means to view, in a dialog, the default predefined script (that is, before changes).

## List of Recently Opened Projects

---

Clicking **File** from the menu bar shows a list of recent opened metadata projects.



The list of projects in the aforementioned image shows the metadata label in the list of projects that were recently opened (that is, the **Show Metadata Label in recent project list** option has been selected in [Preferences > Setting](#)) and each project's path.

A file selection history list is also available whenever files or folders are selected and opened in the Velocity Studio. The designer.properties file in your `<installation_folder>\designer\env` folder contains your file selection history.

## Edit Menu

---

The **Edit** menu contains menu options as described in the following table:

Menu	Shortcut	Function
 <b>Cut</b>	<i>Ctrl-X</i>	Cut the selected metadata object.
 <b>Copy</b>	<i>Ctrl-C</i>	Copy the selected metadata object.
 <b>Paste</b>	<i>Ctrl-V</i>	Paste the cut or copied metadata object.
 <b>Delete</b>	<i>Delete</i>	Delete the selected metadata object.
<a href="#"><b>Independent Script Editor</b></a>		Open and save independent JavaScript in a file.

## Independent Script Editor

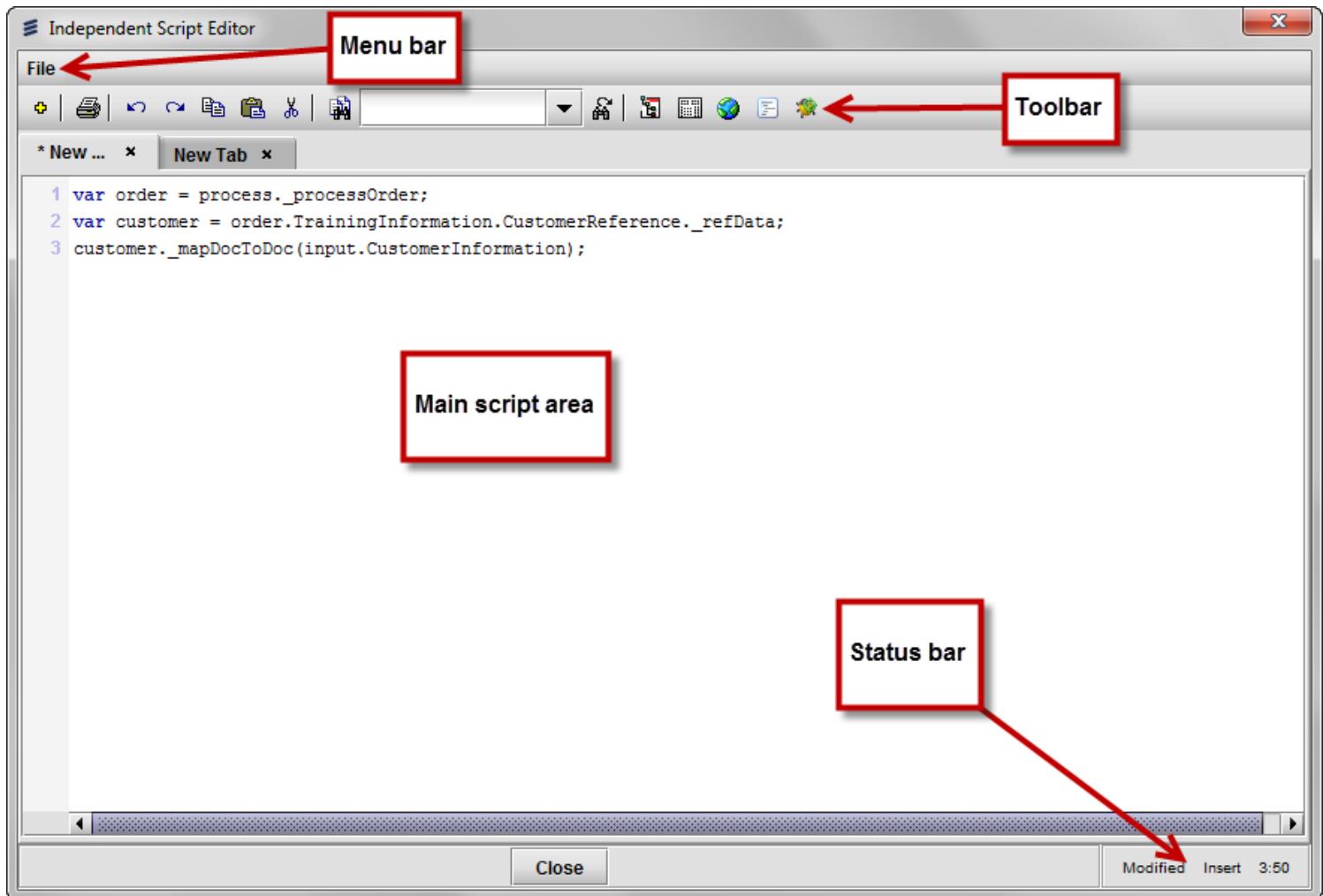
Velocity Studio provides an Independent Script Editor, which extends the [JavaScript Editor](#) and implements multiple tab functionality. You can open and save independent JavaScript in a file.

To open the Independent Script Editor, in Velocity Studio, click **Edit > Independent Script Editor** from the menu bar.

### Independent Script Editor - User Interface

The Independent Script Editor consists of four main areas:

- [Menu bar](#)
- [Toolbar](#)
- [Main area for script development](#)
- [Status bar](#)



#### Menu Bar

The Independent Script Editor contains a menu bar, which differs from the [JavaScript Editor](#). The following options are available when you click **File** from the menu bar:

Menu Option	Description
New Tab	Open multiple tabs within the script editor.
Open File	Open a dialog to specify the JavaScript file that you want to open.
Save	Save your JavaScript file.

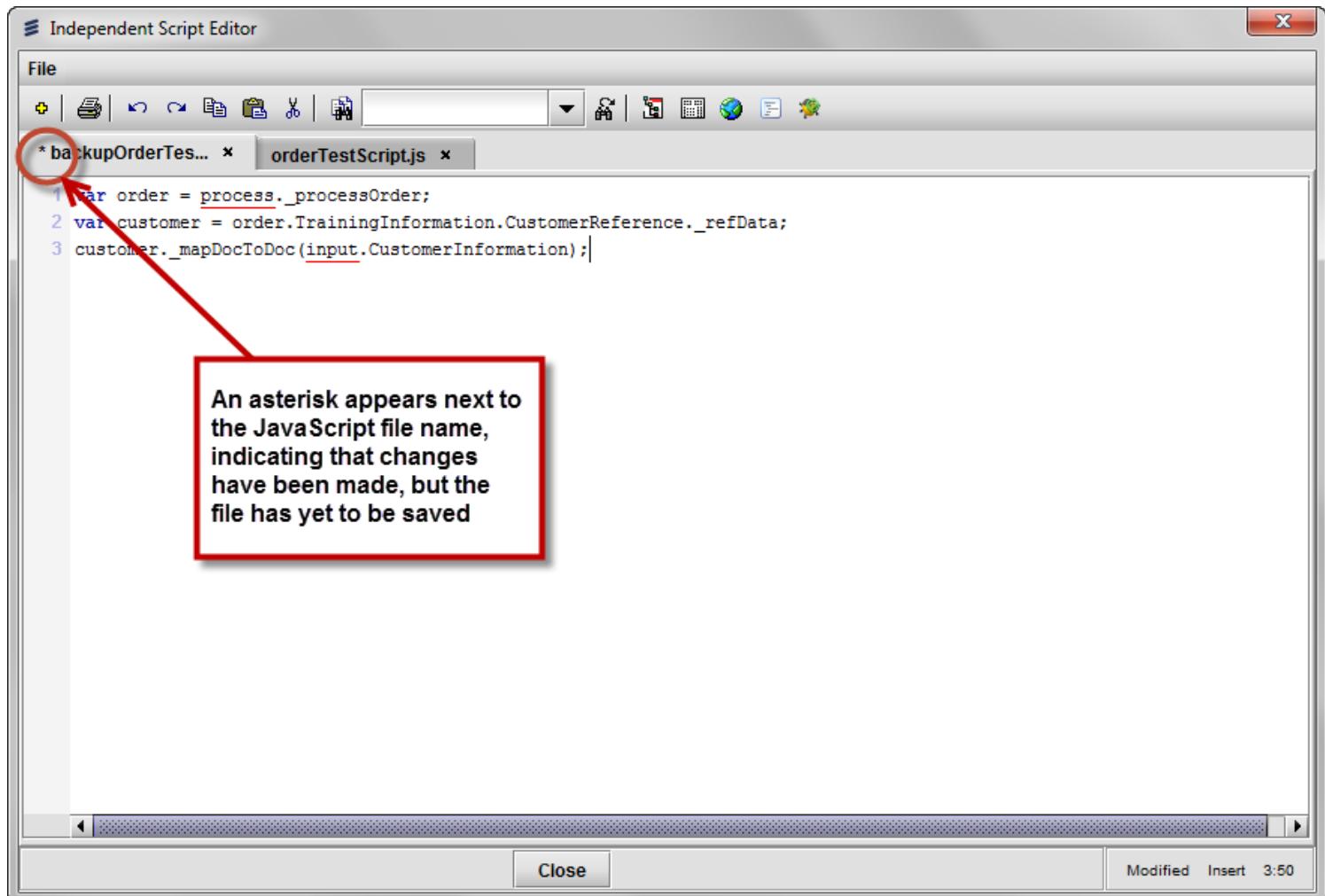
## Save As

Specify the location where you want to save your JavaScript file and provide a name for your file.

## Tabs

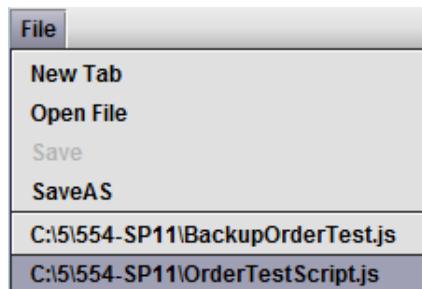
You can open multiple tabs within the Independent Script Editor. Each tab contains the name of your JavaScript file, if you have already named the file or you have opened an existing JavaScript file.

If you have made changes to your script and have not saved your file, an asterisk (\*) appears in the tab area.



## History

The Independent Script Editor contains a history feature. The editor keeps track of the last four JavaScript files that you have opened. The following example shows that two files were open from the editor.



## Toolbar

The Independent Script Editor toolbar allows you to perform a number of actions to help you with writing your script. The following table describes what each button does within the script editor.

Button	Shortcut	Function
--------	----------	----------

		Open multiple tabs within the script editor.
	<i>Ctrl + P</i>	<a href="#">Print the code</a> displayed in the Independent Script Editor.
	<i>Ctrl + Z</i>	<a href="#">Undo the last action</a> that you performed.
	<i>Ctrl + Y</i>	<a href="#">Redo an action</a> that you performed.
	<i>Ctrl + C</i>	<a href="#">Copy</a> highlighted text into your script.
	<i>Ctrl + V</i>	<a href="#">Paste</a> cut or copied text into your script.
	<i>Ctrl + X</i>	<a href="#">Cut</a> highlighted text into your script.
	<i>Ctrl + F</i>	<a href="#">Find specific words or text</a> within your script. You also have the option of finding words or text, and replacing them.
	<i>Ctrl + K</i>	The <a href="#">drop-down box</a> allows you to select from previous search criteria that you have used. Click the <b>Find Next Occurrence</b> button to the right of the drop-down field to find the next occurrence of your search criteria in your script.
	<i>Ctrl + Spacebar</i>	<a href="#">List methods</a> that you can use in your script by clicking this button. Select the method you want to insert into your script by double-clicking your selection.
	<i>Ctrl + M</i>	Open the <b>Select Method</b> dialog, which <a href="#">lists the available JavaScript functions</a> that can be selected and added into the script editor.
	<i>Ctrl + E</i>	Opens the <b>Select Element</b> dialog, which <a href="#">lists the available metadata elements</a> that can be selected and added into the script editor.
		Click this button to launch the JavaScript Code Template dialog, allowing you to <a href="#">insert code templates</a> or standard code patterns into the script editor.

## Main Area for Script Development

The Independent Script Editor's main area allows you to code and edit your scripts. You use this area with its toolbar, which provides you with a number of useful actions to help you develop and edit your scripts.

## Status Bar

The Independent Script Editor's status bar shows a number of useful details:

- Whether the script has been changed.
- Whether you are in **Insert** mode (insert text as you use the Independent Script Editor) or in **Overwrite** (enter text over existing text) mode.
- What line and character position your cursor is on. In the example that follows, your cursor is on line three, at character 50.



To the left of the status bar, the **Close** button closes the Independent Script Editor.

## Runtime Script Editor

You can use a system template UI that allows you to import, export, and edit JavaScript at runtime. To access this system template, do the following:

1. Create an application or application menu to display com.conceptwave.system/ScriptEditorUI.
2. At runtime, display this user interface.

## Edit JavaScript

To edit JavaScript from this user interface, complete these steps:

1. Click the **Edit** button (top right) to launch the editor.
2. Make your changes in the text area and then click the **OK** button.

### **Import a JavaScript File**

To import a JavaScript file from this user interface, complete these steps:

1. Click the **Import** button (icon with the down arrow) to launch the upload file dialog.
2. Select the .js file that you want and then click the **Upload** button.
3. The contents of your .js file appear in the main screen.

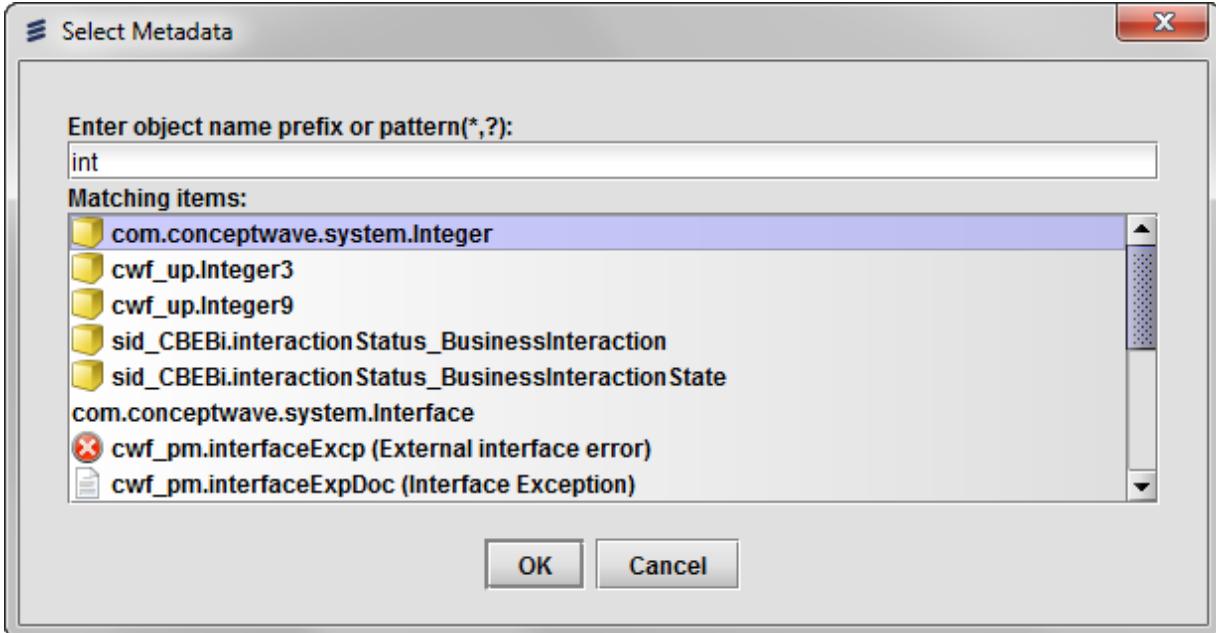
### **Export a JavaScript File**

To export a JavaScript file from this user interface, complete these steps:

1. Click the **Export** button (icon with the up arrow) to launch the Save As dialog.
2. Specify the name of the .js file that you want to export your script to and then click the **Save** button.

## Navigate Menu

The **Navigate** menu contains menu options as described in the following table:

Menu	Shortcut	Function
<b>Open Type</b>	<i>Ctrl + Shift + T</i>	This menu command provides a keyboard shortcut to navigating top-level metadata objects, an alternative to using the Navigation Pane. The <b>Select Metadata</b> dialog box appears.   <p>The dialog box contains all metadata objects of the project, including both the application and library metadata. See <a href="#">instructions</a> on how to use this dialog to search for the desired metadata object.</p>
<b>Forward</b>	 <i>Alt + Right Arrow</i>	Navigates to the next object in metadata object selection history.
<b>Back</b>	 <i>Alt + Left Arrow</i>	Navigates to the previous object in metadata object selection history.

## Velocity Studio - Search Menu

---

The Velocity Studio **Search** menu contains the following menu items and brief descriptions of each menu item:

### [Search](#)

Search in the project with name, label or script of metadata object.

### [API Search](#)

Search for an API in your metadata project scripts. Your search results appear in a log file.

### [API Refactor](#)

The API Refactor function allows you to do the following:

- Search for an API
- Refactor an API by specifying the original API with its replacement API

### [API Replace](#)

Replace an API call with another one in your metadata project scripts, but not in the modules (templates).

### [Replace References](#)

Replace all object (element) references in your metadata.

### [Replace Deprecated](#)

Replace all usage of deprecated system methods in all scripts of the project.

# Search

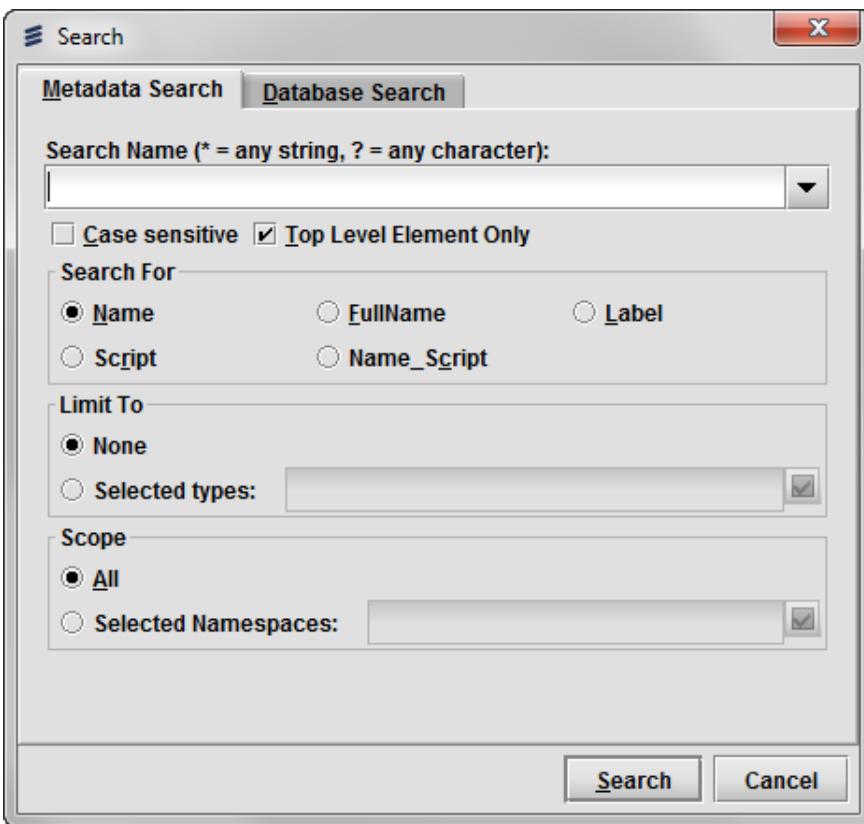
Search functionality is provided in Velocity Studio at **Search > Search...** menu (or use shortcut key *Ctrl-H*). Two search functions are provided, in separate tabs of the **Search** dialog that is opened:

- [Metadata Search](#)
- [Database Search](#)

You can toggle between the Metadata Search and Database Search tabs by pressing the *Alt + M* and *Alt + D* keys, respectively.

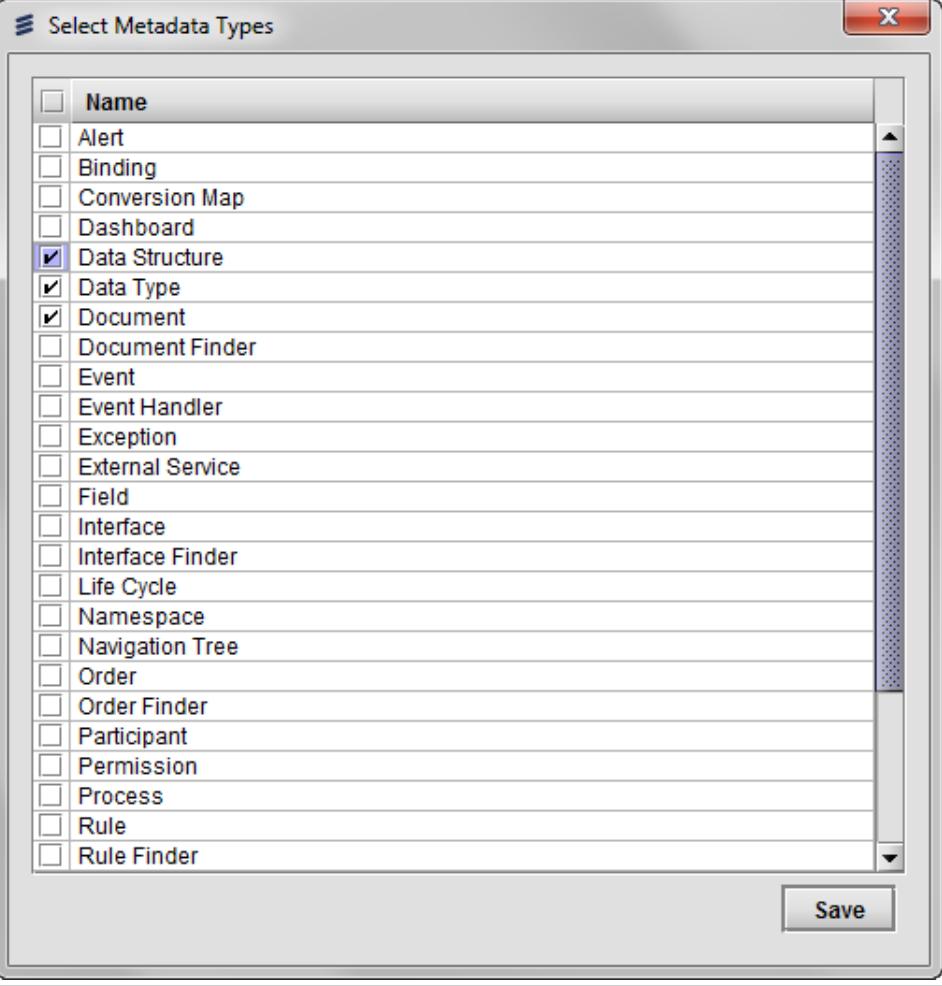
## Metadata Search

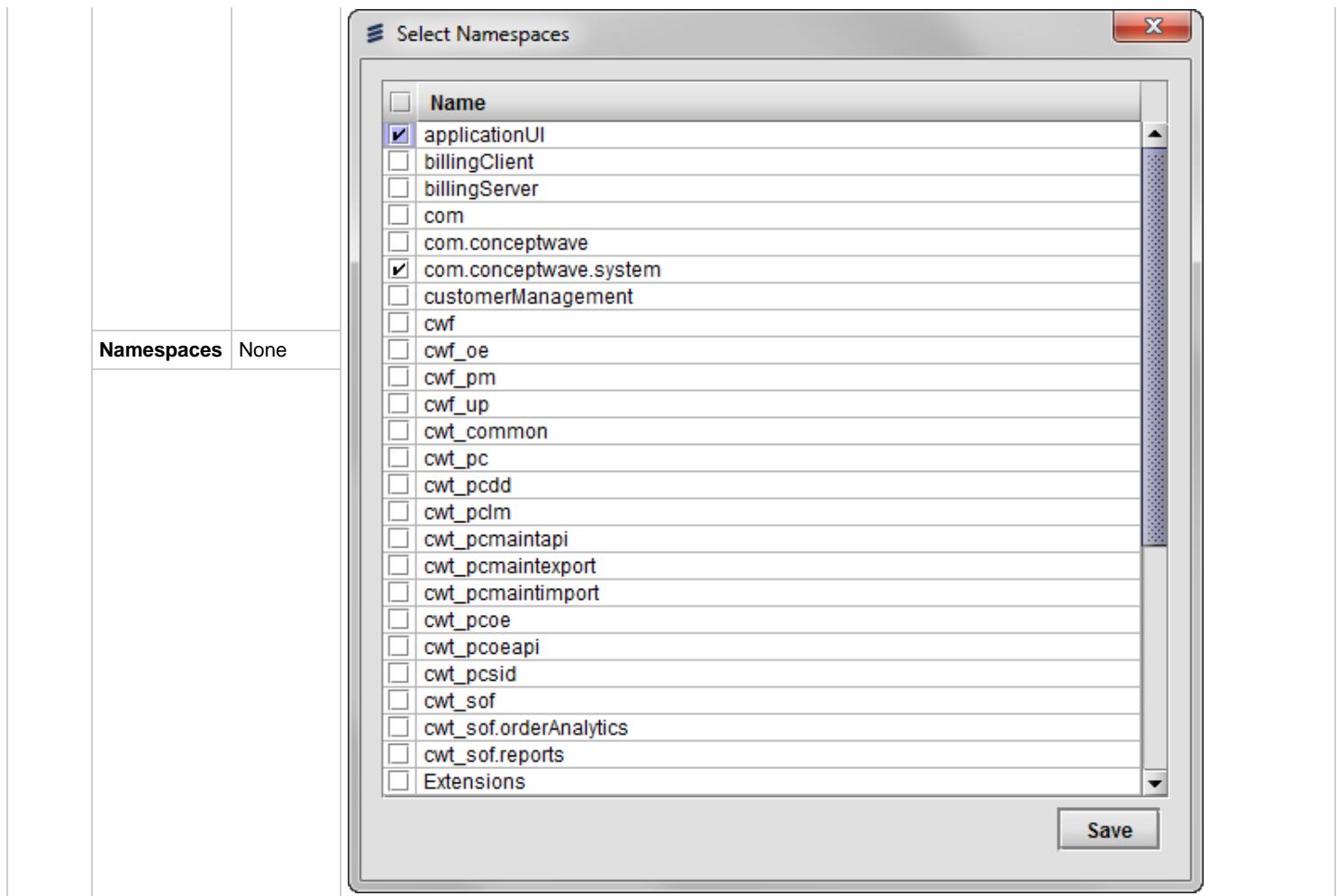
With **Metadata Search** tab of the **Search** Dialog, you can search for a metadata object by metadata object's Name, Label or Script content, with pattern matching on the search key. Both application metadata (that is, top-level metadata, UI element-level metadata, scripts, user interface, and Form (overlay)) and libraries are searched.



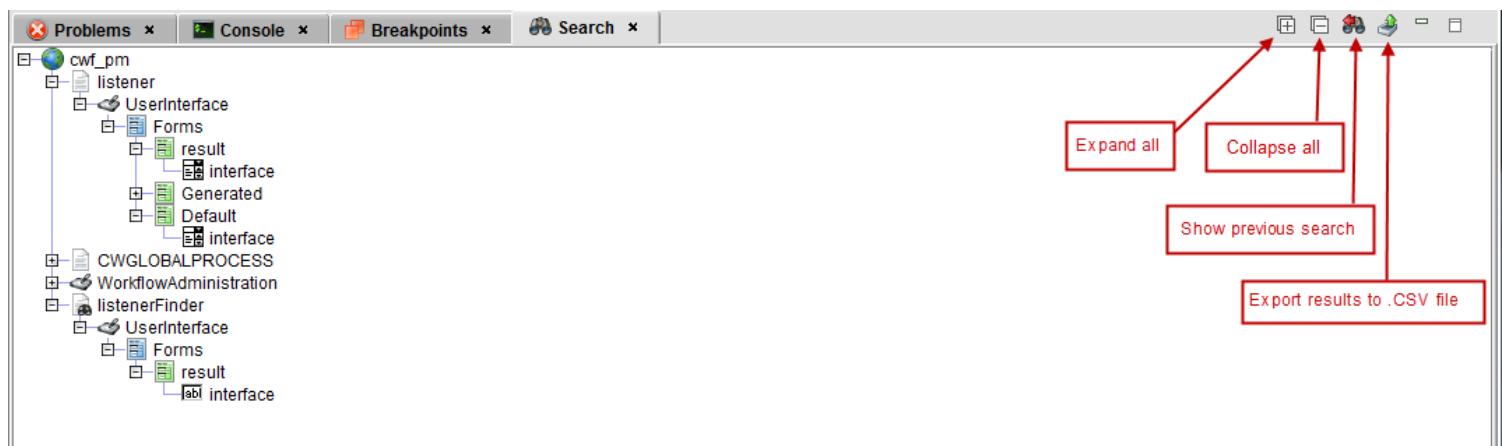
The following table explains the parameters of the search dialog:

Area	Parameters	Keyboard Shortcut	Description
Search Key	Search Name	None	<p>Enter the search key to search against. Wildcard characters *, ? and \ are supported.</p> <ul style="list-style-type: none"><li>• An asterisk matches any sequence of characters, including none.</li><li>• A question mark matches any single character.</li><li>• A backslash indicates that the character that follows should be interpreted as a literal character, not a wild card character, allowing the wildcard characters to be matched.</li></ul> <p>Without wildcards, exact match is to be performed.</p>
	Case sensitive	<i>Alt + C</i>	If checked, the search key is to be matched with exact case (for example, lowercase, uppercase as specified). Default is not checked.
	Top Level Element Only	<i>Alt + T</i>	By default, this parameter is checked, which means that the search results only contain top-level elements. Otherwise, deselecting this checkbox indicates that the search results contain all element types in the metadata.

Search For	Name	Alt + N	Search the search key by object names. When searching by name, the names of objects inside a metadata object, such as <b>Variables</b> , <b>Methods</b> and <b>Forms</b> , are included in the search as well.
	Label	Alt + F	Search the search key by object labels. When searching by label, the label of objects inside a metadata object, such as <b>Variables</b> and <b>Forms</b> , are included in the search as well.
	Full Name	Alt + L	Search the search key by the object's full name.
	Script	Alt + R	Finds a text fragment in the scripts. All script fields in all metadata are searched.
	Name_Script	Alt + C	Searches both object names and within scripts. The search results will contain results from both Name and Script criteria.
Limit To	None	None	Limits the search to select metadata object types. You can choose <b>None</b> to not limit by type, or <b>Selected Types</b> , which opens the <b>Select Metadata Types</b> dialog upon clicking the checkbox icon <input checked="" type="checkbox"/> to choose metadata object types to search.
Selected Types	None		 <p>The dialog box is titled "Select Metadata Types". It contains a list of metadata types with checkboxes. The selected types are: Data Structure, Data Type, Document, Document Finder, Event, Event Handler, Exception, External Service, Field, Interface, Interface Finder, Life Cycle, Namespace, Navigation Tree, Order, Order Finder, Participant, Permission, Process, Rule, and Rule Finder. A "Save" button is located at the bottom right of the dialog.</p>
Scope	All	Alt + A	Limits the search to select namespaces. Can choose <b>All</b> to not limit by namespace, or <b>Selected Namespaces</b> which opens the <b>Select Namespaces</b> dialog upon clicking the checkbox icon <input checked="" type="checkbox"/> to choose namespace(s) to search.



After specifying search parameters, click the **Search** button to search, which opens the [Search Pane](#), with all matched results listed in namespace hierarchy. Expand the namespaces using the expand icon, and double-click a metadata object beneath it to jump to that object in the navigation and detail panes.



If no results are found, a **No items found** pop-up box is displayed.

#### Show History of Previous Searches Performed

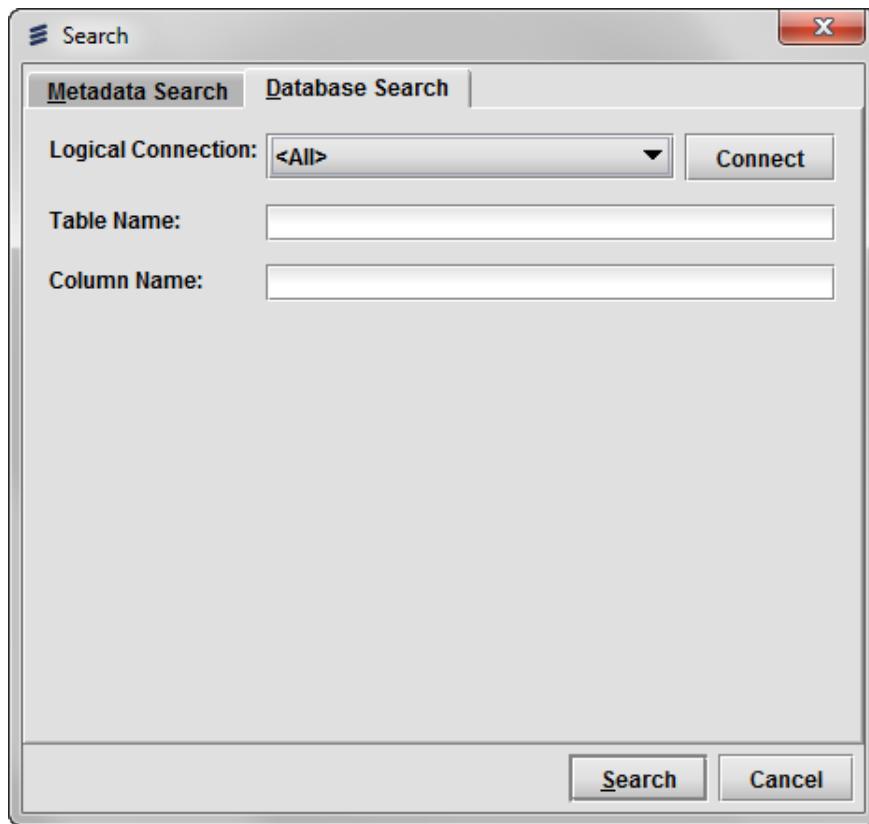
You can click the **Show Previous Searches** button to display the last five searches that you have previously performed, including the number of successful search references.



Selecting the **Clear History** option from the menu clears your search history.

## Database Search

With **Database Search** tab of the **Search** Dialog, you can search for a metadata object by Table Name and Column Name. Both application metadata and libraries are searched.



The table below explains the parameters of the search dialog. The **Table Name** and **Column Name** are optional parameters (for example, specify only column name parameter search all database tables with such column name).

Parameter	Description
<b>Logical Connection</b>	Choose the Logical Connection (as defined in <a href="#">Configuration application's Database page</a> ) to connect to the database. Or, choose <All> to search through all of them.
<b>Table Name</b>	The database tablename to search for. Must be exact match.
<b>Column Name</b>	The database table's column name to search for. Must be exact match.

Similar to Metadata Search, click the **Search** button to search after specifying search parameters, which opens the [Search Pane](#), with all matched results listed in namespace hierarchy. Expand the namespaces using the expand icon, and double-click a metdata object beneath it to jump to that object in Navigation Pane and Detail Pane.

If no results are found, a **No items found** pop-up box is displayed.

## API Refactor

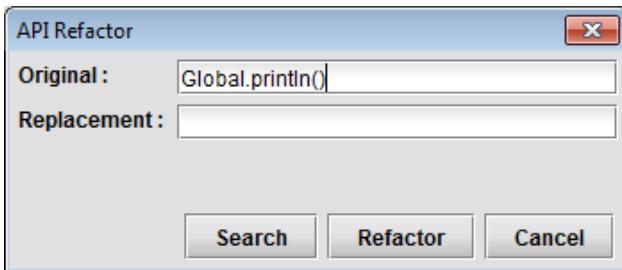
The API Refactor function allows you to do the following:

- Search for an API
- Refactor an API by specifying the original API with its replacement API

### Search function

The search function allows you to find occurrences of your API in your metadata:

1. Click **Search > API Refactor** to launch the API Refactor dialog
2. Enter the API that you are searching for in the **Original** field. The **Replacement** field is optional.



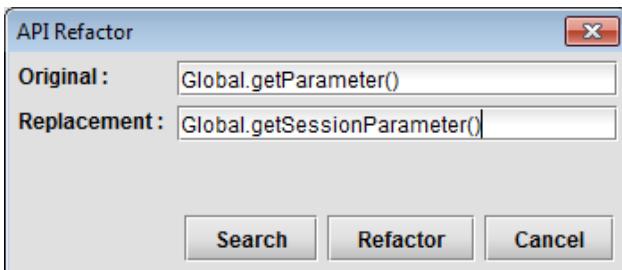
3. Click the **Search** button.
4. Your search results appear in the Search pane. In this example, occurrences of the Global.println() API appear in the TestScript script



### Refactor function

The refactor function allows you to find occurrences of your API in your metadata and replace them with another API:

1. Click **Search > API Refactor** to launch the API Refactor dialog
2. Enter the API that you are searching for in the **Original** field and its replacement API in the **Replacement** field.



3. Click the **Refactor** button.
4. Your search results appear in the Search pane. In this example, refactoring changes have been made to the TestScript script.



**Note:** The Refactor action class extends SDAPISearchAction.



## API Search

---

Clicking **Search > API Search** from Velocity Studio's menu bar allows you to search for an API in your metadata project scripts. Your search results appear in a log file.

### API Search Log File

The API search function automatically creates a log file with the name CWQCAPISearchyyyymmdd-hhmmss.html, which is located in the `<installation_root_directory>\designerenv\log` folder, where:

- *yy* denotes the year
- *mm* represents the month
- *dd* indicates the day
- *hh* denotes the hour
- *mm* represents minutes
- *ss* indicates seconds

The log contains an HTML table with three columns:

- The API used as the search criterion.
- The location in your metadata where the API appears. The location has the format: *namespace:element-name.method-name;line number*. If the element is a global script, the method name does not appear in the syntax.
- The API instance in your metadata.

The following is a sample log table row:

mapDocToDoc(?1)	mapDocToDoc (TestNamespace.TestScript; Line 1)	mapDocToDoc(blah);
-----------------	--	--------------------

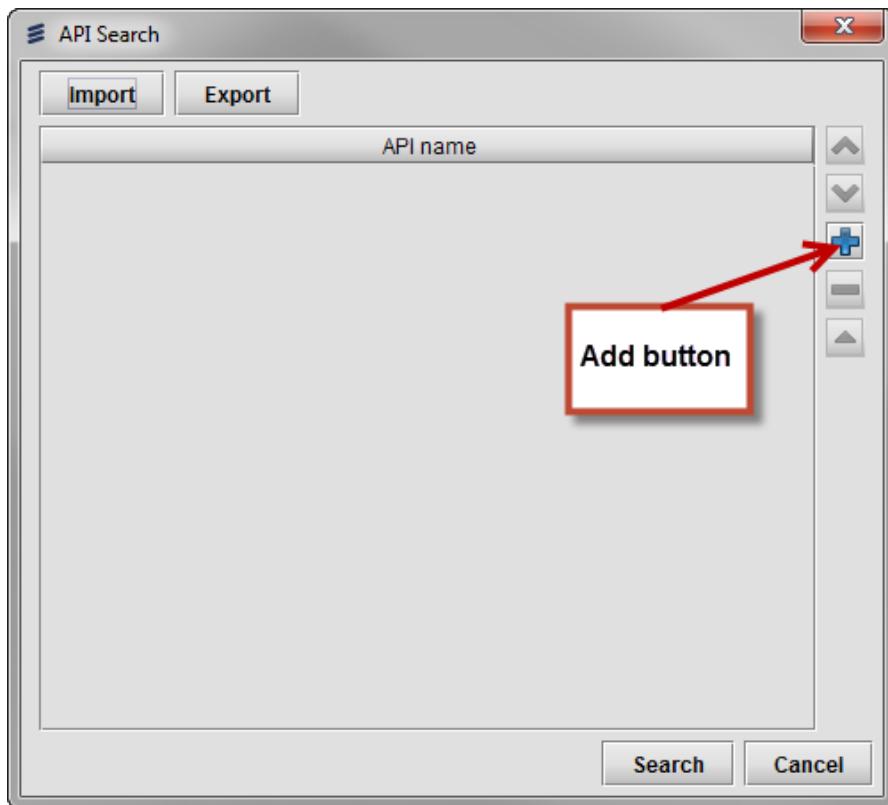
### API Search Syntax

The API search syntax is the same as the [API replace](#) rule syntax.

### Use API Search

To use the API search function, complete the following steps:

1. From the API Search screen, click the **Add** button.



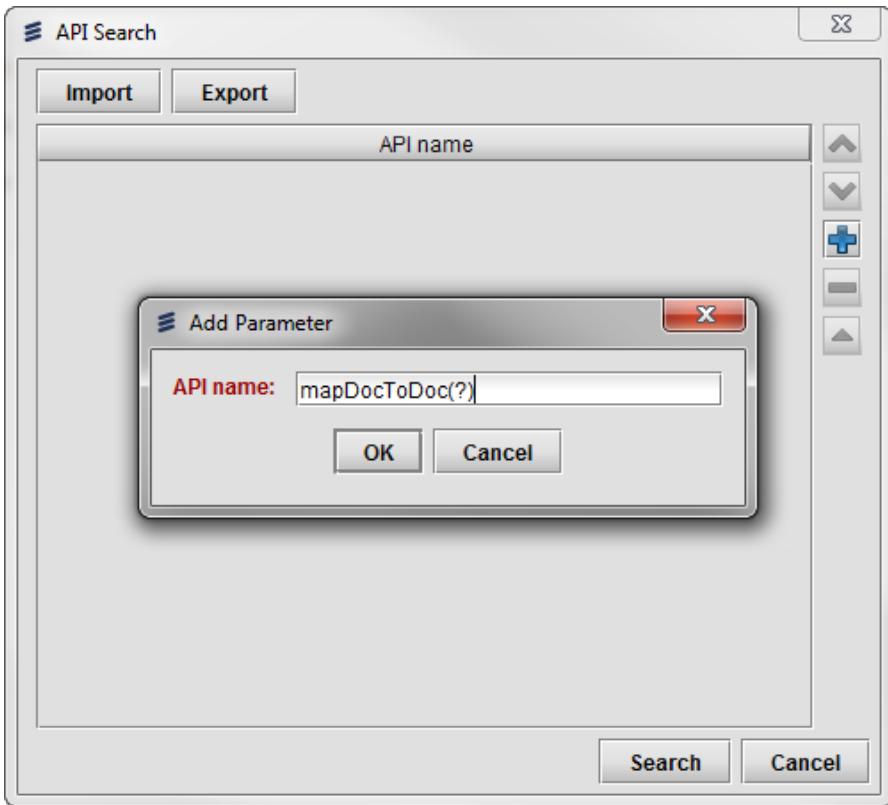
This example uses the API Search function to search for the mapDocToDoc() API, which is present in TestScript:

Name	Type	Description
blah	com.conceptwave.system.Boolean	

```

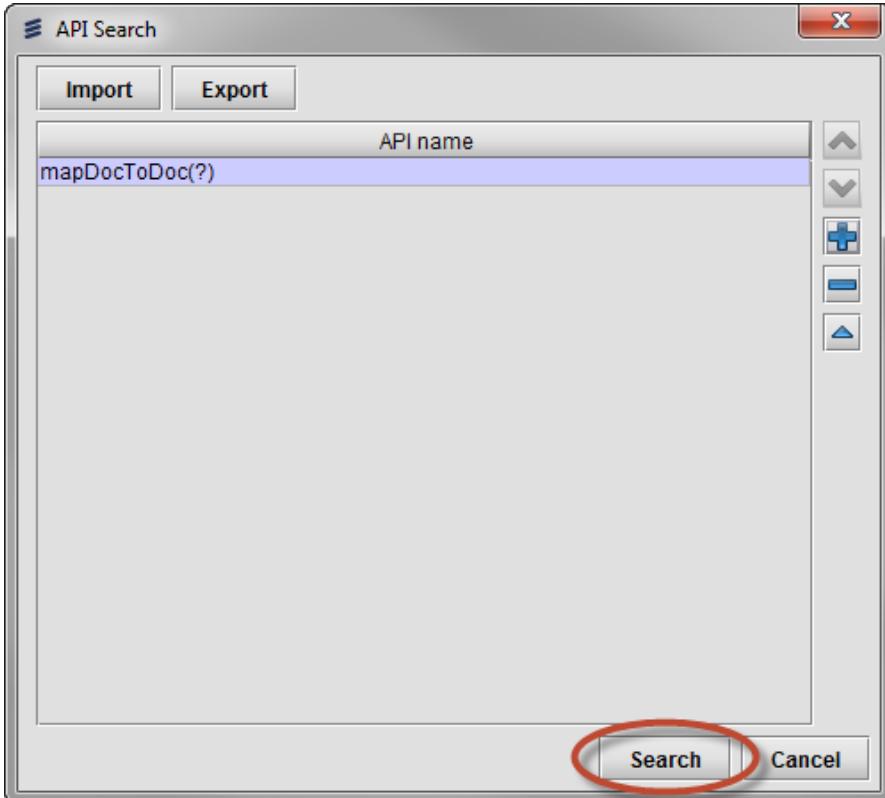
function TestScript(blah){
    mapDocToDoc(blah);
}
  
```

- The Add Parameter dialog appears. Enter the API that you want to search for in your metadata project in the **API name** field. For the **Replacement** field. See the [API Replace Syntax for Original and Replacement Fields](#) section for syntactical details that also apply to the API Search function.



Click the **OK** button to continue.

3. If you have more APIs you want to add, repeat the two previous steps steps of this procedure. You can also delete APIs by highlighting a row and clicking the **Delete** button. Otherwise, to search for your APIs in your metadata, click the **Search** button.



4. Check the log file for search results and to ensure that there were no errors:

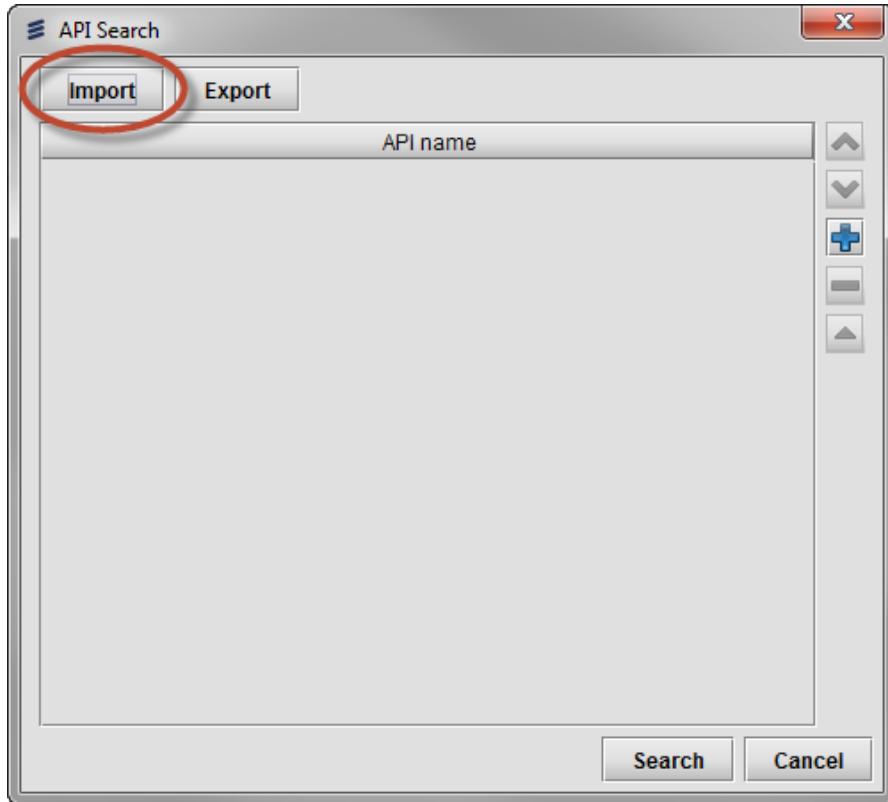
Result :

mapDocToDoc(?1) mapDocToDoc (TestNamespace.TestScript; Line 1) mapDocToDoc(blah);

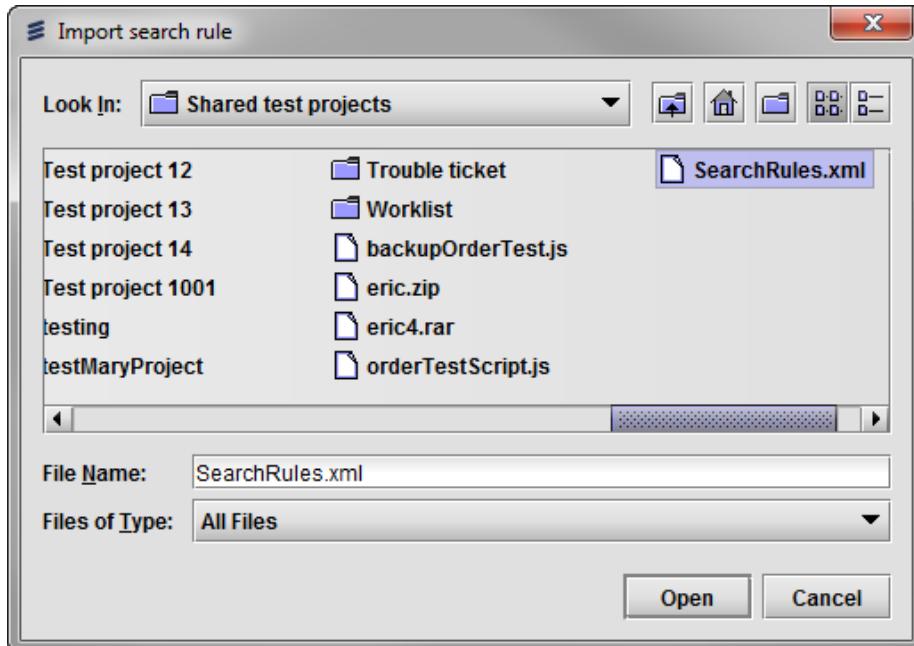
## Import Your API Search Parameters

You can import your API parameters that you have specified in an XML file by following these steps:

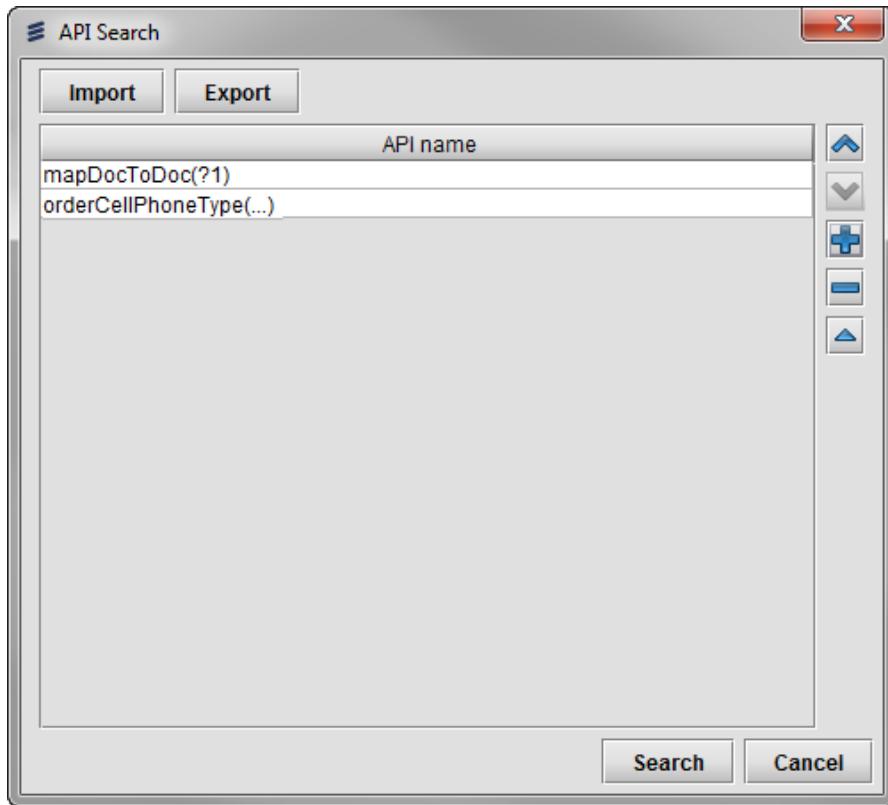
1. From the API Search screen, click the **Import** button.



2. The Import search rule dialog appears. Specify the XML file that contains your API parameters, and then click the **Open** button.



3. The API parameters specified in your XML file are imported to Velocity Studio.

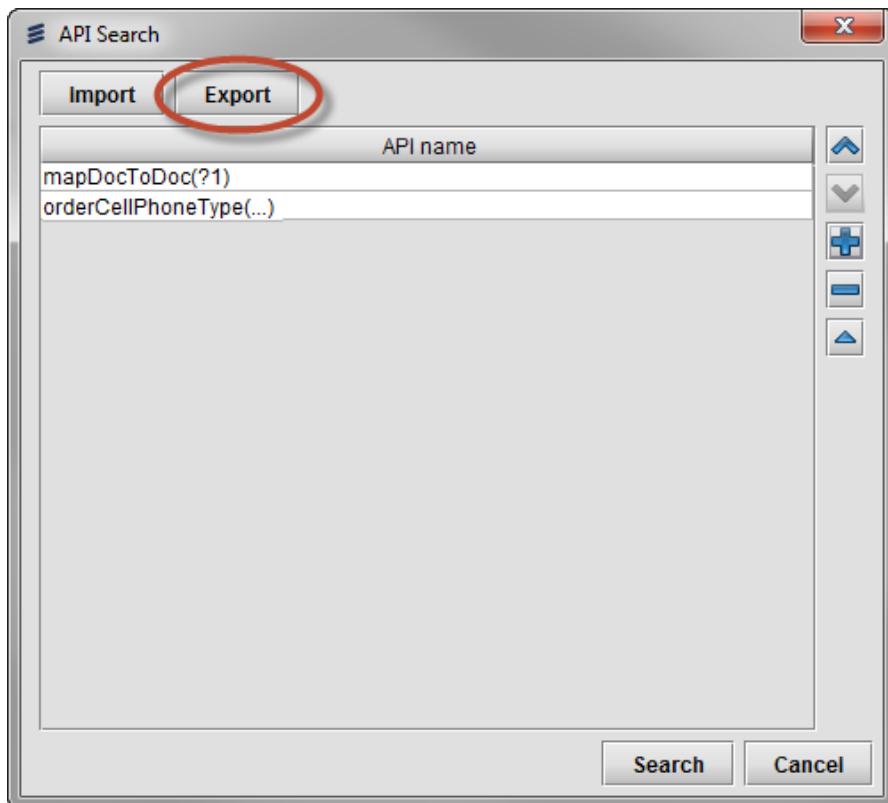


You can either click the **Add** button to add more API parameters, or click the **Search** button to perform your search.

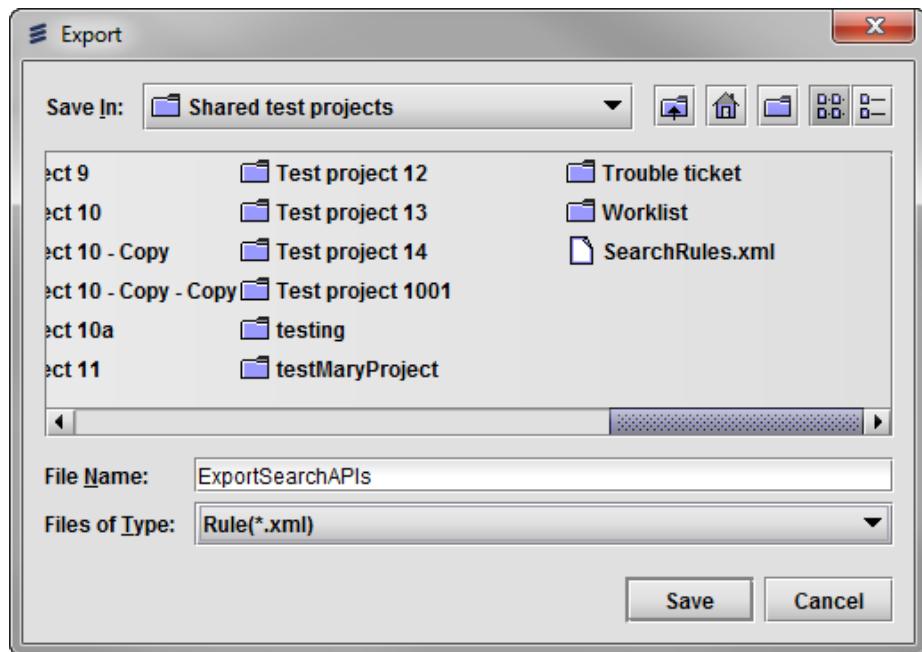
### Export Your API Search Parameters

You can export your API parameters that you have specified in Velocity Studio to an XML file by following these steps:

1. Enter your API parameters in the **API name** field as indicated in the [Use API Search](#) procedure, and then click the **Export** button.



2. The Export dialog appears. Specify the **File Name** that you want to save your API parameters to, and then click the **Save** button.



3. Your API parameters are exported as an XML file.

## API Replace

Clicking **Search > API Replace** from Velocity Studio's menu bar allows you to replace an API call with another one in your metadata project scripts, but not in the modules (templates). You can use this feature in a couple of ways:

- Replace deprecated APIs by using an internal list of migration (replacement) specifications
- Use your own migration specification for renaming or migrating your project metadata.

### API Replace Log File

The API replace function automatically creates a log file with the name CWQCAPIMigrateyyymmdd-hhmm, which is located in the `<installation_root_directory>\designer\env\log` folder, where:

- *yy* denotes the year
- *mm* represents the month
- *dd* indicates the day
- *hh* denotes the hour
- *mm* represents minutes

The log contains an HTML table with three columns:

- The old API name.
- The new API name.
- Change locations that are sorted by the old API names. The locations have the format: *namespace:element-name.method-name;line number*. If the element is a global script, the method name does not appear in the syntax.

The following is a sample log table row:

Document.myOldAPI	theDocumentCache.myNewAPI	myns:myel1.onStore; 21 myns:myel2.method; 3 myns:myscript;234
-------------------	---------------------------	---

#### Notes:

- The table is sorted by the first column
- The list in each cell in the third column is sorted alphabetically
- If there are rule parsing errors, they will appear in your log file

### API Replace Syntax for Original and Replacement Fields

When adding API parameters using the API replace function, both the **Original** and **Replacement** fields take the format *object-name.API-name(parameter specification)*, where the *object-name* and the dot after it are optional. The parameter specification is either an ellipsis (...) specifying any number of parameters or a list of parameters in format ?n, where n is from 1 to the required number of parameters.

**Note:** The old API must specify the parameter number in a natural sequence (that is, 1, 2, 3, ...), while in the new API can specify it in any order. Naturally, the API may not specify parameters at all – only the opening and closing parentheses.

The following table provides examples of using the syntax for the **Original** and **Replacement** fields, a description of the syntax, and how the APIs are migrated:

Original field	Replacement field	Description	Example
o1.api1(?1, ?2)	o2.api2(?2, ?1)	Replace the call to o1.api1 with the call to o2.api2, and reverse the two parameters. The migration only occurs if the o1 API has two parameters.	var x = b + o1.api1(c + d, 7); var y = x + o1.api1(x);  will be migrated as: var x = b + o2.api2(7, c + d); var y = x + o1.api1(x);
o1.api3(...)	o2.api4(...)	Replace the call to o1.api3 with the call to o2.api4, and keep all parameters. The migration occurs, independent of the number of parameters in the o3 API call.	var x = b + o1.api3(c + d, 7); var y = x + o1.api3(x);  will be migrated as: var x = b + o2.api4(c + d, 7); var y = x + o2.api4(x);
.api5()	.api6()	Replaces any occurrence of api5 with api6, as long as the API is preceded by a dot and has no parameters.	var x = b + o1.api5(c + d, 7); var y = x + o1.api5();  will be migrated as: var x = b + o1.api5(c + d, 7); var y = x + o1.api6();
.api5(?1,?2)	.api6(?1,?2)	Replaces any occurrence of api5 with api6, as long as the API is preceded by a dot and has exactly two parameters.	var x = b + o1.api5(c + d, 7); var y = x + o1.api5();  will be migrated as: var x = b + o1.api6(c + d, 7); var y = x + o1.api5();
api7(...)	api8(...)	Replaces any occurrence of api7 with api8, as long as the API is not preceded by a dot, independent of the number of parameters.	var x = b + o1.api7(c + d, 7); var y = x + api7();  will be migrated as: var x = b + o1.api7(c + d, 7); var y = x + api8();

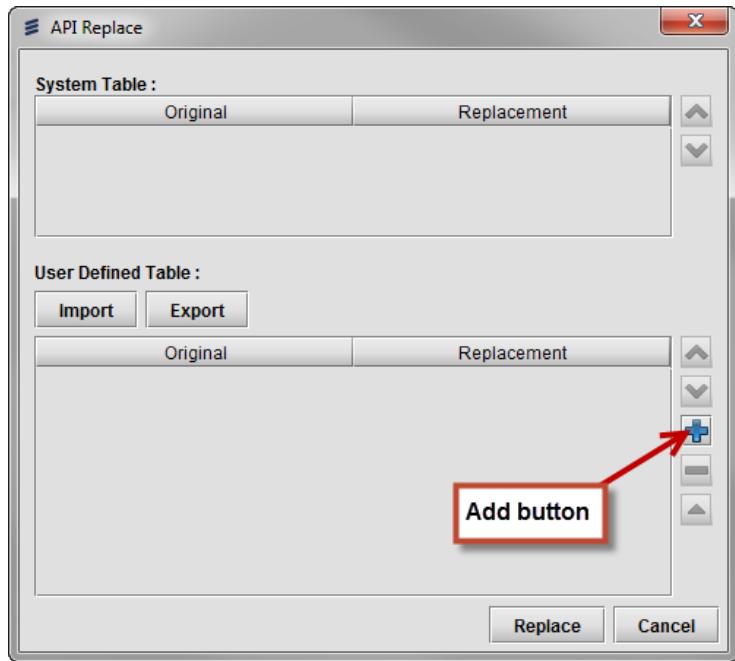
o1.api7(...)	api8(...)	Replaces any occurrence of o1.api7 with api8, independent of the number of parameters.	var x = b + o1.api7(c + d, 7); var y = x + o1.api7();  will be migrated as: var x = b + api8(c + d, 7); var y = x + api8();
a.b()	c()	Replaces any occurrence of a.b() with c().	a.b().x(); will be replaced by c.x();
a(...)	c(?1, "1", null, 3)	Replaces any occurrence of a with c. Note that the original and replacement APIs can have a different number of parameters, including null.	a(1, "x", "5") will be replaced by c(1, "1", null, 3);
a(...)	c(?2, 0, ?1)	Replaces any occurrence of a with c. Note that the original and replacement APIs can have a different number of parameters.	a(1) will be replaced by c(null, 0, 1);
api9(?2,?1)	api10(?1,?2)	Invalid specification, as the parameter numbers in the old API are incorrect.	

**Note:** If the old API specifies any number of parameters, the new API should specify the same.

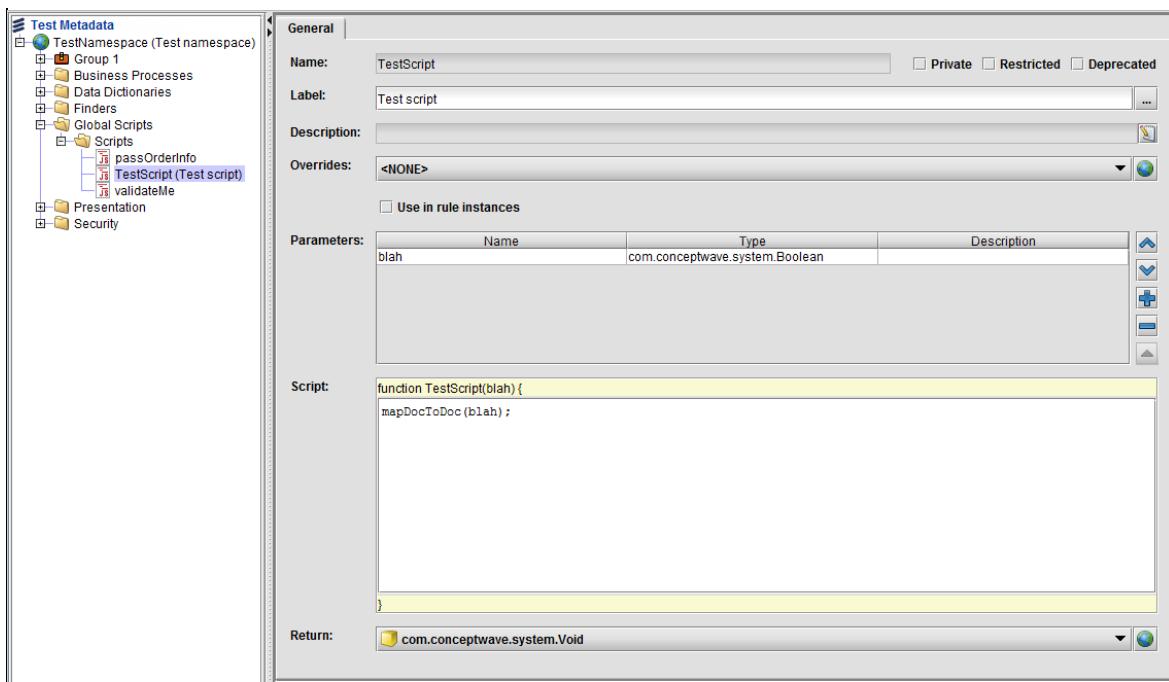
## Use API Replace

To use the API Replace function, complete the following steps:

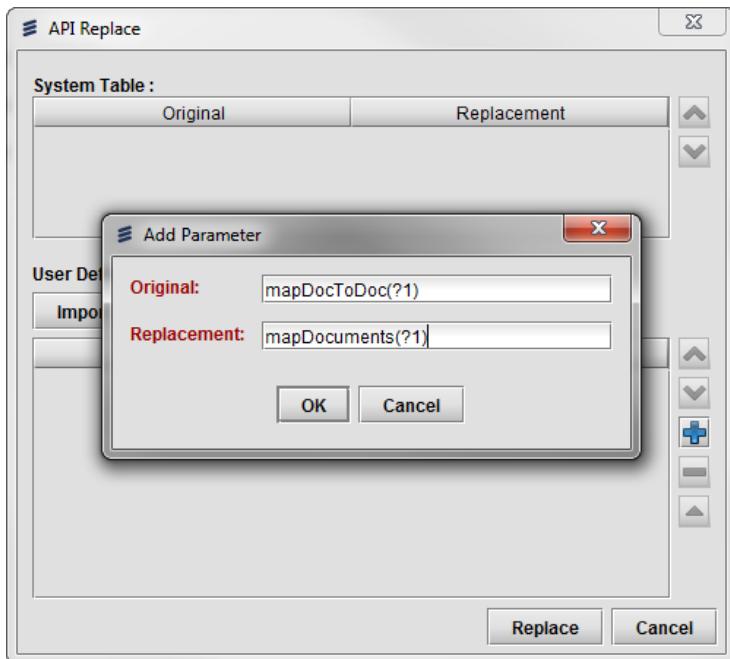
- From the API Replace screen, click the **Add** button.



This example uses the API Replace function to replace the mapDocToDoc() API, which is present in TestScript:

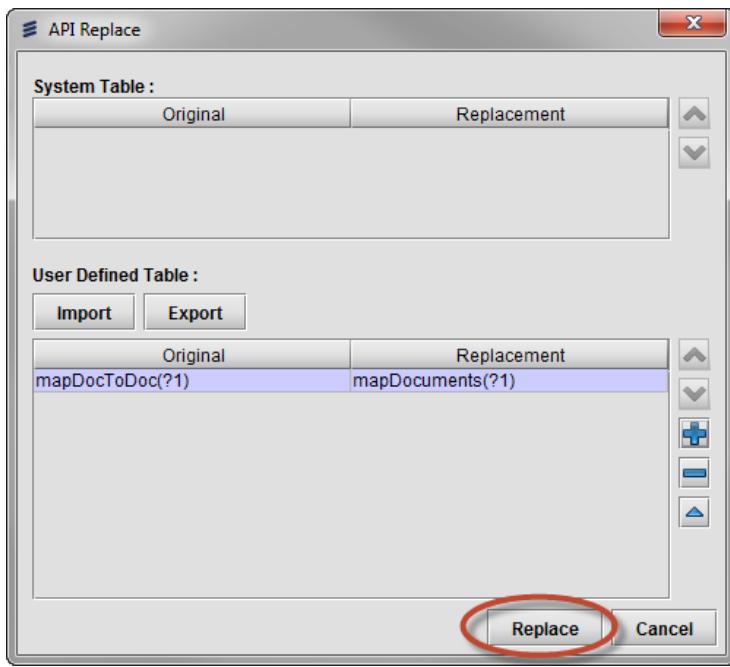


2. The Add Parameter dialog appears. Enter the old API that you want to replace in your metadata project in the **Original** field. For the **Replacement** field, enter the API that will replace the old one. See the [API Replace Syntax for Original and Replacement Fields](#) section for syntactical details.



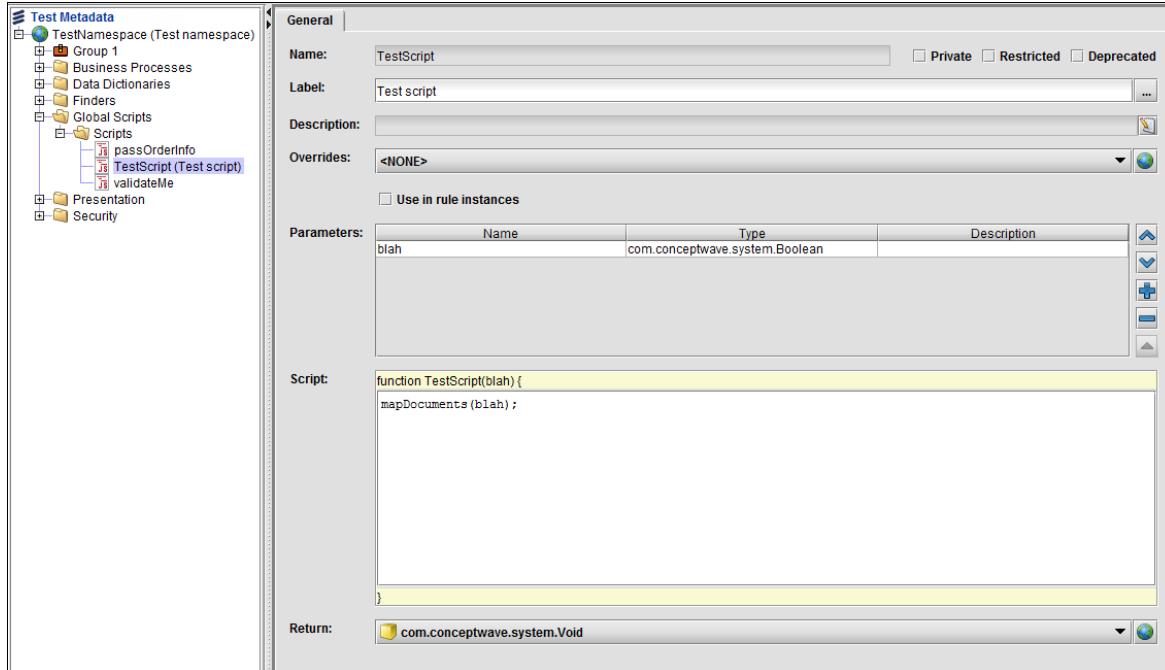
Click the **OK** button to continue.

3. If you have more APIs you want to add, repeat the two previous steps steps of this procedure. Otherwise, to replace your original APIs with their corresponding new APIs, click the **Replace** button.

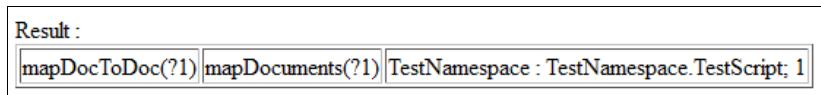


4. You can check whether the API Replace function was successful by performing one of the following actions:

- o Checking your scripts to see the change. In TestScript, the API change has been made.



- o Checking the log file to ensure that there were no parsing errors:



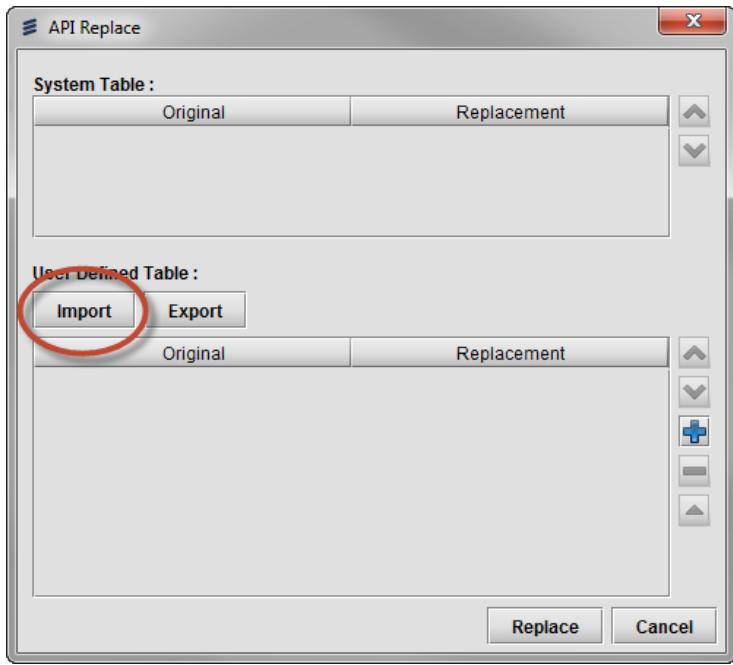
#### Notes:

- After all API replacement changes have been completed, the migration automatically stores all changed files.
- The migration cannot start under these conditions:
  - o If there are changes in memory that are not stored in the file system. You are prompted to save your changes before continuing.
  - o If the log file cannot be created.
  - o If the user-defined migration specification has errors.

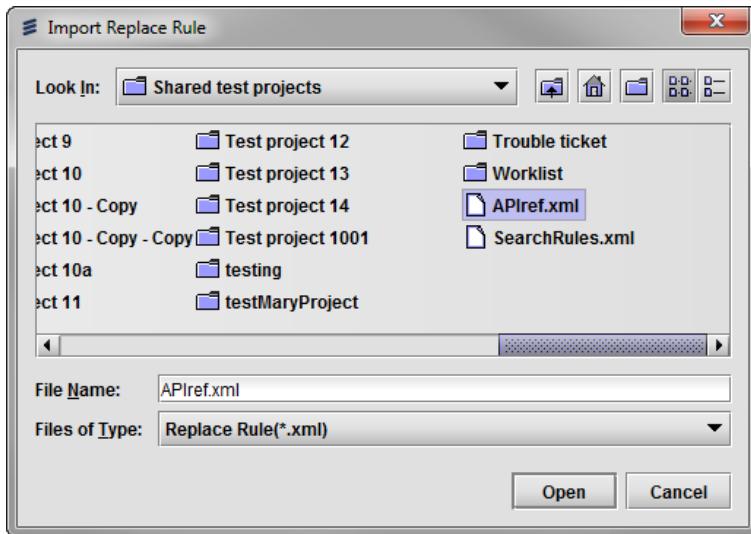
#### Import Your API Replace Parameters

You can import your original and replacement APIs that you have specified in an XML file by following these steps:

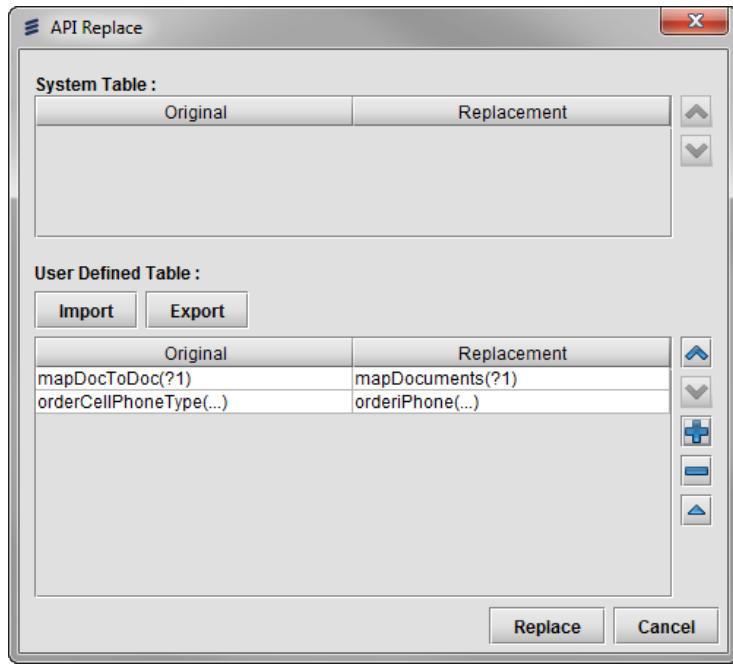
1. From the API Replace screen, click the **Import** button.



2. The Import Replace Rule dialog appears. Specify the XML file that contains your original and replacement APIs, and then click the **Open** button.



3. The original and replacement APIs specified in your XML file are imported to Velocity Studio.

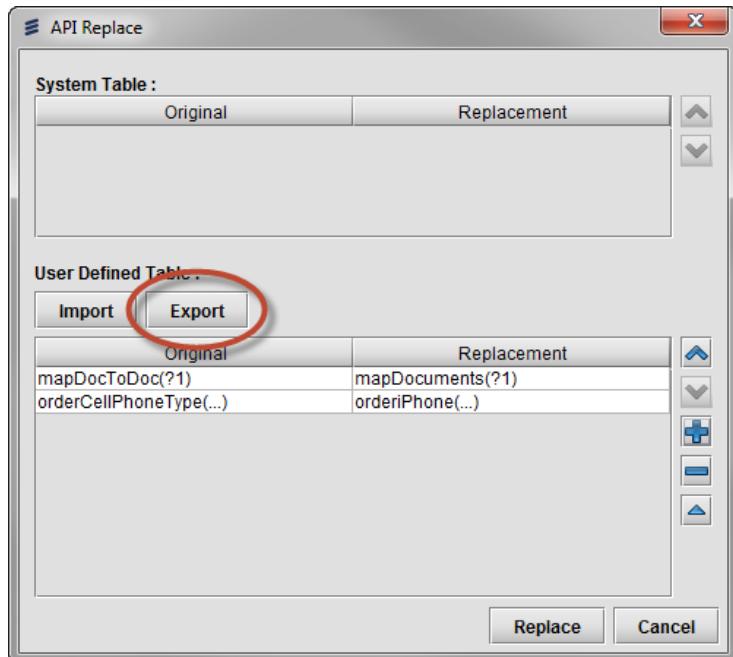


You can either click the **Add** button to add more original and corresponding replacement APIs, or click the **Replace** button to apply the changes to your metadata.

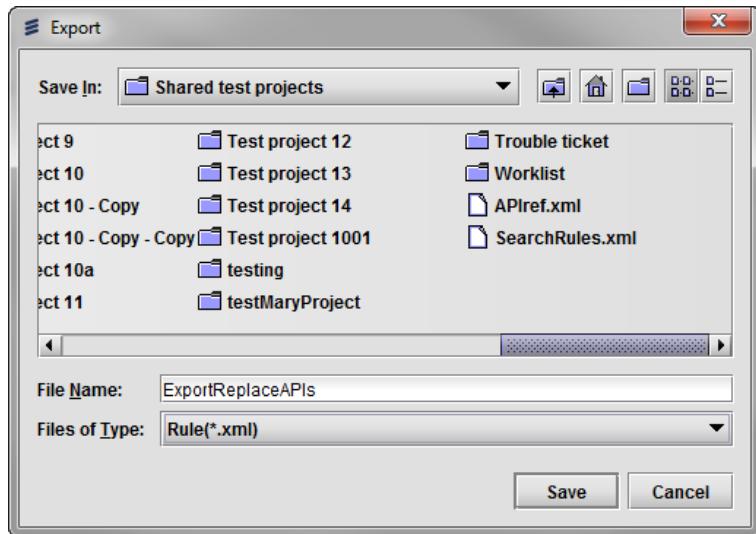
### Export Your API Replace Parameters

You can export your original and replacement APIs that you have specified in Velocity Studio to an XML file by following these steps:

1. Enter your **Original** and **Replacement** APIs in the fields provided, and then click the **Export** button.



2. The Export dialog appears. Specify the **File Name** that you want to save your original and replacement APIs to, and then click the **Save** button.



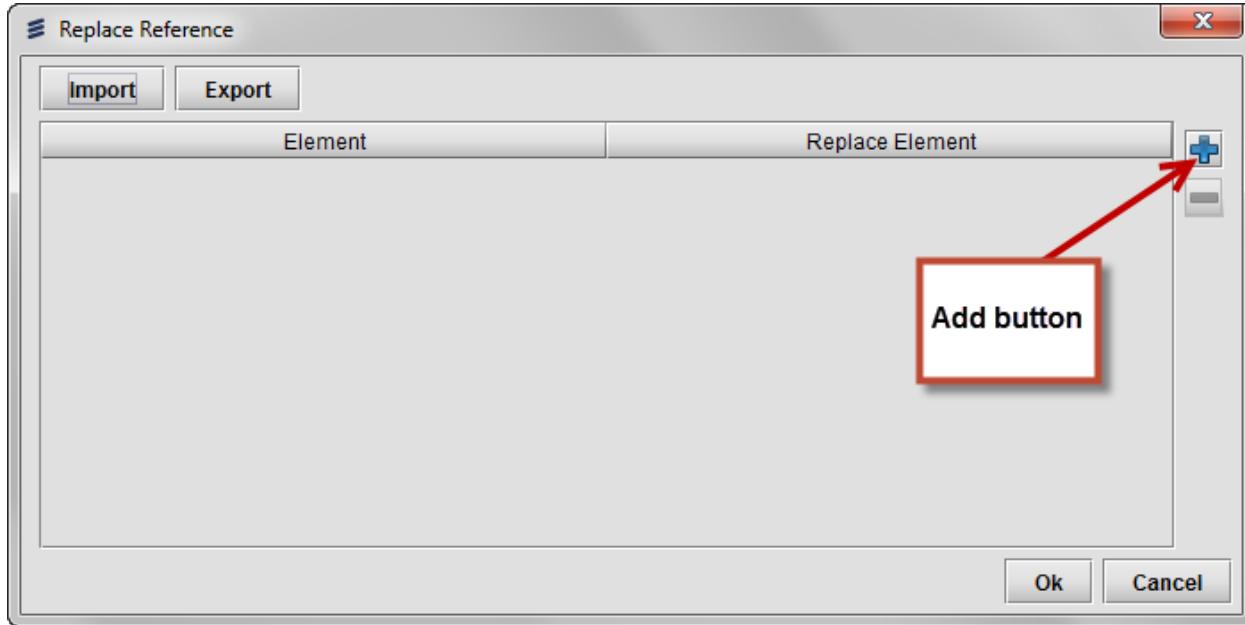
3. Your original and replacement APIs are exported as an XML file.

## Replace References

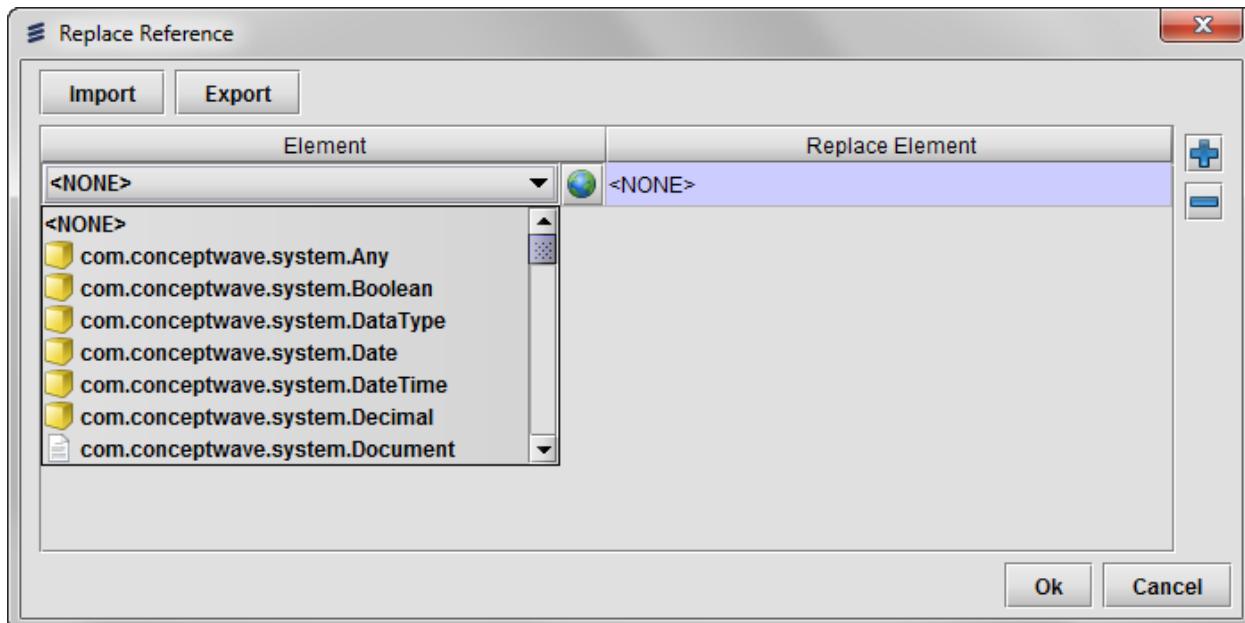
Clicking **Search > Replace References** from Velocity Studio's menu bar allows you to replace object (element) references in your metadata. This function searches your metadata and replaces an element reference with a replacement one that you have specified.

To use the Replace References function, do the following:

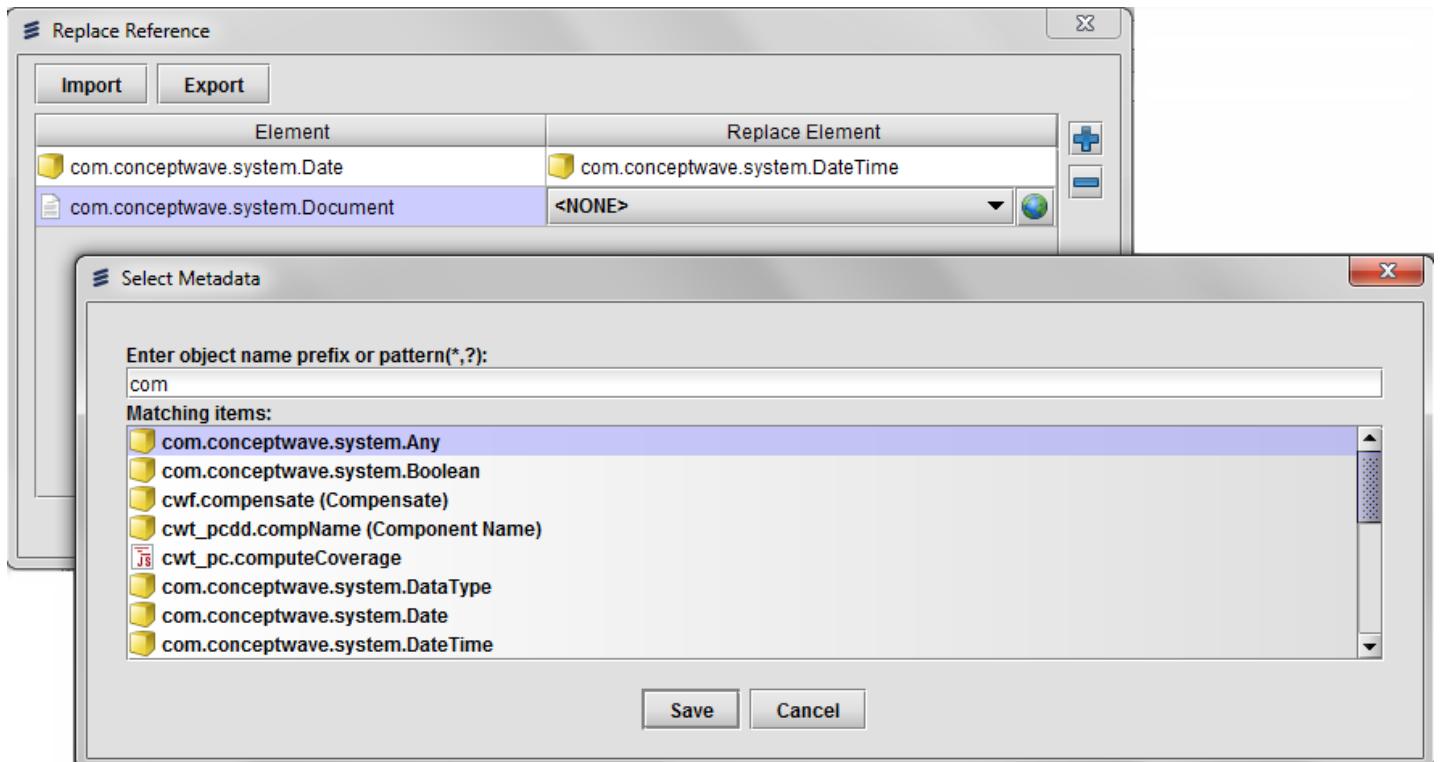
1. From the Replace Reference screen, click the **Add** button.



2. Click the **Element** field of the new row that you have added and select the element that you want from the list. Alternatively, you can click the **Select Element** button (with a globe icon) and enter the object name that you want in the field provided.



3. Repeat the previous step, but for the **Replace Element** field, to specify the element that will replace the one that you specified in the **Element** field.
4. If you have more elements that you want to add, repeat the first three steps of this procedure.

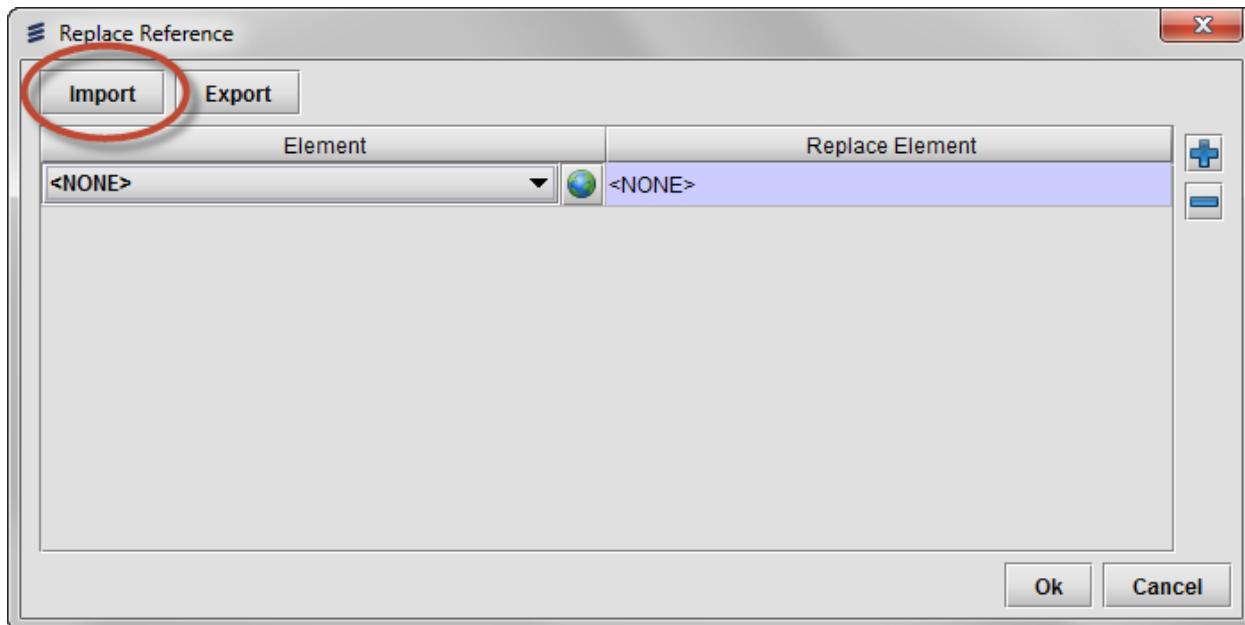


- When you have finished adding your elements and their corresponding replacement elements, click the **OK** button. The elements that you have specified to be replaced have been applied to your metadata.

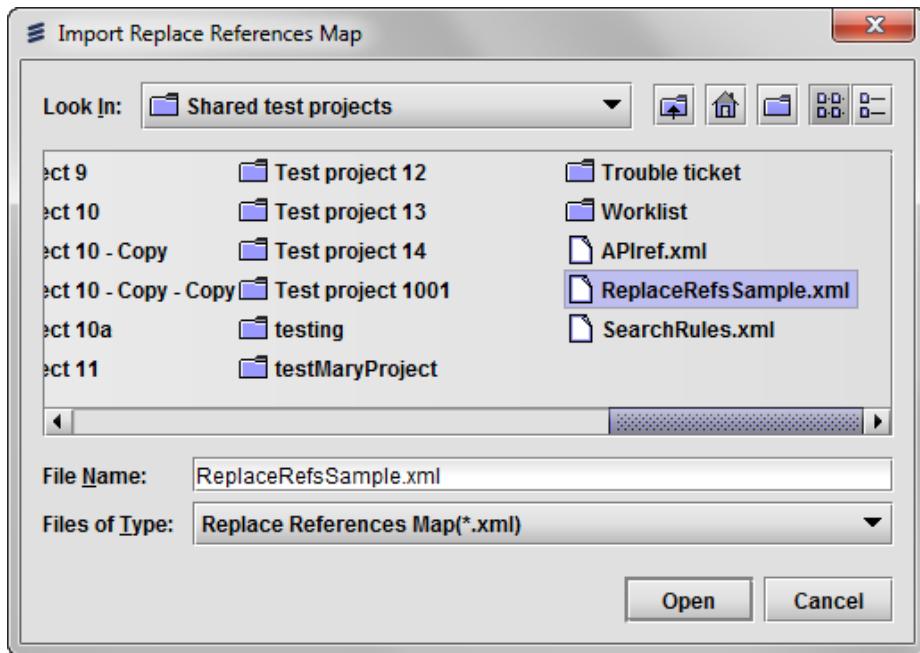
## Import Your Replace References

You can import element and replace element references that you have specified in an XML file by following these steps:

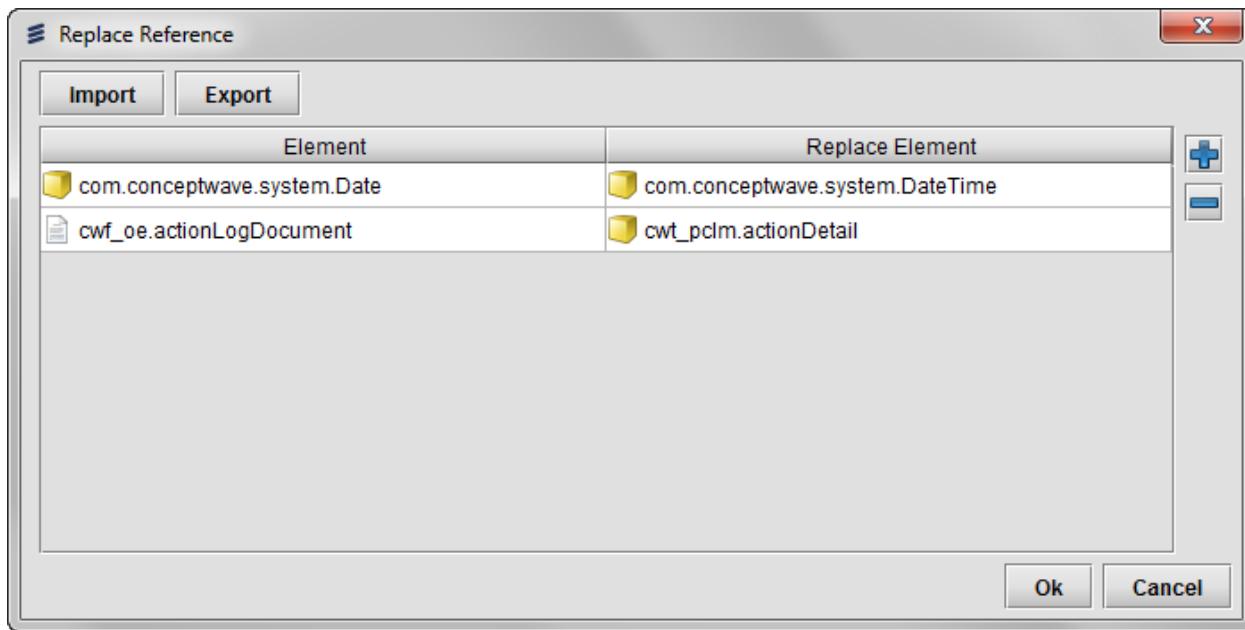
- From the Replace Reference screen, click the **Import** button.



- The Import Replace References Map dialog appears. Specify the XML file that contains your replace references and then click the **Open** button.



3. The replace references specified in your XML file are imported to Velocity Studio.

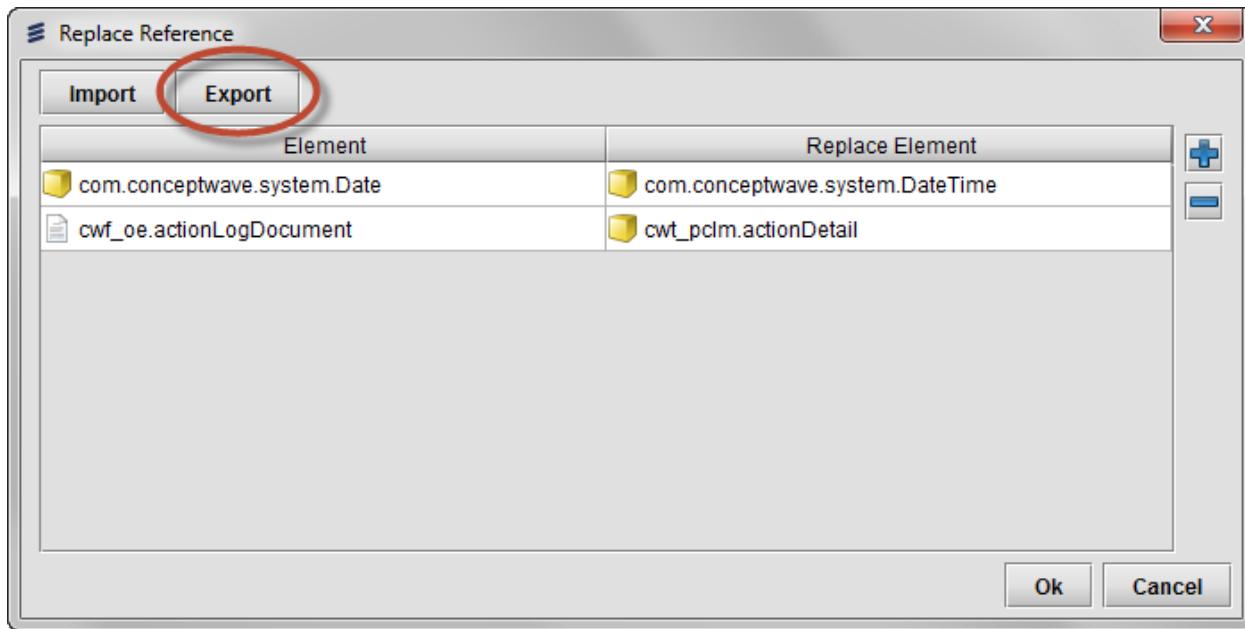


You can either click the **Add** button to add an **Element** and its corresponding **Replace Element**, or click the **OK** button to apply the element replacements to your metadata.

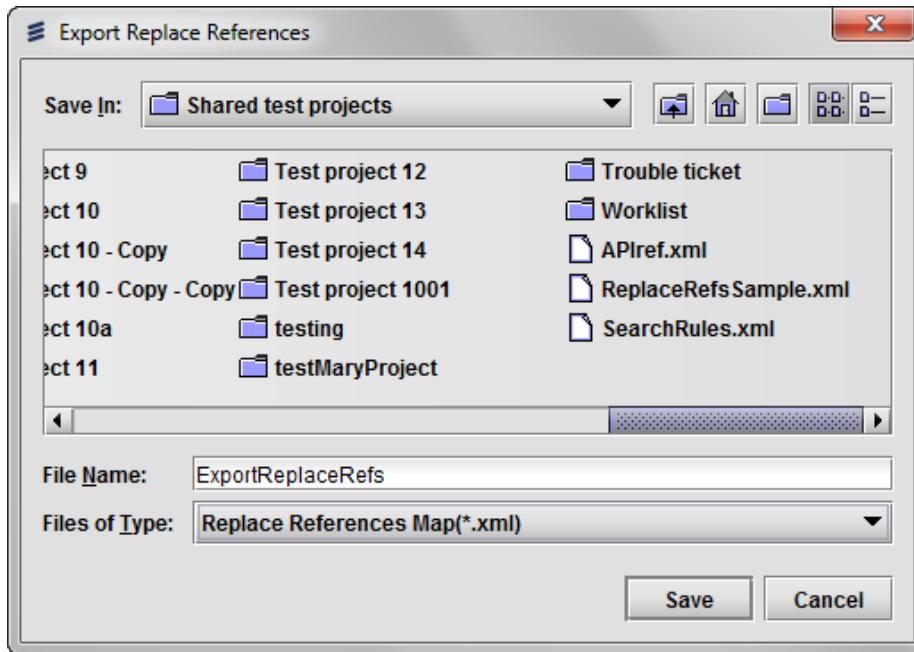
## Export Your Replace References

You can export the element and replace element references that you have specified in Velocity Studio to an XML file by following these steps:

1. Enter your **Element** and **Replace Element** references in the fields provided, and then click the **Export** button.



2. The Export Replace References dialog appears. Specify the **File Name** that you want to save your replace references to, and then click the **Save** button.



3. Your replace references are exported as an XML file.

## Replace Deprecated Methods

---

The Replace Deprecated function allows you to update your metadata JavaScript. The function searches your metadata and replaces only those function calls which are considered safe to be replaced automatically.

To use this function, perform these steps:

1. With your metadata project open, click **Search > Replace Deprecated** from the menu bar.
2. A prompt appears, indicating that the majority of static JavaScript methods that have been deprecated will be replaced. Additionally, your JavaScript code will be changed. Click the **Yes** button to continue.
3. A confirmation message appears. Click the **OK** button to continue.
4. Save your metadata.

### Notes:

- Velocity Studio automatically runs this feature the next time you open your metadata by setting the **Auto replace** property in **File > Preferences > Setting**.
- If you have clicked the **No** button to not run the Replace Deprecated feature, the **Auto replace** property remains unchecked.
- If you are replacing deprecated methods manually, Velocity Studio checks for the current system API version. If compatible, a message appears, indicating that you do not need to do any replacing, as you have the latest API version.

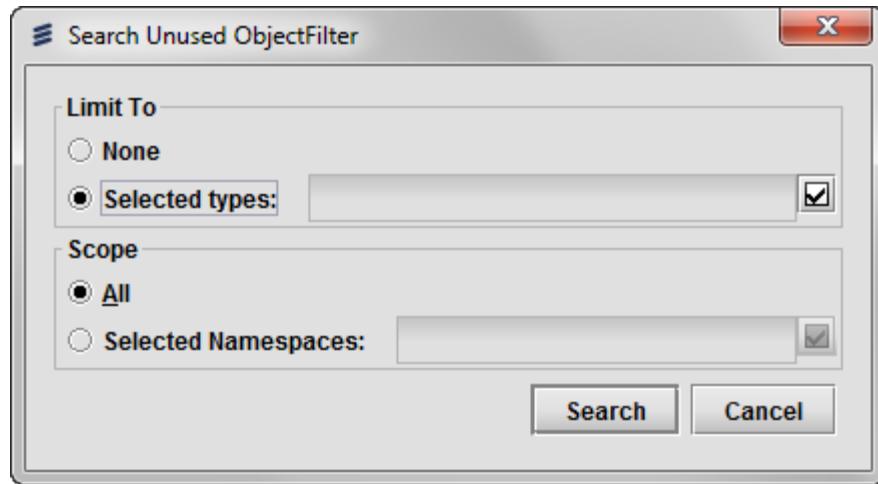
## Show Unused Objects

---

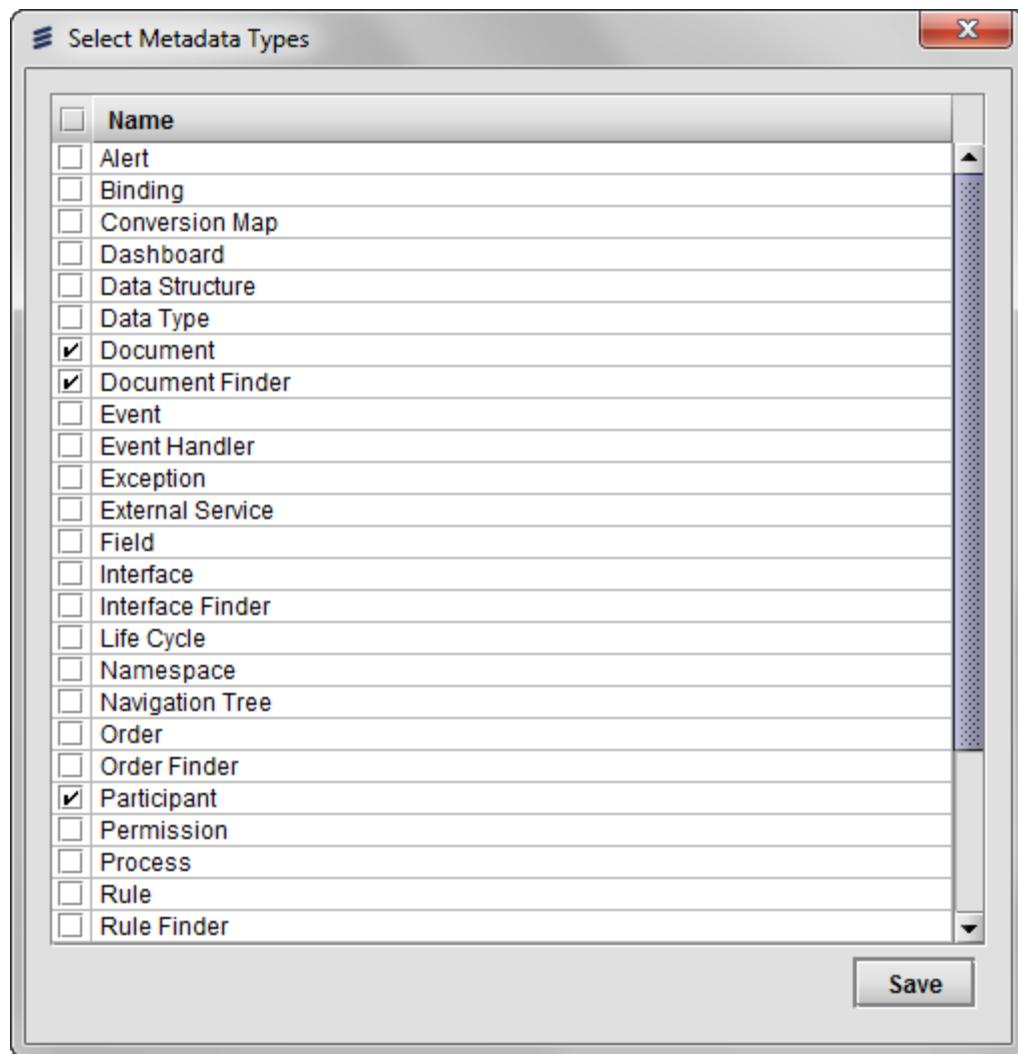
Clicking **Search > Show Unused Objects** from Velocity Studio's menu bar allows you to check an object's usage of reference and usage in scripts. This feature also allows you to filter by metadata type and namespace. It only searches for local, top-level metadata.

To use the Show Unused Objects function, do the following:

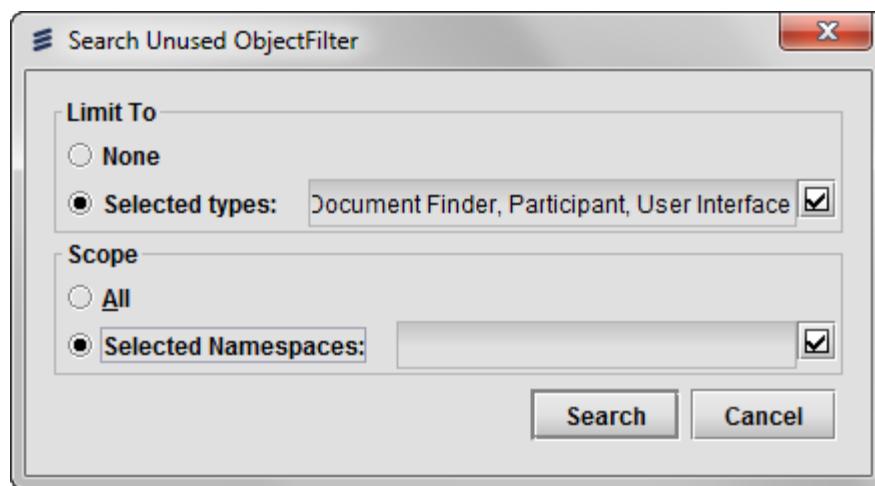
1. From the Search Unused Object Filter dialog, the default value for limiting the number of selected metadata types is **None**. To specify the metadata types that you want to filter, select the **Selected Types** option, and then click the checkbox option to launch the Select Metadata Types dialog.



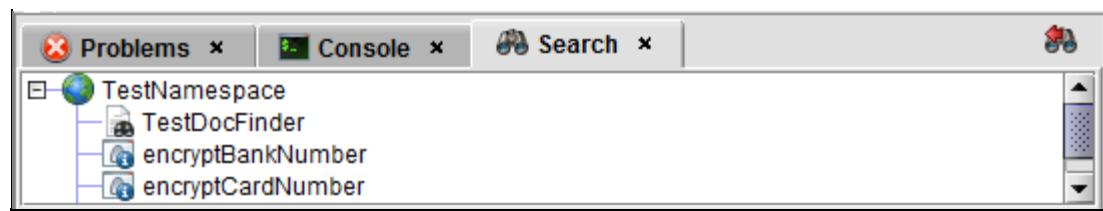
2. From the Select Metadata Types dialog, select the metadata types that you want to filter and then click the **Save** button to return to the Search Unused Object Filter dialog.



3. From the Search Unused Object Filter dialog, the default value for the **Scope** field is **All**. To specify the namespaces that you want to filter, select the **Selected Namespaces** option, and then click the checkbox option to launch the Select Namespaces dialog.



4. Select the namespaces that you want to filter from the Select Namespaces dialog, and then click the **Save** button.
5. Click the **Search** button to perform your search and display a list of unused objects. Your results appear in the Search tab.



## Velocity Studio - Runtime Menu

---

The Velocity Studio **Runtime** menu contains the following menu items and brief descriptions of each menu item:

### Run

Start running AVM within Velocity Studio using the project's metadata. This option is only visible if it has not been run.

### Stop

Stop running AVM within Velocity Studio using the project's metadata. This option is only visible if it has been run or debugged.

### Debug

Start running AVM within Velocity Studio with script debugging enabled; only visible if it has not started debug.

### Select Application

Select the application name, version, and the node for runtime.

### Deploy Application

Deploy the metadata to the database.

### Delete Application

Delete the deployed application from database.

### Difference Database Application

Find the difference in deployed and current application.

### Open Database Application

Open the deployed application from database.

### Build Deployment WAR

Package the metadata into deployment WAR file.

### Build Library JAR

Build metadata as a library, and export as a JAR file.

### Build Deployment JAR

Build a deployment file that contains all of the libraries included in the metadata, and export as a JAR file.

### Analyze Processes

To analyze metadata processes against deployed processes.

### Clean Non-Running Revisions

Remove process revisions that have no running instances.

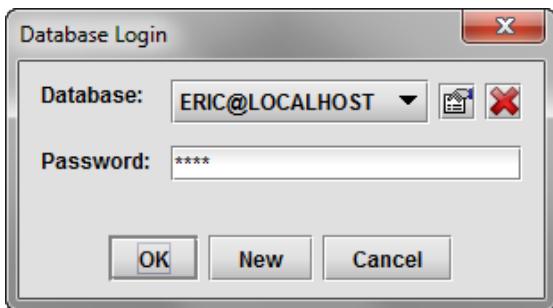
### Clean All Process Revisions

Remove the process revisions that do not have active or suspended processes in the database.

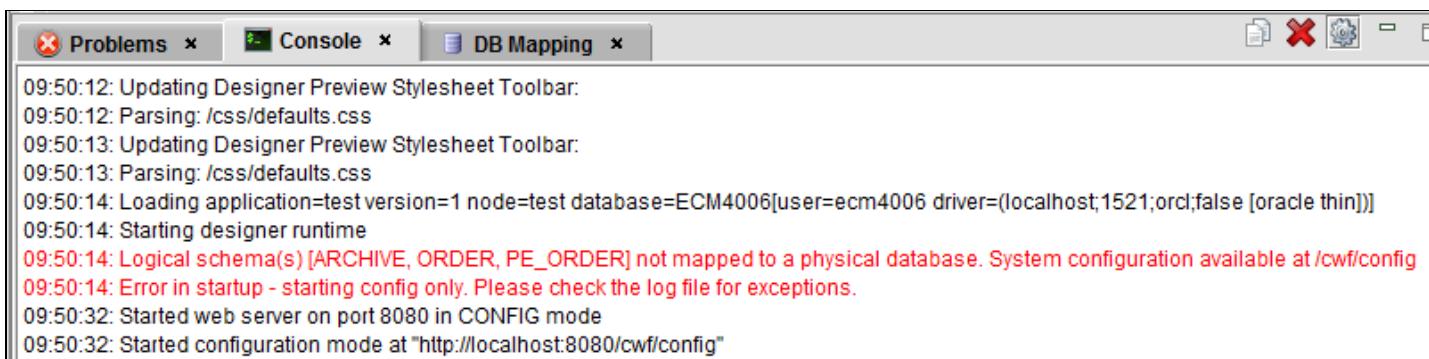
## Run

Clicking **Runtime > Run** allows you to run the project's metadata in Velocity Studio. The following are the steps to run the metadata:

1. Click **Runtime > Run** from the menu bar.
2. Login to the database associated with your project . If the project is not connected to the database, a **Database Login** dialog appears prompting you to login to the database. Refer to [connecting to database](#) for details.



3. The [Select Application](#) dialog appears. Select the application's name, version and the node. While the metadata is interpreted by AVM to start running, the metadata popup appears, and then disappears when startup has finished loading.
4. The console displays information on starting runtime. It displays the node, database, and the port information at which the Web server runs the application.



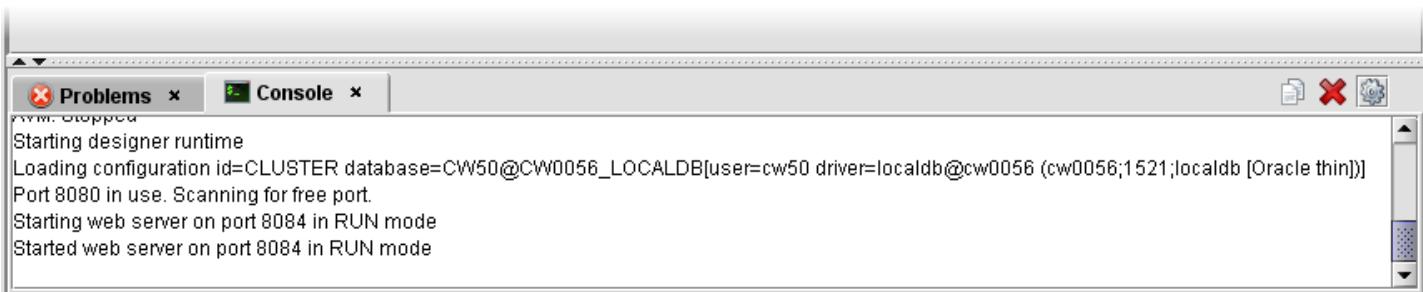
5. The application starts in the configuration mode; login to the System Configuration application and set up the [database connection configuration](#).
6. When done with configuration, restart the runtime in Velocity Studio.
7. Upon successful start of runtime in Velocity Studio, you can access the started Web application with your Web browser. By default, the URL to access the application is:  
**http://hostname:port/cwf/** (for example, <http://localhost:8080/cwf/>) where *hostname* is the hostname or IP address of the machine and *port* is the HTTP Port of the runtime AVM.

### Notes:

- o In runtime, the **Back** and **Forward** buttons are enabled, allowing you to switch between objects. Additionally, you can perform sorting functions in runtime.
- o Refer to [installation user guide](#) for more information.

### Port and URL Configurations

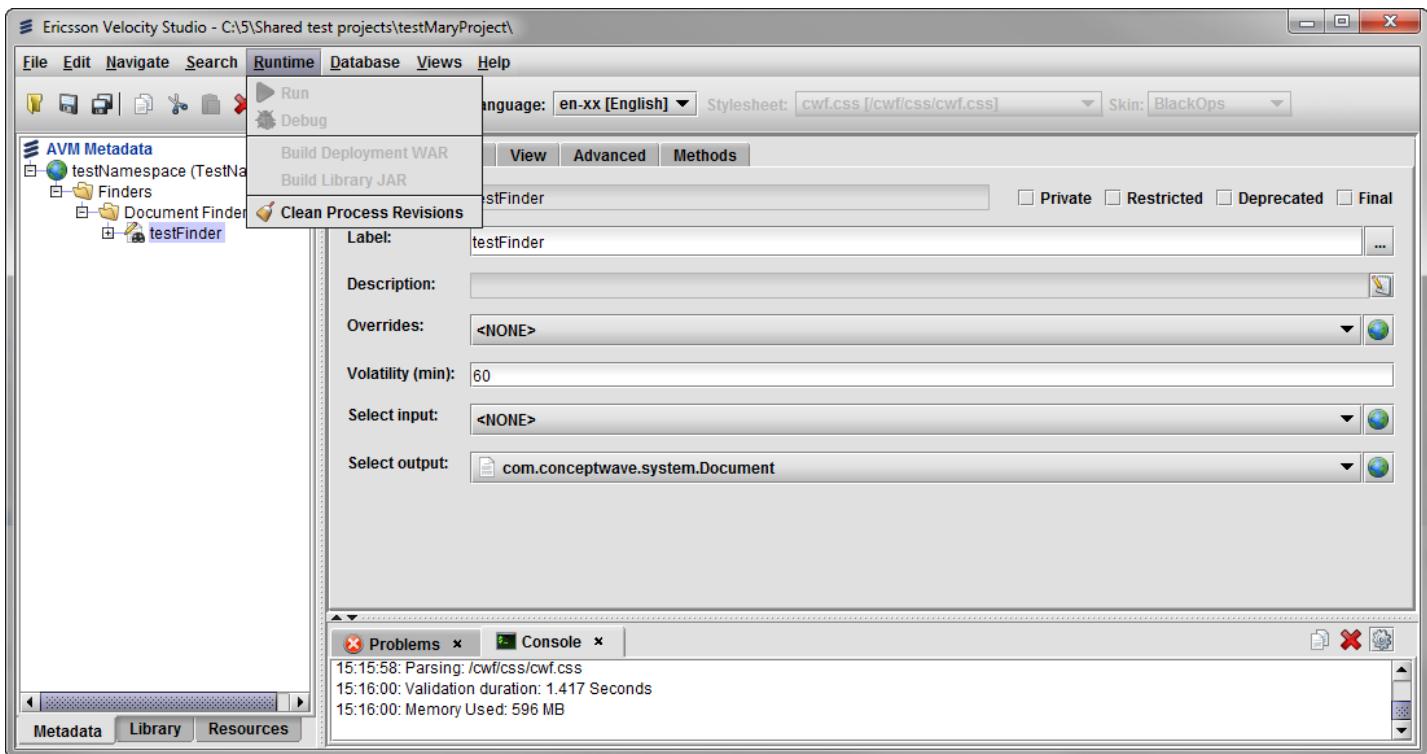
The HTTP Port number can be changed at [preferences](#). Furthermore, if the port number is already in use by the machine, the runtime will attempt to bind to the next available incremental port number. The port-binding attempts will be shown in console.



**Note:** User Interface metadata objects in the application can be mapped into a relative URL path using [URL Mapping](#) in the top-level application metadata node. For example, the default path "/" is defaulted to map to the default application login page `cwf_oe.login.UserInterface`. While you cannot change system URL mappings, you can add mappings to expose Page objects of your application as relative URLs.

#### Metadata must have no validation errors

**Note:** This menu command is only enabled when a project has been opened and no [validation errors](#) exist in the metadata. Otherwise, the Run menu is disabled (that is, greyed out). You must fix all errors in the metadata before being able to run the project.



#### Node ID Selection

The application metadata is run with a runtime configuration that is set in [Configuration application](#). By default, the [internal name](#) of the application is used as the base node name. You can change that node preference to run the metadata with a specific node configuration from the select application dialog. Refer to [node menu](#) for more information.

#### Failure to start metadata: CONFIG mode

If there are unforeseen errors during running of the metadata, the AVM stop running metadata but continue to start the Web server in **CONFIG mode**. This means neither the application metadata or library metadata are started (for example, <http://localhost:8080/cwf/> does not respond, which results in the [system administration application](#) being unavailable), but you can access the System Configuration application (<http://localhost:8080/cwf/config>).

Often, the error occurs because of mis-configuration (or the lack of configuration, initially). You can login to the System Configuration application to change any necessary runtime configurations, and can also change the Node ID of the configuration to apply to running the metadata (see above). Then, re-run the metadata.

#### License

The Run command, which can be found in `<installation_directory>\designer\env`, must be accompanied by a valid license file.

## **Stop Running Metadata**

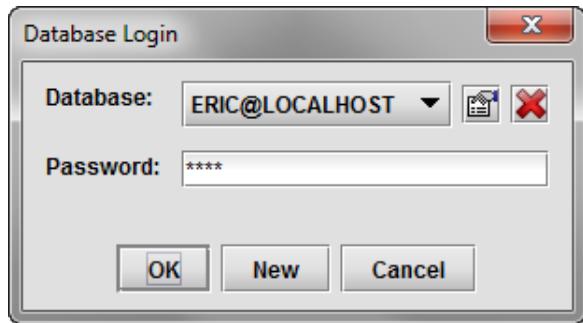
Once the metadata is running, in either RUN mode or CONFIG mode (can be seen in [console](#)), the Velocity Studio disallows any metadata changes. Most menu commands are disabled, and metadata objects become read-only in the Detail Pane.

To change the metadata again, you must stop running the metadata by selecting **Runtime > Stop** from the menu bar.

## Debug

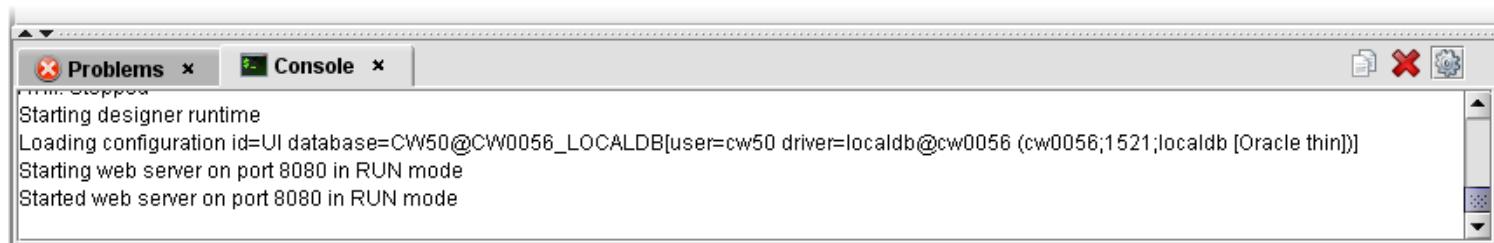
If there are no [validation errors](#), the project's metadata can be debugged with menu **Runtime > Debug**. In addition to the startup of the runtime similar to [Run](#), the Velocity Studio launches the third-party Rhino JavaScript debugger to perform the actual debugging.

If the project has not been connected to a database upon clicking the menu command, a **Database Login** pop-up dialog appears to prompt for database login before the metadata can be debug. Refer to [Database Connect](#) for details.

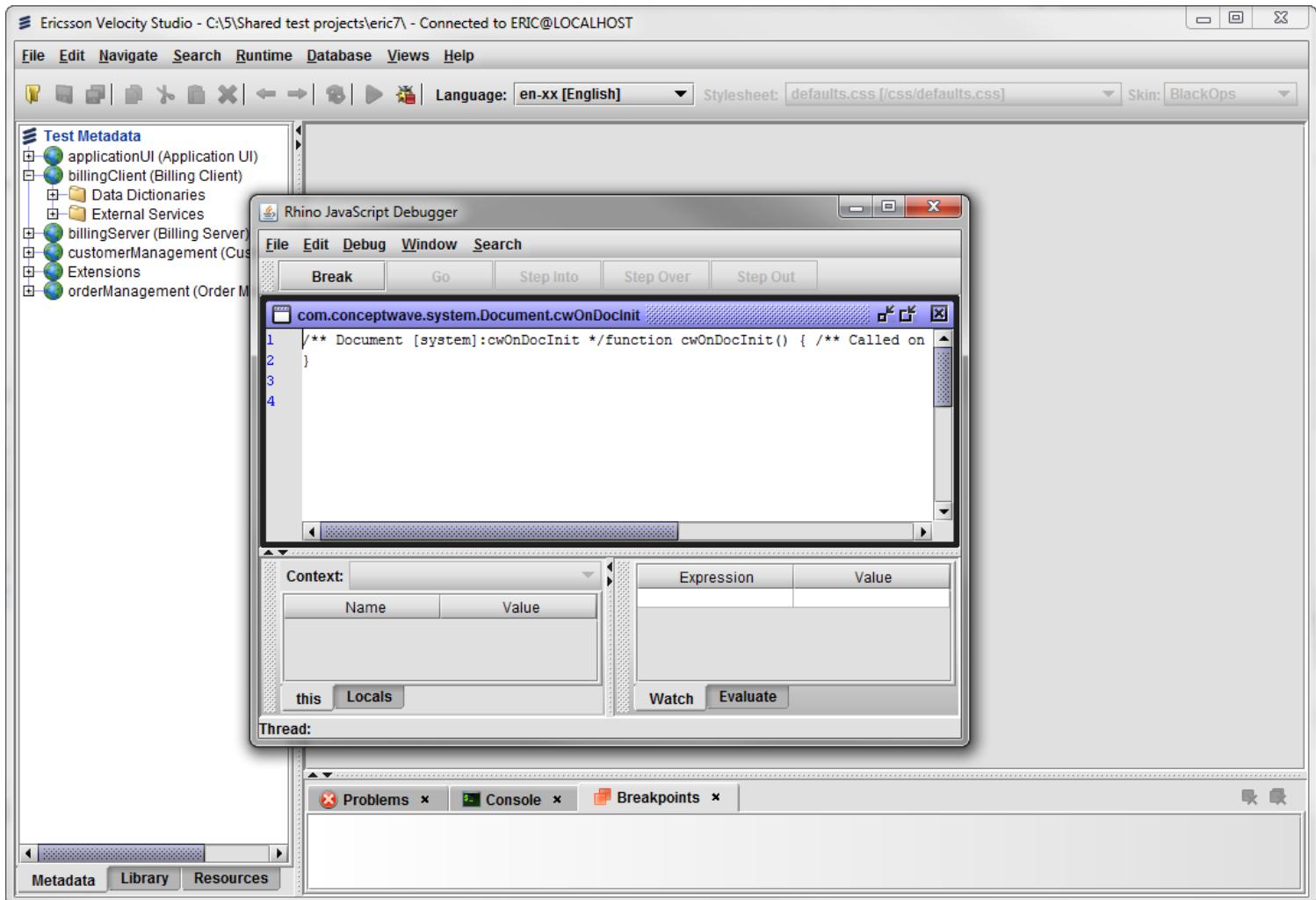


While the metadata is interpreted by AVM to start running, the metadata popup appears, and then disappears when startup has finished loading.

The console displays information on starting runtime. Note the port at which the Web server runs the application.



Similar to running metadata, you can access the started Web application with Web browser upon successful start of runtime in Velocity Studio. In addition, the Rhino JavaScript Debugger is launched in a separate window.

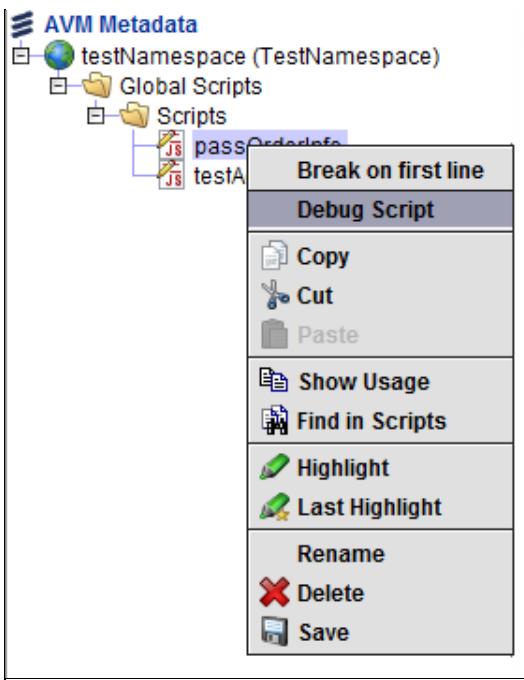


Closing the Rhino JavaScript Debugger will also stop the metadata.

Also similar to running metadata, the **Runtime > Debug** command:

- may have [Port and URL Configurations](#)
- must not have [validation errors in metadata](#)
- may [select Node ID](#) in preferences
- may fail to start such that [CONFIG mode](#) is run instead
- requires a [runtime license](#)
- can be stopped by menu [Stop](#)

You can also right-clicking a script in Velocity Studio and then selecting **Debug Script**, which allows for the single script to be debugged.



Runtime starts when you use this menu option, but workflow engine-processing does not occur in the background.

**Note:** If you use the **Debug Script** menu option, the **Update** button does not appear in your script's General tab in Velocity Studio while you are running the script. However, if you run the debugger first (that is, click **Runtime > Debug** from the menu bar), and then use the **Debug Script** menu option, the **Update** button appears in your script's General tab, allowing you to make changes to your script as you debug it.

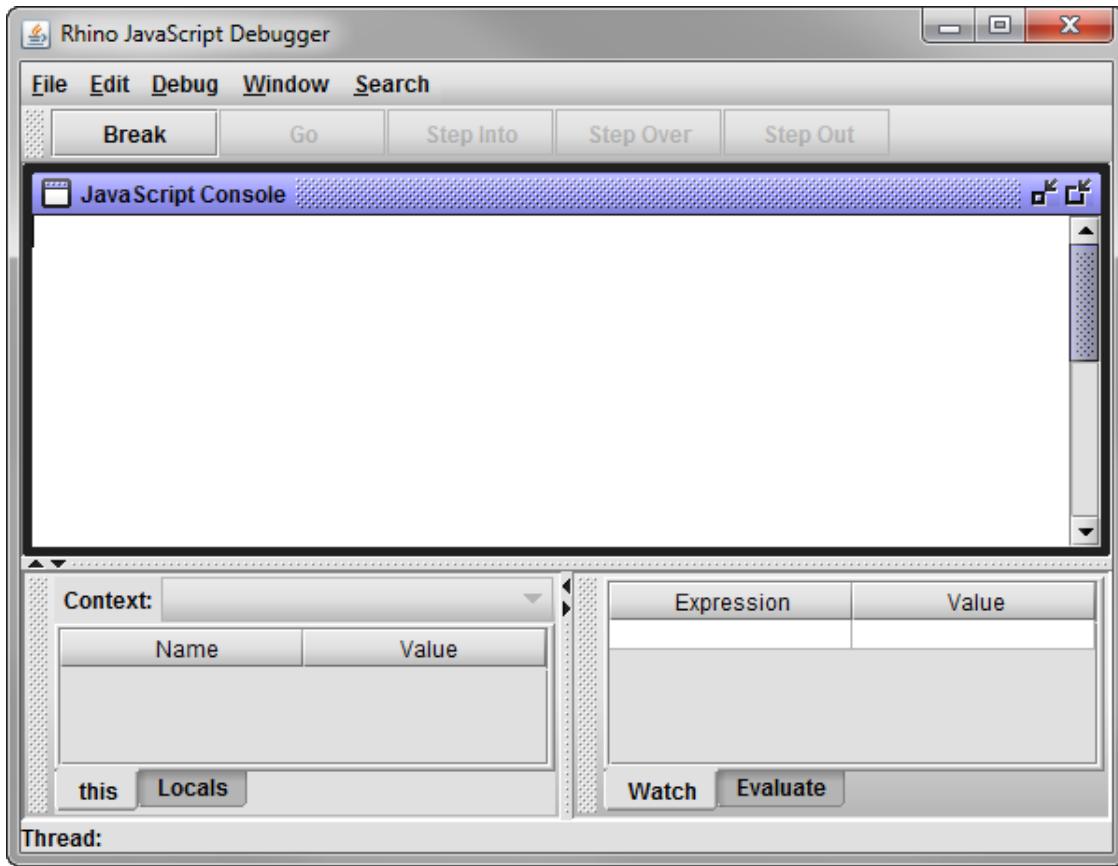
## Using Rhino Debugger

A full description of how to use the Rhino JavaScript Debugger can be found at the following Web site:

<http://www.mozilla.org/rhino/debugger.html>

The Rhino JavaScript Debugger can debug scripts running in multiple threads and provides facilities to set and clear breakpoints, control execution, view variables and evaluate arbitrary JavaScript code in the current scope of an executing script.

For instance, as you start using the runtime (for example, logging into application in browser), the JavaScript invoked is displayed in the debugger.

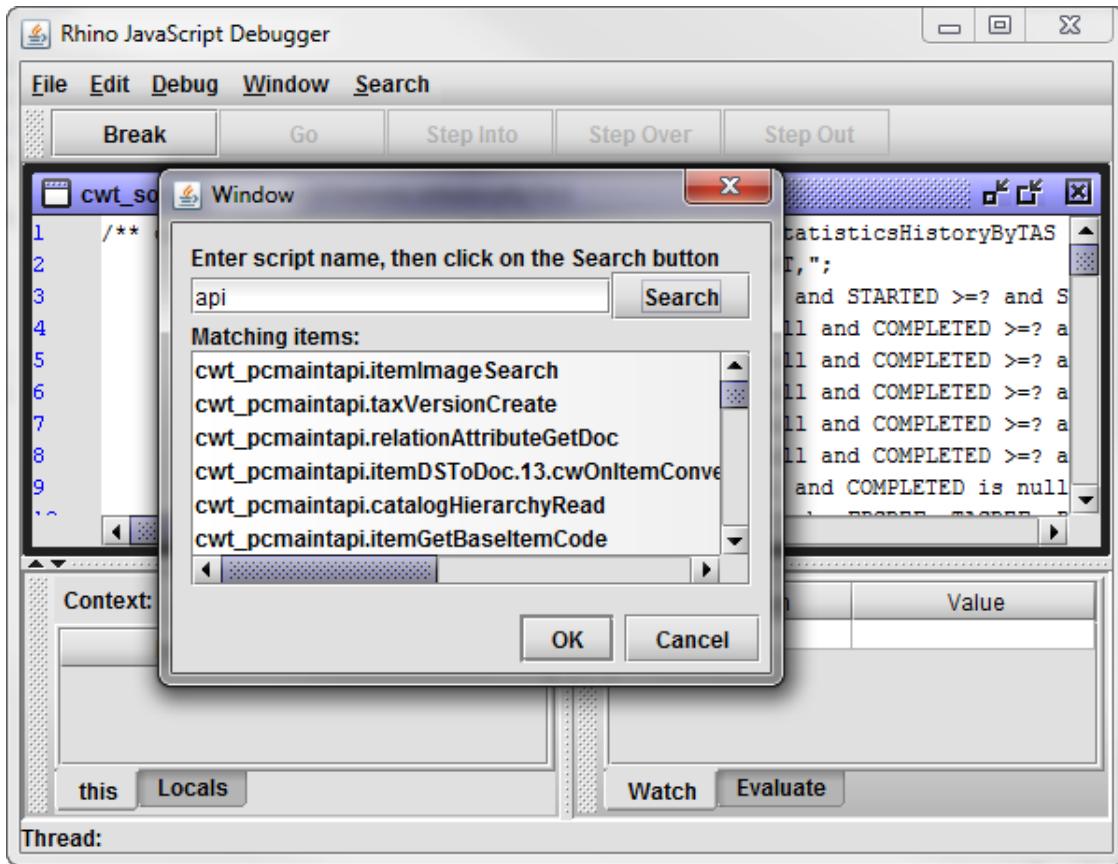


## Breakpoints

You can set breakpoints in a script in the Rhino JavaScript Debugger, which will stop on the first line of the script when this script is loaded for execution. Breakpoints can be set or unset without a debugger session running for all scripts except validation scripts. When the debugger starts, and when breakpoints are either set or unset, the **Breakpoints** tab appears at the bottom of Velocity Studio. Additionally, the **Break on first line** menu option for scripts is always displayed. For validation script debugging, breakpoints can only be set once the Debugger has been initiated.

## Search

The Rhino JavaScript Debugger also allows you to search for scripts at runtime, including catalog rules. To search for scripts, click **Search > Script** from the menu bar. A dialog box opens with the search functionality.



**Note:** For scripts that have parameters, initialized parameters are remembered between debugger sessions.

In the Rhino JavaScript debugger, you can set the **Debug UI Path** checkbox, which shows the full form path. By default, the flag is unchecked.

## Console Window

The debugger redirects the System.out, System.in, and System.err streams to an internal JavaScript console window which provides an editable command line for you to enter JavaScript code and view system output. The console window maintains a history of the commands you have entered. You may move backward and forward through the history list by pressing the Up/Down arrow keys on the keyboard.

## Controlling Execution

The debugger provides the following facilities for you to control the execution of the scripts you are debugging:

### Step Into

To single step enter any function calls, you may do any of the following:

- Select the **Debug > Step Into** menu item on the menu bar.
- Press the **Step Into** button on the toolbar.
- Press the F11 key on the keyboard.

Execution will resume. If the current line in the script contains a function call, control will return to the debugger upon entry into the function. Otherwise, control will return to the debugger at the next line in the current function.

### Step Over

To single step to the next line in the current function, you may do any of the following:

- Select the **Debug > Step Over** menu item on the menu bar.
- Press the **Step Over** button on the toolbar.
- Press the F7 key on the keyboard.

Execution will resume but control will return to the debugger at the next line in the current function or top-level script.

### Step Out

To continue execution until the current function returns, you may do any of the following:

- Select the **Debug > Step Out** menu item on the menu bar.

- Press the **Step Out** button on the toolbar.
- Press the F8 key on the keyboard.

Execution will resume until the current function returns or a breakpoint is hit.

#### *Go*

To resume execution of a script, you may do any of the following:

- Select the **Debug > Go** menu item on the menu bar.
- Press the **Go** button on the toolbar.
- Press the F5 key on the keyboard.

Execution will resume until a breakpoint is hit or the script completes.

#### *Break*

To stop all running scripts and give control to the debugger you may do any of the following:

- Select the **Debug > Break** menu item on the menu bar.
- Press the **Break** button on the toolbar.
- Press the Pause/Break key on the keyboard.

#### *Break on Exceptions*

To give control to the debugger whenever a JavaScript exception is thrown select the **Debug > Break on Exceptions** checkbox from the menu bar. Whenever a JavaScript exception is thrown by a script a message dialog will be displayed and control will be given to the debugger at the location the exception is raised.

#### *Break on Function Enter*

Selecting **Debug > Break on Function Enter** will give control to the debugger whenever the execution is entered into a function or script.

#### *Break on Function Exit*

Selecting **Debug > Break on Function Return** will give control to the debugger whenever the execution is about to return from a function or script.

#### *Moving Up and Down the Stack*

The lower-left pane in the debugger main window contains a **Context** combo-box which displays the current stack of the executing script. You may move up and down the stack by selecting an entry in the combo-box. When you select a stack frame the variables and watch windows are updated to reflect the names and values of the variables visible at that scope.

#### *Setting and Clearing Breakpoints*

The main desktop of the debugger contains file windows that display the contents of each script you are debugging. You may set a breakpoint in a script by doing one of the following:

- Place the cursor on the line at which you want to set a breakpoint and right-click with the mouse. This action will display a pop-up menu. Select the **Set Breakpoint** menu item.
- Click the line number of the line at which you want to set a breakpoint.
- If the selected line contains executable code a red dot will appear next to the line number and a breakpoint will be set at that location.

You may clear a breakpoint in a script by doing one of the following:

- Place the cursor on the line at which you want to clear a breakpoint and right-click with the mouse. This action will display a pop-up menu. Select the **Clear Breakpoint** menu item.
- Click the red dot or the line number of the line at which you want to clear a breakpoint.
- The red dot will disappear and the breakpoint at that location will be cleared.

#### **Viewing Variables**

The lower-left pane in the debugger main window contains a tab-pane with two tabs, **this** and **Locals**. Each pane contains a tree-table which displays the properties of the current object and currently visible local variables, respectively.

**Note:** Arrays are expandable in the debugger's variable view, which makes viewing data structures (for example, arrays within arrays) visible.

#### *This*

The properties of the current object are displayed in the **this** table. If a property is itself a JavaScript object the property may be expanded to show its sub-properties. The **this** table is updated each time control returns to the debugger or when you change the stack location in the **Context** window.

#### *Locals*

The local variables of the current function are displayed in the **Locals** table. If a variable is itself a JavaScript object the variable may be expanded to show its sub-properties. The **Locals** table is updated each time control returns to the debugger or when you change the stack location in the **Context** window.

## Watch Window

You may enter arbitrary JavaScript expressions in the **Watch** table located in the lower-right (dockable) pane in the debugger main window. The expressions you enter are re-evaluated in the current scope and their current values are displayed each time control returns to the debugger or when you change the stack location in the Context: window.

## Evaluation Window

The **Evaluate** pane located in the lower-right (dockable) pane in the debugger main window contains an editable command line where you may enter arbitrary JavaScript code. The code is evaluated in the context of the current stack frame. The window maintains a history of the commands you have entered. You may move backward or forward through the history by pressing the Up/Down arrow keys on the keyboard.

## Validation Script Debugging

Validation rules for data objects (documents, orders, data structures) can be created on different levels: as a validation method at the data type level, at the leaf level (for documents) or validation script within the object **Methods** tab.

At runtime, the validation scripts are triggered when validating a data object (document, data structure or order). For instance, before a document is **Saved** the validation occurs in the following sequence:

1. Data Type
2. Leaf validation
3. Document
4. Order

The usual way of debugging scripts is to set a breakpoint in Velocity Studio. However, this approach is not working for validation scripts because when metadata are compiled, a runtime script is created. For validation scripts, the system is not using the scripts created in Velocity Studio but instead is using this runtime script. This runtime script is an all-encompassing script that contains all scripts. This script is used during runtime and debugging. Breakpoints must be placed on the runtime script when debugging validation scripts.

To run the debugger, click on the Debugger icon  from the menu bar within Velocity Studio. The runtime script can be opened in Debugger through **Search->Scripts**. A dialog box is open that allows you to search for the runtime scripts. The names of the validation runtime scripts generated are:

### Data Type

The *.cwOnDataValidation* script is generated when the metadata is compiled. This script contains the validation methods for Data Types.

### Leaf and Document

The *.cwOnGenDocValidation* script is generated when the metadata is compiled. This script contains the validation methods for leafs and documents. At runtime, this script will validate at the leaf level and then the document level.

### Order

The *.cwOnOrdValidate* script is generated when the metadata is compiled. This script contains the validation methods orders.

### Order

The *.cwOnOrdValidate* script is generated when the metadata is compiled. This script contains the validation methods orders.

### List of Errors

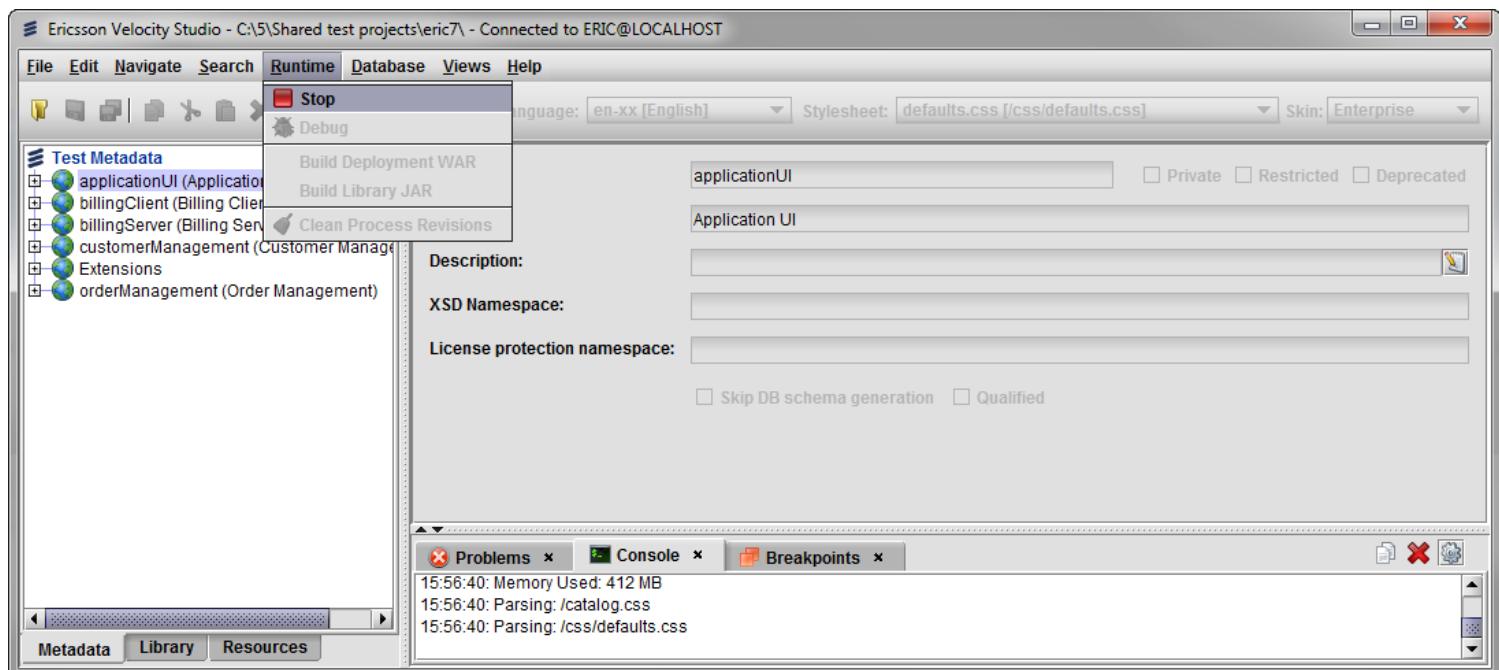
During validation, a list of errors are stored in **\$cwerr** (ValidationErrorResponse name) variable and can be retrieved with the following methods:

- **\$cwerr.getErrorList()**: This method returns an array of strings.
- **\$cwerr.toMessageArray()**: This method returns an array of **com.conceptwave.system.Message** objects.

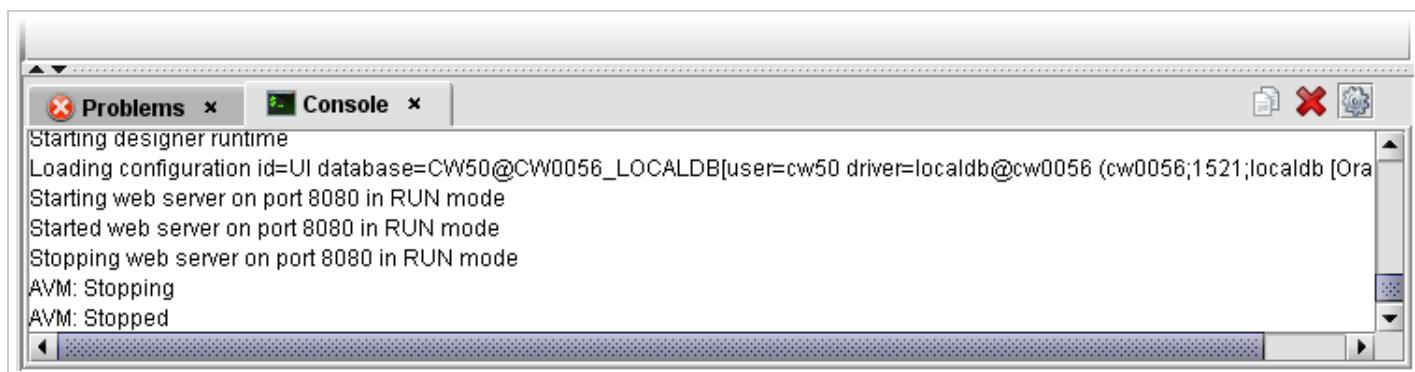
## Stop

After the project's metadata is run within Velocity Studio with menu [Run](#), the Velocity Studio disallows any metadata changes. Most menu commands are disabled, and metadata objects become read-only in Detail Pane.

Use the menu command **Runtime > Stop** to stop the runtime AVM. You can also click the **Stop** button () from the menu bar. Velocity Studio menus and metadata objects are allowed to be accessed and edited respectively again.



The console displays information on stopping runtime.



## Select Application

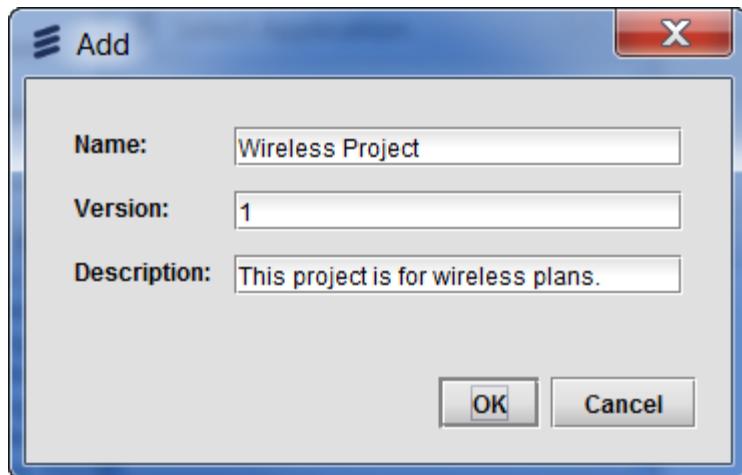
When running an application, after database connection you are prompted to select an application to connect to. The **Runtime > Select Application** menu option allows you to choose the application name, version, and the node to run the application. The application connections settings remain the same until you change applications, disconnect from the database, or load a different metadata project. To add new applications, follow these steps:

1. Click the **Runtime > Select Application** menu; the Select Application dialog appears.

The following table describes the fields available:

Field	Description
<b>Application</b>	This field denotes application's name that is based on the project's internal name. The project's internal name allows creating a group of related database applications with compatible configuration and database schemas. If the project's internal name is not set, an Application Select Error dialog provides the information that project's internal name must be set.  You can select different applications available in the drop-down list of this field.
<b>Version</b>	This field denotes the version of the application. One application can have different versions. Select one from the drop-down list.
<b>Node</b>	This field indicates the node name to be used in the <a href="#">System Configuration application</a> as cluster node. By default, the application's internal name appears as the node name. You can change the node name after logging into the System Configuration application.  After configuration is done, all the configured or active nodes appear in the drop-down list of this field.

2. Click the **New** button from the select application dialog; another **Add** dialog appears.



3. Enter the information in the available fields and click the **OK** button.

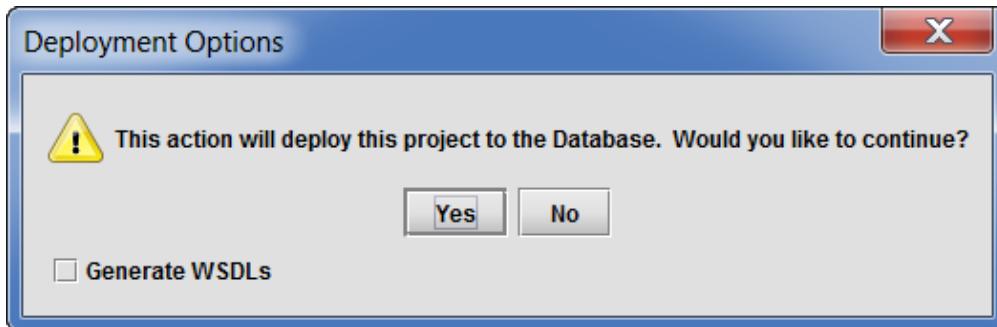
**Note:** You can change the name of the application.

4. The information appears on the Select Application dialog; click the **OK** button to save the application preferences.

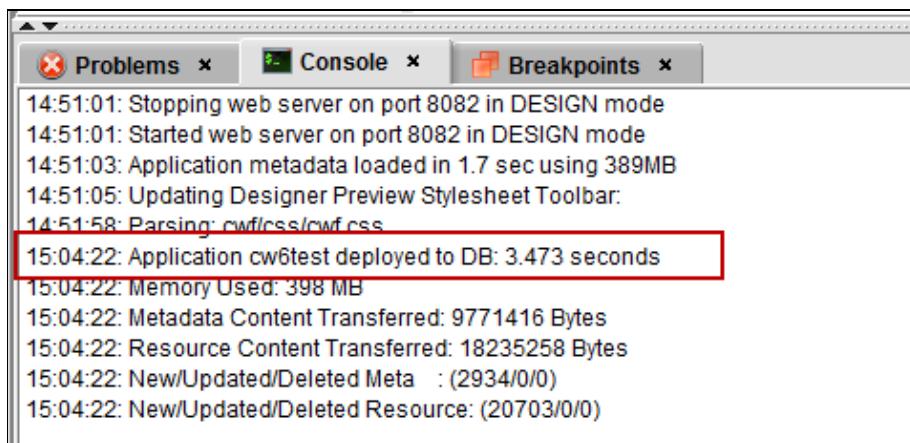
## Deploy Application

To run your metadata from an application server, the metadata needs to be in the database, instead of in the file system. Once you have created your metadata, you can deploy it to the database from within Velocity Studio. To deploy the application to the database, do the following in Velocity Studio:

1. Click **Runtime > Deploy Application**. The system will display a confirmation screen, as shown in the following figure.



2. If you want to generate the WSDLs and have them saved to the database, select the **Generate WSDLs** checkbox. If this checkbox is selected, the WSDLs and the schemas will be generated to the CW\_METADATA table in the database.
3. Click the **Yes** button. Once the deployment is complete, the system will display a message in the Console pane, as shown in the following figure.



**Note:** The **Deploy Application** command is only enabled when a project has been opened and no validation errors exist in the metadata. Otherwise, the command is disabled (that is, greyed out).

## Deploy through the Command Line

You can deploy your application using a command line:

- In Windows, use `deploy.cmd`
- In Unix or Linux, use `deploy.sh`

The following describes the syntax of the command-line interface and the available arguments. A shortcut command can be found in `<installation_folder>\designer\env\deploy.cmd`:

```
@echo off
if ""%1""=="""" goto echoSyntax
if ""%2""=="""" goto echoSyntax
if ""%3""=="""" goto echoSyntax
if ""%4""=="""" goto echoSyntax
if ""%5""=="""" goto echoSyntax
if ""%6""=="""" goto echoSyntax
```

```

IF ""%1""="" -logFile" " (
set LOG_FILE=%2
set JDBC_URL=%3
set APP_NAME=%4
set APP_VERSION=%5
set META_DIR=%6
set CREATE_APP=%7
set GENERATE_WSDL=%8
) ELSE (
set JDBC_URL=%1
set APP_NAME=%2
set APP_VERSION=%3
set META_DIR=%4
set CREATE_APP=%5
set GENERATE_WSDL=%6
)

set CWOC_CLASSPATH=C:\6\264
set CLASSPATH=%CWOC_CLASSPATH%\designer\Designer.jar;^
%CWOC_CLASSPATH%\lib\oracle\ojdbc6.jar;^
%CWOC_CLASSPATH%\lib\oracle\orai18n.jar;^
%CWOC_CLASSPATH%\lib\oracle\orai18n-mapping.jar;^
%CWOC_CLASSPATH%\lib\ilog\sdworkflowmodeler.deployed.jar;^
%CWOC_CLASSPATH%\lib\jdic\JDICplus.jar;^
%CWOC_CLASSPATH%\lib\axis2\addressing-1.41.mar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-1.5.mar;^
%CWOC_CLASSPATH%\lib\axis2\mex-1.4.1.mar;^
%CWOC_CLASSPATH%\lib\axis2\wstx-asl-3.2.4.jar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-core-1.5.jar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-policy-1.5.jar;

set JAVA_OPTS=-Xms1024m -Xmx1024m -XX:MaxPermSize=128m

IF ""%1""="" -logFile" " (
"C:\java\jdk1.6.0_16\bin\java" %JAVA_OPTS% -cp "%CLASSPATH%" com.conceptwave.servicedesigner.ServiceDesigner -
deploy -logFile %LOG_FILE% -jdbc=%JDBC_URL% -app_name=%APP_NAME% -app_version=%APP_VERSION% -
create_app=%CREATE_APP% -generate_wsdl=%GENERATE_WSDL% %META_DIR%
) ELSE (
"C:\java\jdk1.6.0_16\bin\java" %JAVA_OPTS% -cp "%CLASSPATH%" com.conceptwave.servicedesigner.ServiceDesigner -
deploy -jdbc=%JDBC_URL% -app_name=%APP_NAME% -app_version=%APP_VERSION% -create_app=%CREATE_APP% -
generate_wsdl=%GENERATE_WSDL% %META_DIR%
)

goto end
:echoSyntax
echo -----
echo Syntax:
echo deploy [-LogFile logfile] JDBC_URL application_name application_version metadata_path
create_database_application(true/false)
echo JDBC_URL could be user@host:port:sid,user@host:port/service_name or any jdbc url
echo Example:
echo deploy "cw@localhost:1521:orcl ON Version1.0 c:\metadata_path"
echo deploy
"cw@oracle.jdbc.driver.OracleDriver;jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on) (ADDRESS=(PROTOCOL=TCP) (HOST=
ON Version1.0 c:\metadata_path"
echo (refer to documentation for details.)
echo -----
:end

```

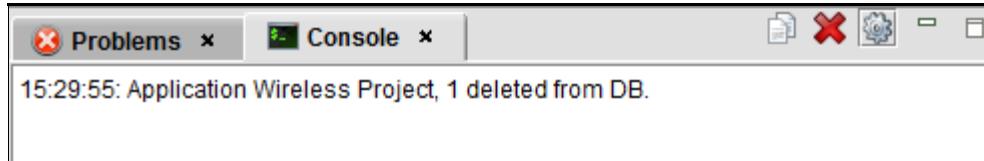
## Delete Application

---

You can delete a database deployed application in Velocity Studio.

To delete a deployed application from the database, follow these steps:

1. Click **Runtime > Delete Application** from the menu bar.
2. The Deleting application from database dialog appears.
3. Upon successful deletion, the Console pane displays a summary of the application.



## Difference Database Application

---

Velocity Studio provides a way to find the difference in your current application and the database deployed application. To find the differences, follow these steps:

1. Click **Runtime > Diff DB Application** from the menu bar.
2. The Save Quality Report dialog appears prompting you to save the HTML file. The default name of the file is AVM Metadata-MD\_Database\_Diff with the extension of date you run this command. You can change the name of the file.
3. Review the file to find the differences.

## Open Database Application

---

You can view an existing database deployed application in Velocity Studio. When opening a database deployed application, Velocity Studio closes your current project and loads the metadata from the database.

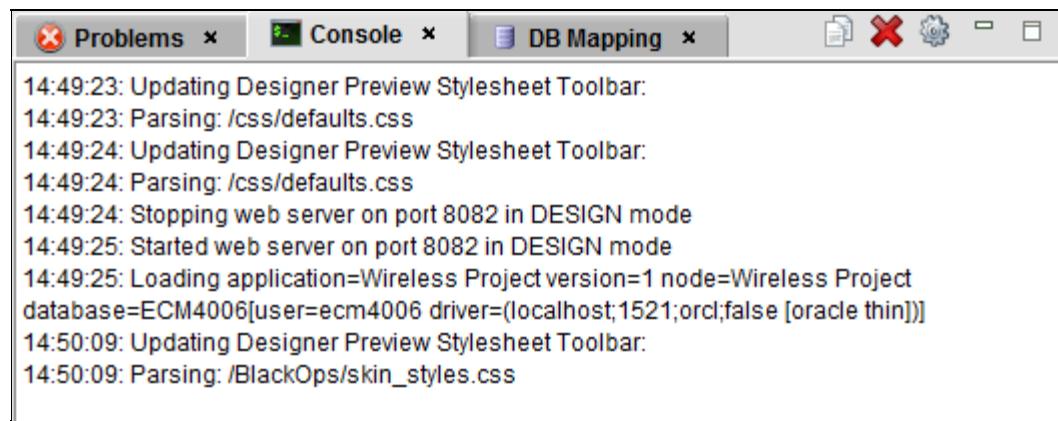
You can then view and run the metadata from Velocity Studio. Any local edits to this application are not saved, but are compiled at runtime system

To open a deployed application from the database, follow these steps:

1. Click **Runtime > Open DB App** from the menu bar.
2. The Opening Database Application dialog appears.

**Note:** An Error Migrating Metadata dialog appears if the application is not deployed to database.

3. Once the metadata is migrated from the database, the Console pane displays a summary of the application.



```
14:49:23: Updating Designer Preview Stylesheet Toolbar:  
14:49:23: Parsing: /css/defaults.css  
14:49:24: Updating Designer Preview Stylesheet Toolbar:  
14:49:24: Parsing: /css/defaults.css  
14:49:24: Stopping web server on port 8082 in DESIGN mode  
14:49:25: Started web server on port 8082 in DESIGN mode  
14:49:25: Loading application=Wireless Project version=1 node=Wireless Project  
database=ECM4006[user=ecm4006 driver=(localhost;1521;orcl;false [oracle thin])]  
14:50:09: Updating Designer Preview Stylesheet Toolbar:  
14:50:09: Parsing: /BlackOps/skin_styles.css
```

**Note:** The **Save** button is not available when viewing the application from database.

## Build Deployment WAR

The project's metadata can be built to a Web Application Archive (WAR) file for J2EE deployment within Velocity Studio by clicking **Runtime > Build Deployment WAR** from the menu bar. This action is also commonly referred as the **packaging** step in the [deployment procedure](#), where the WAR file (file with .war extension) is created for the export. A [command-line equivalent of packaging metadata](#) is also available outside of Velocity Studio.

**Note:** Running the runPackage command from a directory other than designer\env will fail.

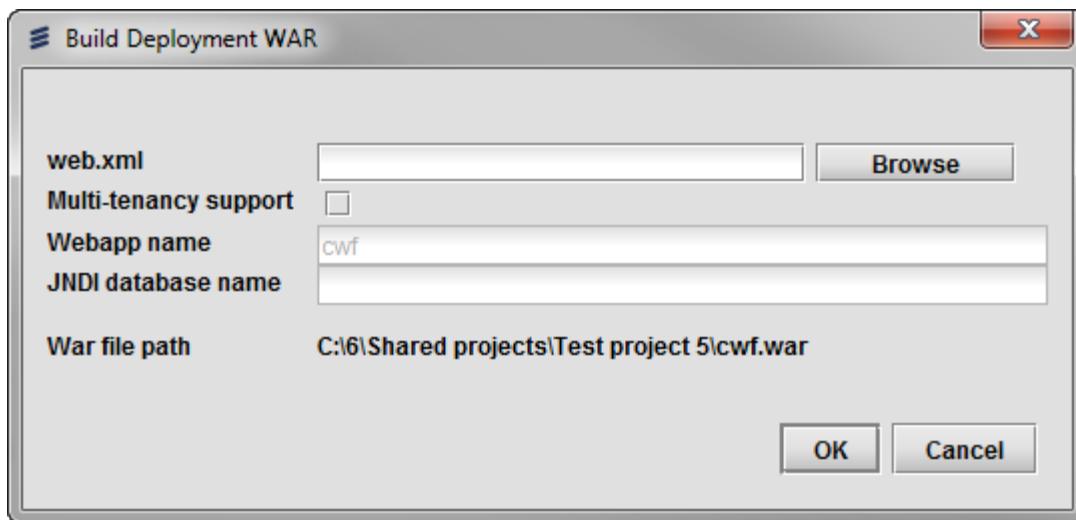
### Menu Command

The **Runtime > Build Deployment WAR** menu option is only enabled when a project has been opened and no [validation errors](#) exist in the metadata. Otherwise, the menu command is disabled (that is, greyed out). All errors must be fixed in the metadata before being able to build the WAR file. This menu command may create and update metadata. During packaging, the WAR file is created in the metadata folder. The size of the metadata is generally not smaller than 70 megabytes. Thus, it is generally *not* recommended to check this file into a version control system due to its large size.

This menu command is only necessary as part of the deployment procedure. It is not recommended to use this command for day-to-day application development and testing, due to its potential to create (potentially unnecessary) Process Revisions. Instead, use **Runtime > Run** to run application metadata within Velocity Studio without deploying any WAR file. If any Process Metadata was changed from its latest [Process Revision](#), a new Process Revision is created in addition to updating the revision number of the Process Metadata. If this happens, it is important to immediately check in these metadata changes into source control system. Otherwise, further development within the process metadata becomes out of sync with the Process Revisions.

### Creating the WAR file

Once you have selected the Build Deployment WAR option from the Runtime menu, the system will prompt for various parameters:



The following table describes the fields:

Field	Description
web.xml	Click the <b>Browse</b> button and select your web.xml that you want to be built into your WAR file. Both the web.xml and weblogic.xml files can be found in the <installation_directory>\conf folder of your Velocity Studio installation. Edit the web.xml file to suit your needs.
Multi-tenancy support	Check this box to create a WAR file that supports multi-tenancy. Multi-tenancy provides the ability to deploy multiple WAR files instances within the same container. For more details on this feature, refer to the Multi-Tenancy Packaging section in the <a href="#">Packaging in Command Line Interface</a> document.

<b>Webapp name</b>	This configures the application URL. For example, <code>http://localhost:8080/cwf</code> used when the user enters <code>cwf</code> in this parameter. The value specified here is also used to define the WAR file name (for example, <code>cwf.war</code> )
<b>JNDI database name</b>	This is the JNDI database name as defined in the <a href="#"><code>oracle-ds.xml</code> code</a> file.
<b>War file path</b>	This is the file location where the WAR file will be created and stored.

While the WAR file is being built, the Creating deployment WAR file popup appears, and then disappears when the process has completed.

The created WAR file created has `<webappname>.war` as its filename and is located in root folder of the project directory.

#### License

The packaging command, which can be found in `<installation_directory>\designer\env`, must be accompanied by a valid license file.

## Build Library JAR

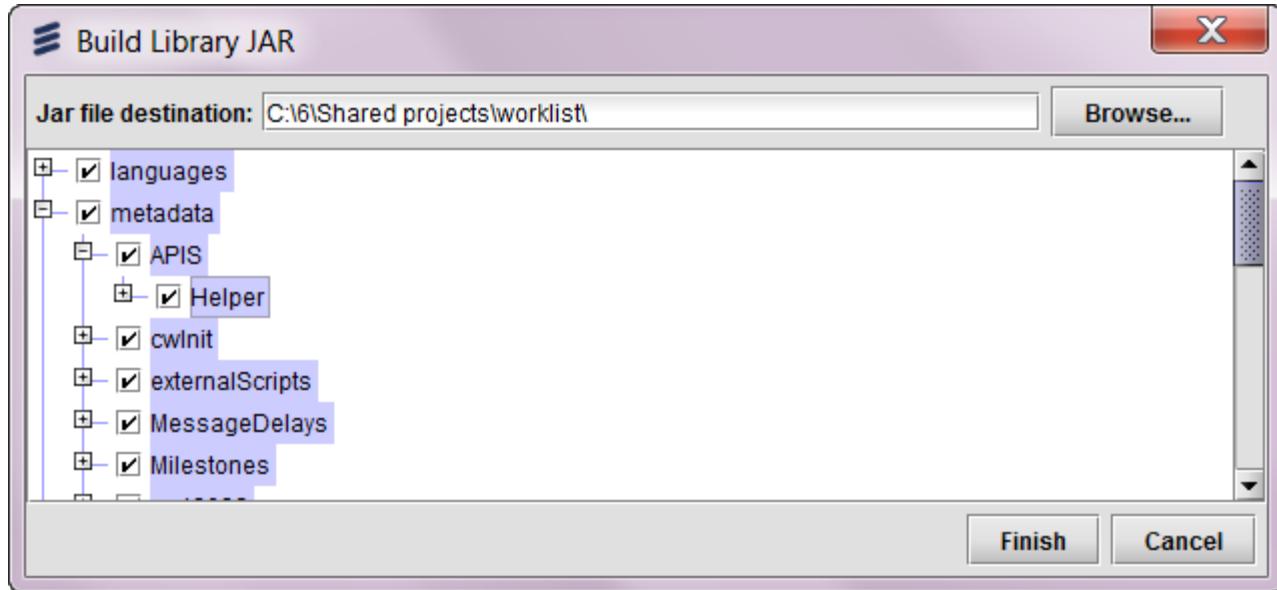
The project's application metadata can be exported as a Java Archive (JAR) file within Velocity Studio with menu **Runtime > Build Library JAR**. The JAR file can then be [imported](#) into another project as a library to be used by the project in Velocity Studio.

Use this command to create your own libraries to aid the distribution and organization of your application. For example, external interface integration based on technical contracts (such as WSDL) can be consumed and implemented in Velocity Studio as a project, and then use this command to create the library. Such a library may be imported into your application, with the benefit that your application metadata being focused on implementing the core application. See [Managing Metadata](#) for background information.

**Note:** This menu command is only enabled when a project has been opened and no [validation errors](#) exist in the metadata. Otherwise, the menu command is disabled (that is, greyed out). You must fix all errors in the metadata before being able to build the JAR file.

To build a library JAR file, follow these steps:

1. From the menu bar, click **Runtime > Build Library JAR**. to launch the Build Library JAR dialog.
2. For the **JAR file destination** field, specify the folder in which to save the output JAR file by clicking the **Browse** button. If you have set the **Export Template JAR Folder** field in your [Preferences](#), this location serves as the default folder for the generated JAR file.



3. By default, all project metadata folders are selected to be included in creating the JAR file. You can select or deselect project folders and their respective sub-folders. Selecting a folder highlights its folder label, indicating that it has been selected for inclusion in the generated JAR file.

**Note:** When you select a folder, that folder and its sub-folders are selected. Similarly, deselecting a folder means that its sub-folders are also deselected.

4. Click the **Finish** button to create the JAR file. The Creating library JAR file dialog appears and then disappears when finished.
5. The JAR file has `<internal_name>.jar` as its filename, where `internal_name` is the **Internal Name** field of your metadata project. You can rename the JAR file without impacting its integrity.

**Note:** The metadata files contained within the JAR file are encrypted for the purpose of protecting intellectual property.

## Build Library JAR through the Command Line

You can generate the Build Library output using a command line:

- In Windows, use `buildLibrary.cmd`
- In Unix or Linux, use `buildLibrary.sh`

The following describes the syntax of the command-line interface and the available arguments. A shortcut command can be found in <installation\_folder>\designer\env\buildLibrary.cmd:

```
@echo off
if ""%1""=="""" goto echoSyntax

set CWOC_CLASSPATH=C:\6\264
%CWOC_CLASSPATH%\lib\oracle\ojdbc6.jar;^
%CWOC_CLASSPATH%\lib\oracle\orai18n.jar;^
%CWOC_CLASSPATH%\lib\oracle\orai18n-mapping.jar;^
%CWOC_CLASSPATH%\lib\ilog\sdworkflowmodeler.deployed.jar;^
%CWOC_CLASSPATH%\lib\jdic\JDICplus.jar;^
%CWOC_CLASSPATH%\lib\axis2\addressing-1.41.mar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-1.5.mar;^
%CWOC_CLASSPATH%\lib\axis2\mex-1.4.1.mar;^
%CWOC_CLASSPATH%\lib\axis2\wstx-asl-3.2.4.jar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-core-1.5.jar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-policy-1.5.jar;

set JAVA_OPTS=-Xms1024m -Xmx1024m -XX:MaxPermSize=128m -Djava.awt.headless=true

"C:\java\jdk1.6.0_16\bin\java" %JAVA_OPTS% -cp "%CLASSPATH%"
com.conceptwave.servicedesigner.ServiceDesigner -buildlibrary %1 %2

goto end
:echoSyntax
echo -----
echo Syntax: buildLibrary [-logFile logFile] metadata_path [library name] [S stop with error]
echo Example: buildLibrary c:\metadata_path
echo buildLibrary c:\metadata_path catalog s
echo -----
:end
```

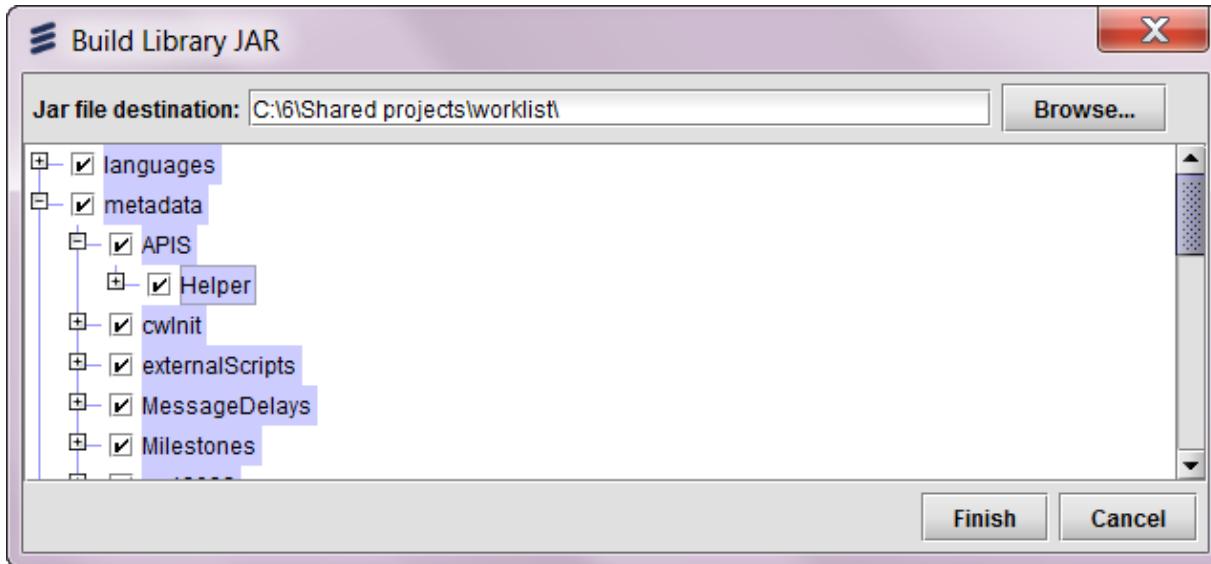
## Build Deployment JAR

In Velocity Studio, a project's application metadata can be exported as a Java™ Archive (JAR) file. The JAR file can then be [imported](#) into another project. This deployment JAR file contains all of the libraries included in the metadata.

**Note:** This menu command is only enabled when a project is opened and no [validation errors](#) exist in the metadata. Otherwise, the menu command is disabled (that is, greyed out). You must fix all the errors in the metadata to build the deployment JAR file.

To build a deployment JAR file, follow these steps:

1. From the menu bar, click **Runtime > Build Deployment JAR** menu.
2. The Select JAR Output Directory dialog appears. Specify the folder where you want to save the output JAR file, and then click the **Open** button.



3. The Creating library JAR file dialog appears, and then disappears when finished.
4. The deployment JAR file is saved in the specified directory with the project's internal application name.

## Build Deployment JAR File through Command Line

You can build the deployment JAR file through the [command line](#). The following describes the syntax of the command-line interface and the available arguments. A shortcut command can be found in `<installation_folder>\designer\env\buildDeployment.cmd`:

```
@echo off
if ""%1""=="""" goto echoSyntax

set CWOC_CLASSPATH=C:\6\264
set CLASSPATH=%CWOC_CLASSPATH%\designer\Designer.jar;^
%CWOC_CLASSPATH%\lib\oracle\ojdbc6.jar;^
%CWOC_CLASSPATH%\lib\oracle\orai18n.jar;^
%CWOC_CLASSPATH%\lib\oracle\orai18n-mapping.jar;^
%CWOC_CLASSPATH%\lib\ilog\sdworkflowmodeler.deployed.jar;^
%CWOC_CLASSPATH%\lib\jdic\JDICplus.jar;^
%CWOC_CLASSPATH%\lib\axis2\addressing-1.41.mar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-1.5.mar;^
%CWOC_CLASSPATH%\lib\axis2\mex-1.4.1.mar;^
%CWOC_CLASSPATH%\lib\axis2\wstx-asl-3.2.4.jar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-core-1.5.jar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-policy-1.5.jar;

set JAVA_OPTS=-Xms1024m -Xmx1024m -XX:MaxPermSize=128m

"C:\java\jdk1.6.0_16\bin\java" %JAVA_OPTS% -cp "%CLASSPATH%" com.conceptwave.servicedesigner.ServiceDesigner -buildDeployment %*
```

```
goto end
:echoSyntax
echo -----
echo Syntax: buildDeployment [-logFile logfile] metadata_path [output dir]
echo Example: buildDeployment c:\metadata_path
echo           buildDeployment c:\metadata_path c:\
echo -----
:end
```

## Analyze Processes

You can use the **Analyze Processes** tool to compare local process metadata and the deployed process metadata. It is highly recommended that you perform the comparison prior to deploying metadata, especially in production, to ensure metadata compatibility and to avoid runtime process exceptions. The tool compares the processes in the local directory and the deployed metadata and looks for processes with modified metadata on the same revision number. Having modified processes with the same revision number is a potential runtime exception.

Based on the comparison it performs, if the **Analyze Processes** tool indicates that the metadata is compatible, the metadata can be deployed. Otherwise, it is the user's responsibility to analyze each non-compatible process case by case and decide whether a new revision is needed or to deploy in the current state.

To use the **Analyze Processes** tool, do the following:

1. Click **Runtime > Analyze Processes** from the menu bar.
2. In the Analyze Processes dialog, choose the **Full Check** option or the **Quick Check** option.
  - o **Full Check:** this is the default and the recommended mode. It provides the most reliable comparison as it involves each process component. The **Full Check** mode compares the following:
    - Process properties, such as type of process, priority, process document reference, "synchronize" flag, etc.
    - Process workflow, i.e. the activity tree
    - Local subflows belonging to the process
    - Activities properties, such as participant references, operations, alerts, etc.
    - Process and activities script: name, parameters, return types, scripts, etc.
  - o **Quick Check:** this mode compares processes by file size and timestamp. The benefit of this mode is that it is quicker to use than the **Full Check** mode. However, this mode has the following limitations:
    - File size can remain the same even if the process was modified. For example, if the process subflow revision reference is changed from 4 to 5, the process size remains the same yet the workflow is modified and can cause runtime exceptions.
    - Timestamp comparison is not completely reliable as identical processes can have different timestamps. The metadata timestamp depends on when the metadata was checked out of the Revision Control System, such as SVN.
3. Choose whether or not to keep the **Open check result** checkbox in Analyze Processes dialog selected. This checkbox allows you to choose whether the system automatically opens the comparison results file once the comparison is done. By default, the checkbox is selected.
4. Click the **OK** button.
5. In the dialog that appears, specify the filename of the text (.txt) file that is to store the comparison results, and the location where the file is to be stored.
6. Click the **Save** button.

**Note:** When using this command, you are prompted to connect to the database, if the database connection is not established already.

## Analyze Processes Command-Line Option

You can also run the Analyze Process tool from the [command line](#). The following describes the syntax of the command-line interface and the available arguments. You can find the command in the `<installation_folder>\designer\analyseProcesses.cmd`:

```
@echo off
if "%1"==" goto echoSyntax
if "%2"==" goto echoSyntax
if "%3"==" goto echoSyntax
if "%4"==" goto echoSyntax
if "%5"==" goto echoSyntax
if "%6"==" goto echoSyntax
if "%7"==" goto echoSyntax
IF "%1"=="-logFile" (
set LOG_FILE=%2
set JDBC_URL=%3
set MODE=%4
set APP_NAME=%5
set APP_VERSION=%6
set NODE=%7
```

```

set META_DIR=%8
set FILE=%9
) ELSE (
set JDBC_URL=%1
set MODE=%2
set APP_NAME=%3
set APP_VERSION=%4
set NODE=%5
set META_DIR=%6
set FILE=%7
)

set CWOC_CLASSPATH=C:\6\411
set CLASSPATH=%CWOC_CLASSPATH%\designer\Designer.jar;^
%CWOC_CLASSPATH%\lib\oracle\ojdbc6.jar;^
%CWOC_CLASSPATH%\lib\oracle\orai18n.jar;^
%CWOC_CLASSPATH%\lib\oracle\orai18n-mapping.jar;^
%CWOC_CLASSPATH%\lib\ilog\sdworkflowmodeler.deployed.jar;^
%CWOC_CLASSPATH%\lib\jdic\JDICplus.jar;^
%CWOC_CLASSPATH%\lib\axis2\addressing-1.41.mar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-1.5.mar;^
%CWOC_CLASSPATH%\lib\axis2\mex-1.4.1.mar;^
%CWOC_CLASSPATH%\lib\axis2\wstx-asl-3.2.4.jar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-core-1.5.jar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-policy-1.5.jar;

set JAVA_OPTS=-Xms1024m -Xmx1024m -XX:MaxPermSize=128m -
Dcom.conceptwave.licenseDir="%CWOC_CLASSPATH%\designer\env -
Dcom.conceptwave.modules.path="%CWOC_CLASSPATH%\modules
IF ""%"=="LogFile" " (
"C:\Program Files\Java\jdk1.7.0_45\bin\java" %JAVA_OPTS% -cp "%CLASSPATH%"
com.conceptwave.servicedesigner.ServiceDesigner -analyze -LogFile %LOG_FILE% -jdbc=%JDBC_URL% -mode=%MODE% -
app_name=%APP_NAME% -app_version=%APP_VERSION% -config_node=%NODE% -metadata_dir=%META_DIR% -file=%FILE%
) ELSE (
"C:\Program Files\Java\jdk1.7.0_45\bin\java" %JAVA_OPTS% -cp "%CLASSPATH%"
com.conceptwave.servicedesigner.ServiceDesigner -analyze -jdbc=%JDBC_URL% -mode=%MODE% -app_name=%APP_NAME% -
app_version=%APP_VERSION% -config_node=%NODE% -metadata_dir=%META_DIR% -file=%FILE%
)
goto end
:echoSyntax
echo -----
echo Syntax:
echo analyseProcesses [-LogFile logfile] JDBC_URL mode(F/Q) application_name application_version config_node
metadata_dir output_file
echo JDBC_URL could be user@host:port:sid,user@host:port/service_name or any jdbc url
echo Example:
echo analyseProcesses cw@localhost:1521:orcl F ON V1.0 PE c:\metadata\ON c:\result.txt
echo analyseProcesses
"cw@oracle.jdbc.driver.OracleDriver;jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on))(ADDRESS=(PROTOCOL=TCP)(HOST=
Q ON V1.0 PE c:\metadata\ON c:\result.txt
echo (refer to documentation for details.)
echo -----
:end

```

**Note:** The `-mode` parameter is optional. If it is not specified, full check mode is used. Otherwise, you can specify the following values for this field:

- `F` - represents full mode
- `Q` - represents quick mode

## Clean Non-Running Revisions

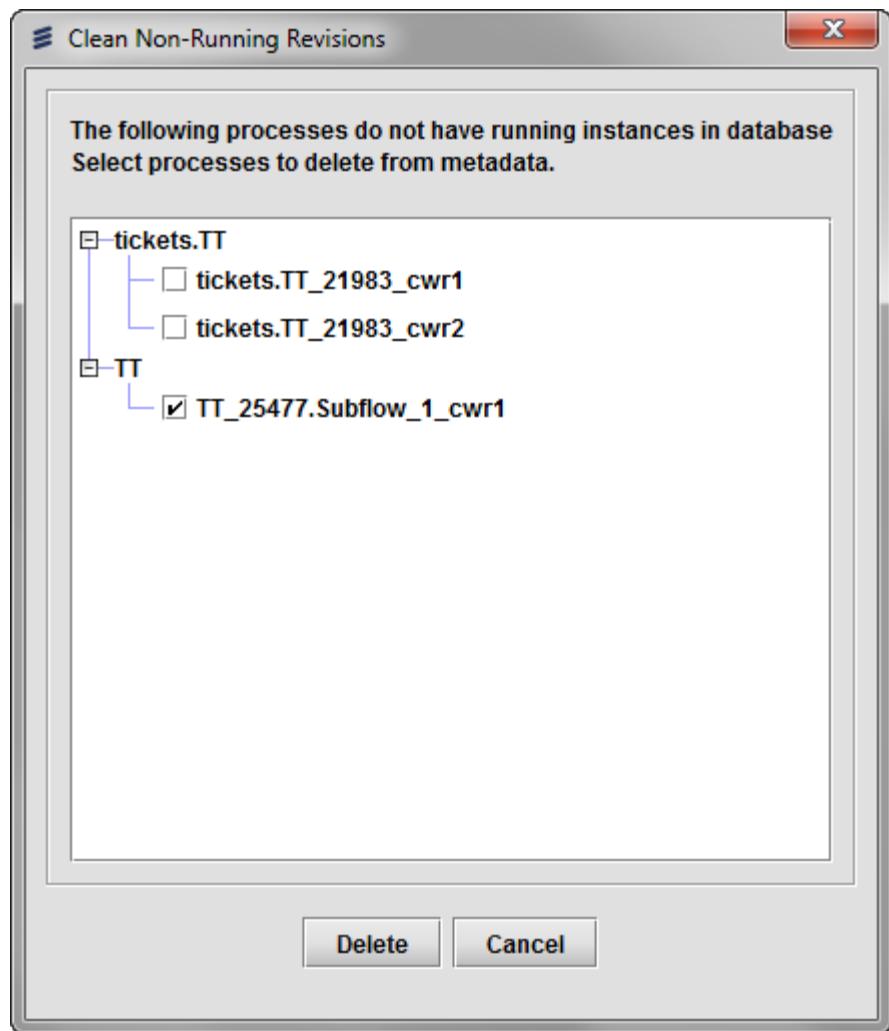
---

The **Clean Non-Running Revisions** command identifies the process revisions that have no running instances or are not used by running instances in the current database schema, and allows you to delete these processes. To use the command, do the following:

1. Click **Runtime > Clean Non-Running Revisions** from the menu bar.

**Note:** When using this command, you are prompted to connect to the database, if the database connection is not established already.

2. When the database connection is established, the command runs and the process revisions that do not have running instances or are not used by running instances in the current database schema are displayed as a list. An empty list is returned if there are no non-running process revisions. Revisions are collected and displayed under their parent processes in a tree structure. You cannot select and delete top-level processes. You can only collapse or expand the top-level process nodes to view the non-running revisions listed under those nodes. The criteria used to determine which non-running revisions are listed is as follows:
  - o Excludes all top-level processes: user, subflow, and global
  - o Excludes all process revisions with running instances in the database
  - o Traverse each process or revision from the two previous list items, and exclude all subflow revisions, nested on any level, such as a subflow within a subflow
  - o List contains all revisions of user processes and subflows in the current metadata that are not in any of the previous list items
  - o List does not contain duplicate entries and is ordered by full name
  - o List does not include revisions from templates (system as well as custom ones)



3. To clean the non-running revision from the metadata, select the checkbox for the revisions in the list, and click the **Delete** button. A confirmation dialog appears. To delete the revisions, click the **Yes** button.
4. A message dialog appears that displays the number of deleted revisions from metadata. Click the **OK** button to close the dialog.

**Note:** This clean-up process is irreversible. The revisions are deleted from the metadata tree and also from database.

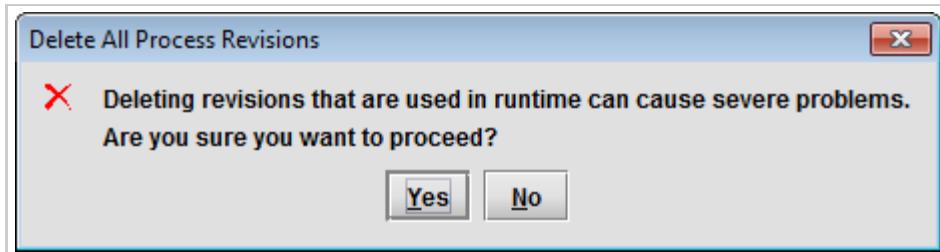
## Clean All Process Revisions

All Process Revisions of all Process metadata in the project can be removed using the **Runtime > Clean All Process Revisions** menu command.

### Important Note

Do not use this command if there are process instances in target deployment (for example, production systems) that use any of these process revisions and yet to be completed! These process instances will no longer be able to be processed by the Process Engine. Only use this command when you are sure that it is safe to do so.

Upon clicking the command, the following prompt is displayed to warn you of the potential consequence of incorrect use:



Click the **Yes** button to proceed, which purges all Process Revisions of all Processes. All the Process Revision metadata files in the application metadata of the project are deleted. The Revision Number in the Process metadata remains unchanged; they are not reset.

Otherwise, click the **No** button to cancel the command.

## Velocity Studio - Database Menu

---

The Velocity Studio **Database** menu contains the following menu items and brief descriptions of each menu item:

**Connect**

Connects to your database. This option is visible only when the database connection is not established.

**Disconnect**

Disconnects your connection to the database. This option is visible only when the database connection is established.

**Check DB Mapping**

Verifies that the currently open metadata has valid database mappings.

**Upgrade System**

Generates an SQL file that changes the database to fit the current project's metadata requirements.

**Map Documents to DB**

Maps the database Documents that have a Database Schema and are not yet mapped.

**Import Procedures**

Allows you to create or update Documents needed to call a database stored procedure in runtime.

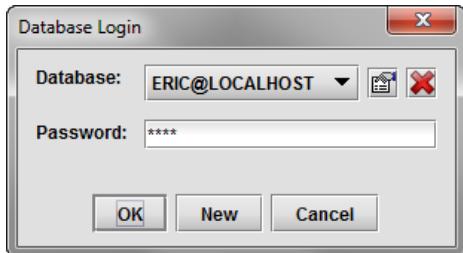
**Import DB Schema**

Creates a metadata Document from a database table.

## Connect to the Database

In Velocity Studio, you can connect your current project to the database by clicking **Database > Connect** from the menu bar. Once connected, other actions that require a database connection no longer prompt for database login in Velocity Studio.

The Database Login dialog appears to prompt for connecting to a database.



If your project does not have a previously configured database connection, there is no connection available when you click the **Database** field's drop-down menu. Click the **New** button to create settings for a new database connection. For an existing and selected database connection in this dialog, you can click the **Edit** icon or **Remove** icon to edit the properties or remove the connection settings, respectively.

The database connection settings are persisted in the project directory (in [.settings in directory root](#)), and are remembered even after you close and re-open the project in Velocity Studio.

### Notes:

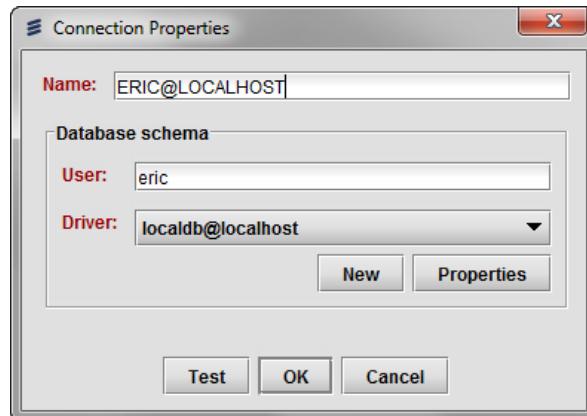
- In a team development environment, this file may be in revision control. As a result, changes to database connections *could be* propagated to other team members.
- This database connection is the *bootstrap* connection to load the project's configuration that is in the database. This bootstrap connection is not related to [physical connections](#) and [logical connections](#) configured within the Configuration application (even though they can, and likely, share the same parameters), which is the database connection for running metadata (for example, using database tables that corresponds to Document's database mapping).

Once you have set up the database connection, ensure that you have entered the **Password**, and then click the **OK** button to connect to database. The Database Login dialog contains the following fields and buttons:

Field	Description
<b>Database</b>	Click the drop-down menu to see a list of database connections that has been previously created in the project.
<b>Password</b>	Enter the login password for the selected database.
<b>Properties</b>	Click this button to open the <a href="#">Connection Properties dialog</a> , which displays the selected database connection's properties in the <b>Database</b> field. You can update these properties, if necessary.
<b>Remove</b>	Click this button to remove the selected database connection in the <b>Database</b> field from <a href="#">.settings</a> file in the project directory.
<b>OK</b>	Click this button to connect to the database according to the database connection setting and password.
<b>New</b>	Click this button to open the <a href="#">Connection Properties dialog</a> to add a new database connection setting. After it has connected successfully, the new entry is added to <a href="#">.settings</a> file and shown in the <b>Database</b> drop-down field.
<b>Cancel</b>	Click this button to cancel the operation.

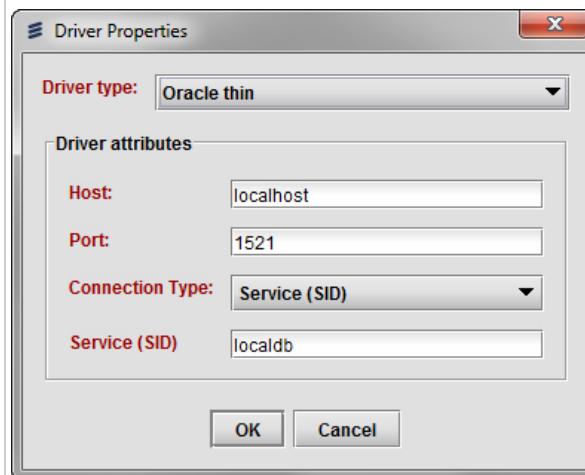
### Connection Properties

The Connection Properties dialog opens if you either click the **New** or **Properties** button from the Database Login dialog.



The Connection Properties dialog contains the following fields and buttons:

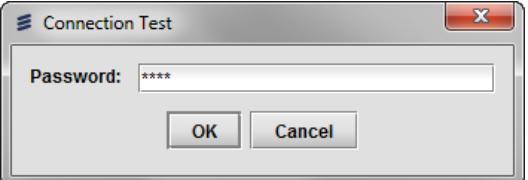
Field	Description
Name	This field is mandatory. It denotes the user-defined name that displays in the Database Login dialog's <b>Database</b> drop-down field.
User	This field is mandatory. It represents the database user name.
Driver	This field is mandatory. A combo box containing a list of available database drivers.
New	Click this button to opens the Driver Properties dialog to add a new database driver. The new entry appears in the <b>Driver</b> field's drop-down menu.
Properties	Click this button to open the Driver Properties dialog to edit the selected database driver's properties.



The following fields appears:

Field	Description
Driver Type	This field is mandatory. Click this field's drop-down menu to view a list of database driver types that you can use to establish a database connection. The following selections are available: <ul style="list-style-type: none"> <li>• Oracle Thin: Oracle database thin driver (pure Java). It is recommended that you select this option for all Oracle database connections.</li> <li>• Oracle OCI: Oracle database OCI driver.</li> <li>• JDBC - Any: Generic JDBC driver.</li> </ul>
Driver Attributes	This field is mandatory. This section contains a set of fields that change depending on the driver selection in the <b>Driver type</b> field. The attributes for each <b>Driver type</b> are as follows: <ul style="list-style-type: none"> <li>◦ Oracle Thin: <ul style="list-style-type: none"> <li>▪ Host (for example, localhost)</li> <li>▪ Port (1521 is the default for an Oracle database)</li> <li>▪ Service (for example, localdb)</li> </ul> </li> <li>◦ Oracle OCI:</li> </ul>

- TNS Name
- JDBC - Any:
  - Class Name (for example, oracle.jdbc.driver.OracleDriver)
  - Driver URL (for example, jdbc:oracle:thin:@(DESCRIPTION=(enable=broken)(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))(CONNECT\_DATA=(SERVICE\_NAME=localdb)))

	Test	Click this button to test the database connection. The Connection Test dialog opens, prompting you to enter the database user password.
		
		Click the <b>OK</b> button to attempt to connect to the database using the defined properties.
OK		Click this button to save the new or edited connection properties, and return to the Database Login dialog.
Cancel		Click this button to cancel the operation.

## Disconnect from the Database

---

If Velocity Studio has been [connected with a database connection](#), clicking **Database > Disconnect** is available from the menu bar, which disconnects the database connection.

Once you have terminated the database connection, the **Database > Connect** option becomes available again from the menu bar for re-establishing the database connection.

## Check Database Mapping

If there are no [validation errors](#), you can verify that the currently open metadata has valid database mappings by clicking **Database > Check DB Mapping** from the menu bar. If there are any invalid mappings, the [DB Mapping Pane](#) will open displaying the list of errors and/or warning due to database mappings. Expand the error list and warning list using the expand icon, and double-click an item beneath it to jump to the Document in Navigation Pane and Detail Pane.

The screenshot shows the 'DB Mapping' pane with three tabs: 'Problems', 'Console', and 'DB Mapping'. The 'Problems' tab is selected, displaying a list of errors and warnings. The 'Description' column lists specific mapping issues, and the 'Object' column lists the corresponding database objects. A detailed description of the errors follows:

Description	Object
Errors(1 items) DE0609: Table "CWPUSRPRTCPNTS" does not exist	CWPUSRPRTCPNTS (Participant Users)
Warnings(20 items) DW0515: Database column is larger than field "Event External Cc" DW0513: Field "Group Name" is not mapped to database DW0513: Field "Name" is not mapped to database DW0513: Field "Active" is not mapped to database DW0513: Field "Manager" is not mapped to database DW0513: Field "YEAR" is not mapped to database DW0513: Field "CALENDAR" is not mapped to database DW0513: Field "DAY" is not mapped to database	mappingForProductProperties (Mapping for product properties) user_group (User or Group) user_group (User or Group) user_group (User or Group) user_group (User or Group) calendarYearByType (Calendar Year By Type) calendarYearByType (Calendar Year By Type) calendarYearByType (Calendar Year By Type)

## Upgrade System

If there are no [validation errors](#), you can generate a SQL file that contains statements to change the database to fit the current product release requirements by selecting the **Upgrade System** command from the **Database** menu. It is necessary to check whether the system upgrade is required every time a new release of the product has been installed. The runtime AVM will not start if the database has not been upgraded.

Once you select the **Upgrade System** command, the **Upgrade SQL** dialog opens where you can specify the directory and the name of the generated file. Click the **Save** button to create the file.

If there are no system upgrades required, a dialog will inform you. Otherwise, once the SQL file is generated, connect to the appropriate database (use SQLPlus or another database utility) and run the SQL.

### Important note:

The generated SQL file **must** be examined and changed when needed by the database administrator. The upgrade command generates only the essential parts of the DDL commands, and not indexes, constraints, triggers or any other advanced DDL constructs. The Velocity Studio is not a database design tool and cannot be used as such. Running the SQL file generated by Velocity Studio without first checking its contents may damage your database. Once the upgrade scripts have been run, the database administrator will need to review the procedures, triggers and views, and recompile them, if required.

By default, this command generates one SQL file. To have this command generate separate SQL script files for each logical schema, uncheck the [Generate upgrade SQL in one output file](#) setting in Velocity Studio's Preferences page.

**Note:** When you add a new JAR file in the [Library](#) and upgrade the system, an invalid database error message may occur. You need to add a logical schema for the JAR file to take system upgrade. For example, if cwl\_dbqueue.jar is a part of the loaded libraries, a new DBQUEUE logical schema must be added in the [System Administration](#) application.

## Upgrade system through the command line

You can generate the Update System output using a command line:

- In Windows, use `generateUpgradeSQL.cmd`
- In Unix or Linux, use `generateUpgradeSQL.sh`

The following describes the syntax of the command-line interface and the available arguments. A shortcut command can be found in `<installation_folder>\designer\env\generateUpgradeSQL.cmd`, and shown as follows.

```
@echo off
if ""%1""=="""" goto echoSyntax
if ""%2""=="""" goto echoSyntax
if ""%3""=="""" goto echoSyntax
if ""%4""=="""" goto echoSyntax
if ""%5""=="""" goto echoSyntax
IF ""%1""=="LogFile" (
set LOG_FILE=%2
set JDBC_URL=%3
set APP_NAME=%4
set APP_VERSION=%5
set META_DIR=%6
set SQL_FILE=%7
set SQL_FILE_OPTION=%8
) ELSE (
set JDBC_URL=%1
set APP_NAME=%2
set APP_VERSION=%3
set META_DIR=%4
set SQL_FILE=%5
set SQL_FILE_OPTION=%6
)

set CWOC_CLASSPATH=C:\6\264
set CLASSPATH=%CWOC_CLASSPATH%\designer\Designer.jar;^
%CWOC_CLASSPATH%\lib\oracle\ojdbc6.jar;^
```

```

%CWOC_CLASSPATH%\lib\oracle\orai18n.jar;^
%CWOC_CLASSPATH%\lib\oracle\orai18n-mapping.jar;^
%CWOC_CLASSPATH%\lib\ilog\sdworkflowmodeler.deployed.jar;^
%CWOC_CLASSPATH%\lib\jdic\JDICplus.jar;^
%CWOC_CLASSPATH%\lib\axis2\addressing-1.41.mar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-1.5.mar;^
%CWOC_CLASSPATH%\lib\axis2\mex-1.4.1.mar;^
%CWOC_CLASSPATH%\lib\axis2\jaxb-xjc-2.1.6.jar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-core-1.5.jar;^
%CWOC_CLASSPATH%\lib\axis2\rampart-policy-1.5.jar;

set JAVA_OPTS=-Xms1024m -Xmx1024m -XX:MaxPermSize=128m
IF "%1"=="-logFile" (
"C:\java\jdk1.6.0_16\bin\java" %JAVA_OPTS% -cp "%CLASSPATH%" com.conceptwave.servicedesigner.ServiceDesigner -generateSql -logFile %LOG_FILE% -jdbc=%JDBC_URL% -app_name=%APP_NAME% -app_version=%APP_VERSION% -metadata_dir=%META_DIR% %SQL_FILE% %SQL_FILE_OPTION%
) ELSE (
"C:\java\jdk1.6.0_16\bin\java" %JAVA_OPTS% -cp "%CLASSPATH%" com.conceptwave.servicedesigner.ServiceDesigner -generateSql -jdbc=%JDBC_URL% -app_name=%APP_NAME% -app_version=%APP_VERSION% -metadata_dir=%META_DIR% %SQL_FILE% %SQL_FILE_OPTION%
)
goto end
:echoSyntax
echo -----
echo Syntax:
echo generateUpgradeSQL [-logFile logfile] JDBC_URL application_name application_version sql_file
echo output_in_one_file(true/false)
echo JDBC_URL could be user@host:port:sid,user@host:port/service_name or any jdbc url
echo Example:
echo generateUpgradeSQL cw@localhost:1521:orcl ON V1.0 c:\metadata\ON c:\upgrade.sql
echo generateUpgradeSQL cw@localhost:1521:orcl ON V1.0 c:\metadata\ON c:\upgrade.sql true
echo generateUpgradeSQL
"cw@oracle.jdbc.driver.OracleDriver;jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on)(ADDRESS=(PROTOCOL=TCP)(HOST=ON V1.0 c:\metadata\ON c:\upgrade.sql
echo (refer to documentation for details.)
echo -----
:end

```

## Map Documents to DB

---

You can map the database Documents which have a Database Schema and are not yet mapped by selecting the **Map Documents to DB** command from the **Database** menu. The function will generate the table name and column names for all Document leaves that are not explicitly marked as do not map and are not yet mapped to database.

Velocity Studio assumes that each Document will be mapped to one database table only. However, the runtime system allows the mapping of a Document to up to four database tables, providing that the primary keys of all the tables are identical by name and type. While this is rarely used, the performance of some applications may be improved by mapping some Documents to more than one table. If this is the case, you must create the tables upfront and map them manually to the Document leaves.

Once the mapping is done, the [Upgrade System](#) command should be used to generate the SQL file that contains the DDL needed to create the tables.

## Import Procedures

The **Import Procedures** command in the **Database** menu automatically generates the data type objects and Document objects needed to call a stored procedure. Velocity Studio allows the stored procedure to be called with atomic parameters only. Array and object parameters are not supported.

The **Import Procedures** command is a helper function that is used to reduce the development work. Instead of manually creating the corresponding data type objects and Document objects, the metadata developer can automatically generate them. When this command is used, certain assumptions are made that may not be valid in all cases. It is recommended that the developer carefully examines the generated objects and does appropriate changes where needed.

The stored procedure call is internally modelled using the WSDL model. Two Documents (input and output messages in the WSDL model) are generated if the stored procedure has output or in/out parameters and one Document if the stored procedure has only input parameters. Velocity Studio always attempts to use the existing metadata objects before generating new ones. In some cases, Velocity Studio may decide to change an existing metadata object to fit the properties required for the stored procedure call, but this will only be done if the modified object is backward compatible with the original one (that is, Velocity Studio changes existing data type objects only if their base type (integer, boolean, string, etc.) is the same, and the length and precision of the modified data type object can accommodate the length and precision of the original type.) For Documents, Velocity Studio adds new Document leaves if needed, but never removes existing leaves. If a metadata object with the same name already exists in the specified namespace and it cannot be reused or changed, Velocity Studio generates a new object that has the name of the existing object with a numeric suffix added for uniqueness.

Before Velocity Studio performs the generation, you must select a namespace in which the objects will be generated, changed, or reused. The required data type objects are created, updated, or reused first, one for each procedure input and output parameter. Then, Velocity Studio creates, or tries to reuse or modify existing Documents if you have specified the corresponding option.

For example, for the stored procedure SQL in the figure below, the **Data Type** objects, *WID* (**Base Type** String), *USERID* (**Base Type** String), and *FOUND* (**Base Type** Decimal), will be created first. Then the **Document** objects, *COMPLETEALERTInput* and *COMPLETEALERTOutput*, will be created. The *COMPLETEALERTInput* Document will have the leaves *WID* and *USERID* and the *COMPLETEALERTOutput* Document will have the one leaf, *FOUND*.

```
CREATE OR REPLACE PROCEDURE CWLC.COMPLETEALERT (wiId IN VARCHAR2, userId IN VARCHAR2, found OUT NUMBER)

AS

BEGIN

select CWDODCID into found from cwpWorkList where CWDODCID = wiId and USER_ID = userId for update;

CreateAlertArchive(wiId,userId);

delete from cwpWorkList where CWDODCID = wiId;

found := 1;

EXCEPTION

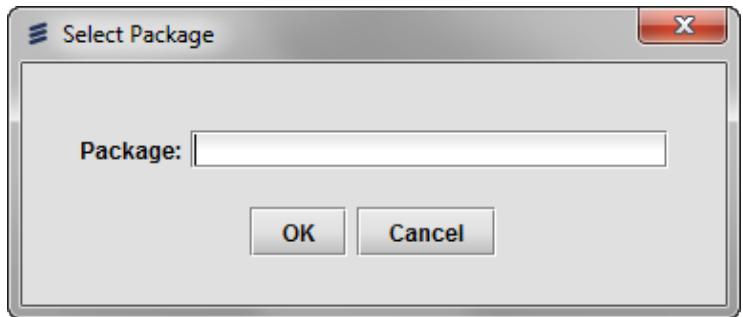
WHEN NO_DATA_FOUND THEN
found := 0;

END; |
```

### Notes:

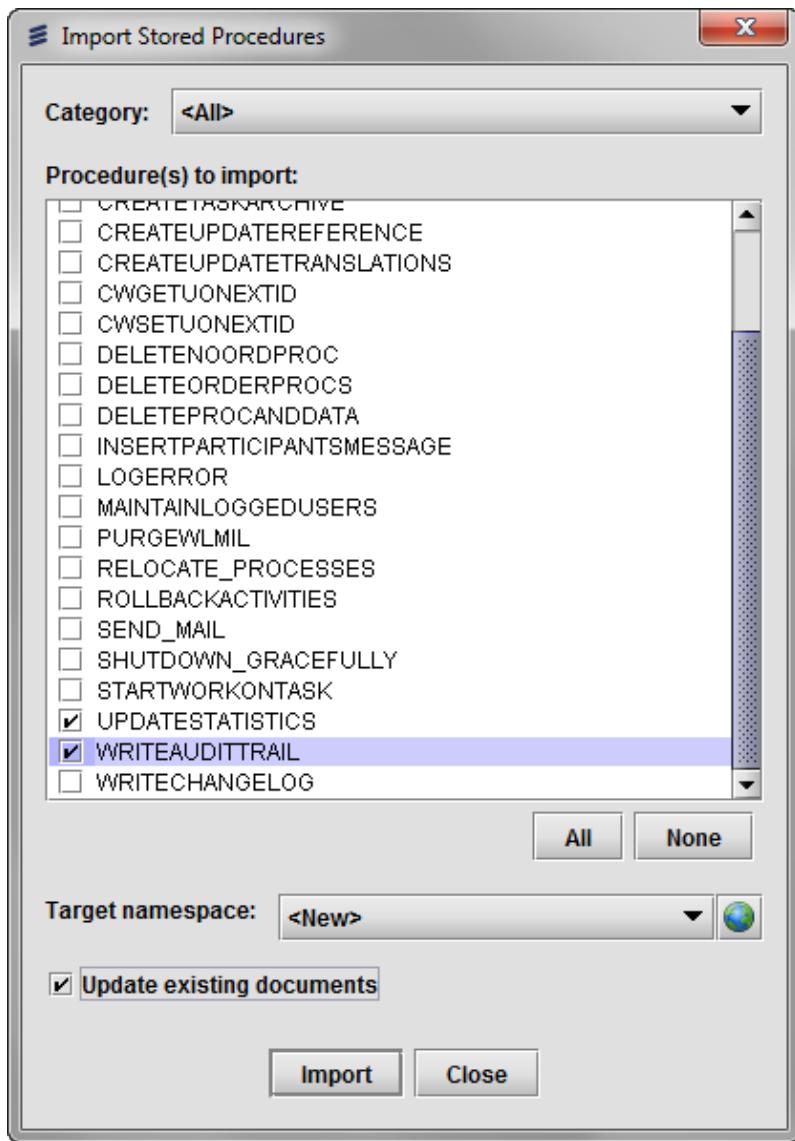
- The **Import Procedure** command is only used for Oracle databases.
- In the *cwpWorkList* table, FLAG=3 means that when a task is reassigned to a user, the original task is set to 3. The new task being created is set to 0.

Once you select the **Import Procedure** command, the [Database Login](#) dialog opens so that you can connect to the desired database where the stored procedures are located. The **Select Package** dialog opens, asking you to enter the package name that contains the stored procedures.



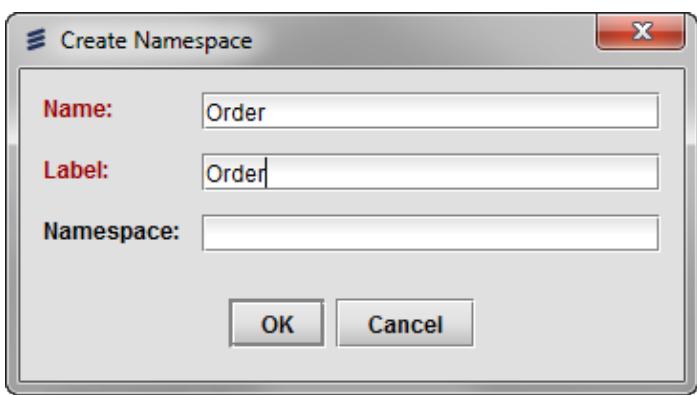
If the stored procedures are not part of a package then this field is left empty. Click the **OK** button.

The **Import Stored Procedures** dialog opens, where you can select the stored procedures to import as Documents.



After the desired stored procedures have been selected, click the **Import** button to start the import. The following table contains a description of each field.

Field	Description
<b>Category</b>	A list of categories in which the imported objects belong to. Currently not used for stored procedure import.
<b>Procedure(s) to import</b>	A list of available stored procedures that can be imported into the metadata.
<b>All</b>	When clicked, all the stored procedures in the <b>Procedure(s) to import</b> list will be selected for import.
<b>None</b>	When clicked, all the stored procedures in the <b>Procedure(s) to import</b> list will be deselected.

<b>Target namespace</b>	<p>The namespace in which the data type objects and Document objects, for the selected stored procedures, will be created/updated. If &lt;new&gt; is selected, then the <b>Create Namespace</b> dialog will open when the <b>Import</b> button is clicked.</p>  <p>You can define the <b>Name</b>, <b>Label</b> and <b>Namespace</b> properties of the new namespace. When the <b>OK</b> button is clicked, the new namespace will be created in the metadata along with the corresponding data type objects and Document objects for the selected stored procedure(s).</p>
<b>Update existing documents</b>	If checked, existing Document objects in the metadata will be updated to reflect changes in the selected stored procedures.

Once the import is completed, the **Import Stored Procedures** completed dialog opens, allowing you to either close the dialog by clicking the **Close Import Dialog** button or do another import by clicking the **Continue Import** button. If you click the **Continue Import** button, you are returned to the original **Import Stored Procedures** dialog.

## Import DB Schema

You can import database tables into the currently opened metadata as Documents by selecting the **DB Schema** command from the **Database** menu. The command works for Oracle databases and all other databases provided that the corresponding JDBC drivers can access the database schema metadata.

The **DB Schema** command is a helper function that is used to reduce the development work. Instead of manually creating the corresponding data type objects and Document objects, the metadata developer can automatically generate them. When this command is used, certain assumptions are made that may not be valid in all cases. It is recommended that the developer carefully examines the generated objects and does appropriate changes where needed.

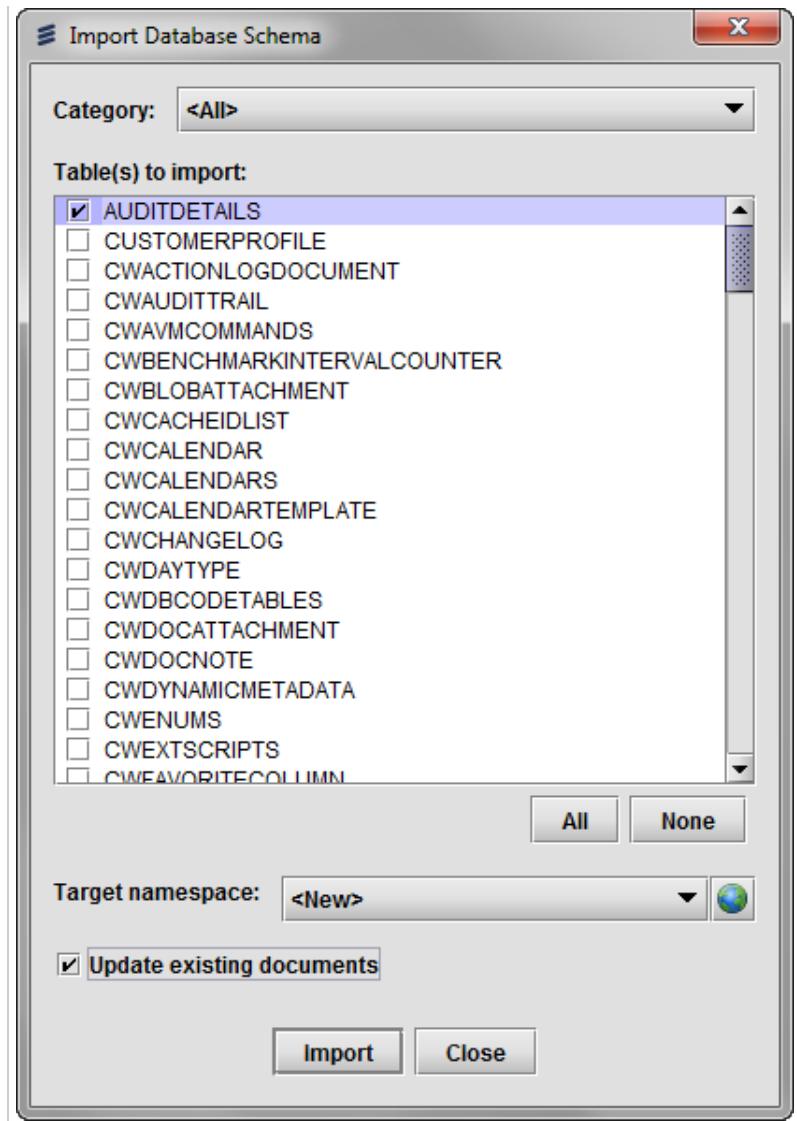
Velocity Studio always attempts to use the existing metadata objects before generating new ones. In some cases, Velocity Studio may decide to change an existing metadata object to fit the required properties, but this will only be done if the modified object is backward compatible with the original object (that is, Velocity Studio will change existing data type objects only if their base type (integer, boolean, string, etc.) is the same and the length and precision of the modified data type object can accommodate the length and precision of the original type). For Documents, Velocity Studio will add new Document leaves if needed but will never remove existing leaves. If a metadata object with the same name already exists in the specified namespace and it cannot be reused or modified, Velocity Studio will generate a new object that has the name of the existing object with a numeric suffix added for uniqueness.

A Document object will be updated/created in the namespace and will have the corresponding structure of the imported table. The required data type objects will be created/updated first, one for each table column. The **Base Type** on the **General** tab of the data type object will correspond to the column data type. These data type objects are added as leaves of the Document object and are mapped to the corresponding columns of the imported database table on the **DB Map** sub-tab in the **Map** tab of the Document object.

The screenshot shows the Velocity Studio interface for mapping database schema to document structures. The top navigation bar includes tabs for General, Variables, Methods, Mapping, DB Map, Text Map, and XML Map. The DB Map tab is selected. A dropdown for 'Logical connection' is set to 'CW50@CW0056\_LOCALDB'. Below it, there are checkboxes for 'Connected' (checked), 'Skip DB schema generation' (unchecked), and 'Map to separate table' (unchecked). On the left, a tree view under 'Tables' shows the structure of the 'AUDITDETAILS' table, with columns: DOCID, DOCMETADATATYPE, UPDATEDBY, LASTUPDATEDDATE, OPERATIONTYPE, ATTRIBUTENAME, NEWVALUE, and OLDDVALUE. On the right, a grid lists these columns with their corresponding types and mapping status. The grid has columns for Leaf, Type, and Don't Map. The 'Leaf' column lists the column names in red. The 'Type' column lists the data types: String, 256; Decimal, 9; String, 64; DateTime; String, 3; String, 128; String, 512; and String, 512. The 'Don't Map' column contains empty checkboxes. At the bottom, buttons include Rename, Remove, Map All Unmapped, and Unmap.

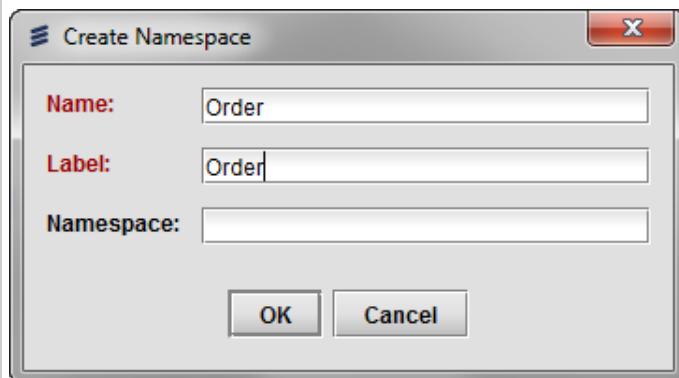
Leaf	Type	Don't Map
DOCID	String, 256	<input type="checkbox"/>
DOCMETADATATYPE	Decimal, 9	<input type="checkbox"/>
UPDATEDBY	String, 64	<input type="checkbox"/>
LASTUPDATEDDATE	DateTime	<input type="checkbox"/>
OPERATIONTYPE	String, 3	<input type="checkbox"/>
ATTRIBUTENAME	String, 128	<input type="checkbox"/>
NEWVALUE	String, 512	<input type="checkbox"/>
OLDDVALUE	String, 512	<input type="checkbox"/>

Once you select the **DB Schema** command, the [Database Login](#) dialog will open so that you can connect to the desired database where the table(s) are located. The **Import Database Schema** dialog will open where the tables to be imported as Documents can be selected.



Description of fields in the Import Database Schema dialog.

Field	Description
<b>Category</b>	Not used by this command.
<b>Table(s) to import</b>	A list of available database tables that can be imported into the metadata.
<b>All</b>	When clicked, all the database tables in the <b>Table(s) to import</b> list will be selected for import.
<b>None</b>	When clicked, all the database tables in the <b>Table(s) to import</b> list will be deselected.
<b>Target Namespace</b>	The namespace in which the data type objects and Document objects, that are based on the selected tables are created or updated. If <new> is selected, the <b>Create Namespace</b> dialog opens after clicking the <b>Import</b> button.



	You can define the <b>Name</b> , <b>Label</b> , and <b>Namespace</b> properties of the new namespace. When the <b>OK</b> button is clicked, the new namespace will be created in the metadata along with the corresponding data type objects and Document objects for the selected table(s).
<b>Update existing documents</b>	If checked, existing Document objects in the metadata will be updated to reflect changes in the selected table(s). It is your responsibility to make sure that the imported Document object does not override existing Document objects used for other purposes.

Once the import is completed, the **Import Database Schema** completed dialog opens, allowing you to either close the dialog by clicking the **Close Import Dialog** button or do another import by clicking the **Continue Import** button. If you click the **Continue Import** button, you are returned to the original **Import Database Schema** dialog.

## Views

---

The following views can be found in the [Velocity Studio User Interface](#):

**Problems**

Opens the Problems tab in the Notifications pane, which lists all errors in project.

**Search**

Opens the Search tab in the Notifications pane, which displays results of search function.

**Console**

Opens the Console tab in the Notifications pane, which displays console output of the project.

**DB Mapping**

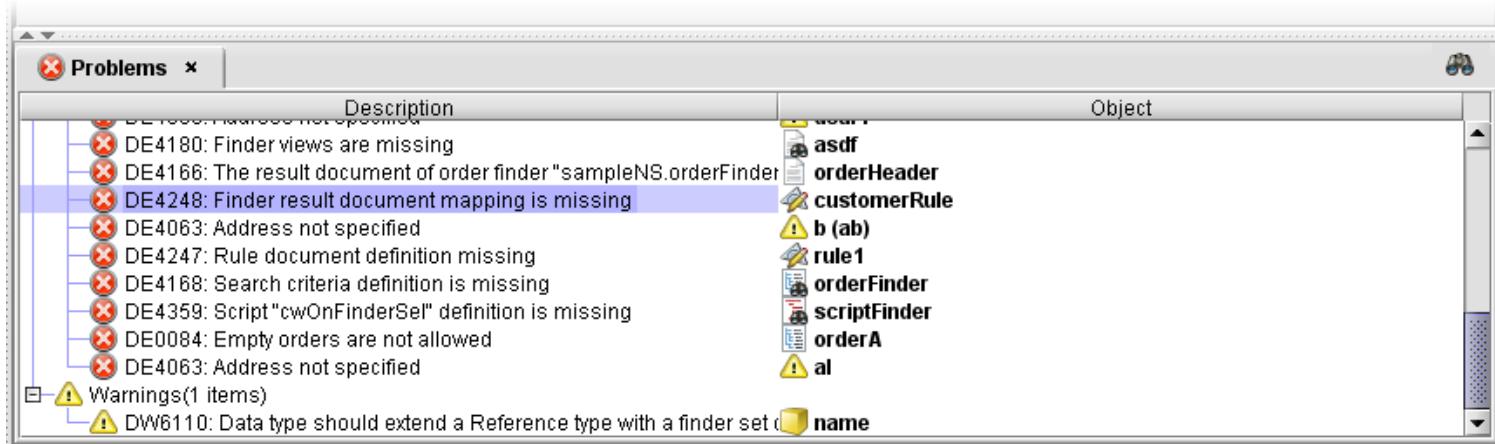
Opens the DB Mapping tab in the Notifications pane, which shows results of DB Mapping function.

**Breakpoints**

Opens the Breakpoints tab in the Notifications pane, which shows a list of scripts containing breakpoints.

## Problems Pane

The Problems pane is located at the bottom-right of Velocity Studio as the **Problems** tab, which contains all validation errors and warnings of all metadata in the project. You can expand the list of errors/warnings using the expand icon, and then double-click the individual error or warning to jump to the metadata object in question. You can then fix the problem at hand.



The Problems pane contains the following columns:

Column	Description
Description	The description of the validation error
Object	The metadata object that contains the validation error

The Problems pane groups messages by warning or error first, followed by the message code. You can also expand and collapse grouped errors and warnings as shown the previous example.

As metadata objects are created and updated by you, Velocity Studio performs validation checks on these updates and generate validation errors and warnings if any is failed. If there exists validation errors, metadata can continue to be edited and can also be saved. However, metadata cannot be [run](#) or [debugged](#). Other project functions such as to [build deployment WAR](#) or [library JAR](#), [check DB mapping](#), as well as [Upgrade System](#) are disabled. The [Debug Script](#) function in top-level Scripts is disabled as well. Therefore, **you must resolve all validation errors before running your metadata**. Warnings, on the other hand, do not disable any running command of metadata.

If the Problems pane is closed, click **Views > Problems** from the menu bar to open the pane again.

### Filter Results button

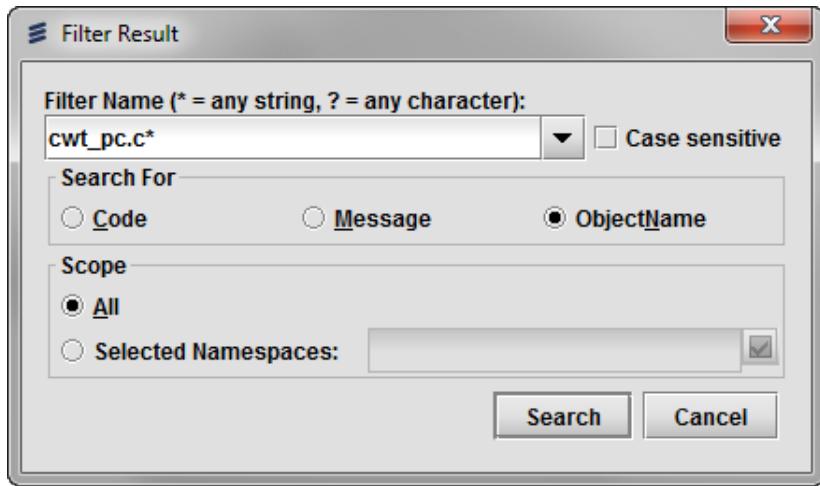
Clicking the **Filter Results** button ( ) allows you to select the following options:

- **Filter Result**, which launches the Filter Results dialog to filter the information in the Problem pane to suit your needs
- **Clear Filter**, which clears all information in the Problem pane

### Filter Result dialog

To use the Filter Result dialog to filter the information in the Problem pane, do the following:

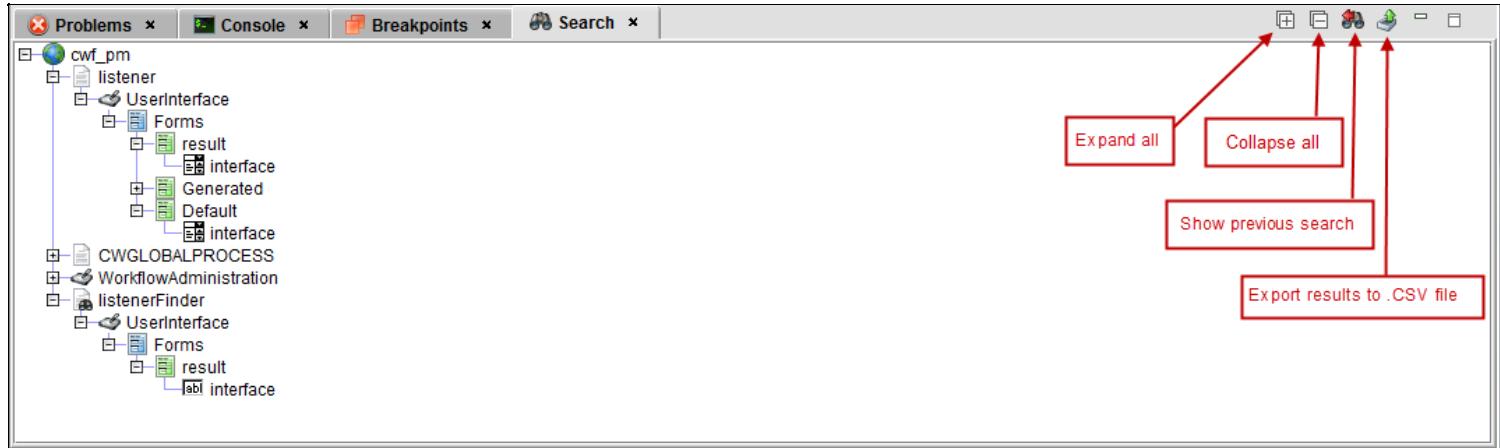
1. In the **Filter Name** field, enter your filter criteria. Wildcard (\*) and any character (?) are acceptable. In this example, filtering is for an object name using a wildcard.



2. In the **Search For** section, **Object Name** has been selected. You can also filter by **Code** or **Message**. If you leave the **Filter Name** field blank, and select either **Code** or **Message**, your results are filtered alphanumerically.
3. For the **Scope** section, the default is **All**. To narrow down your filtering, you can click **Selected Namespaces**:
  - a. Proceed to click the **Checkmark** icon to launch the Select Namespaces dialog and specify the namespaces that you want.
  - b. Click the **Save** button to return to the Filter Result dialog.
4. Click the **Search** button to filter your results, which appear in the Problem pane.

## Search Pane

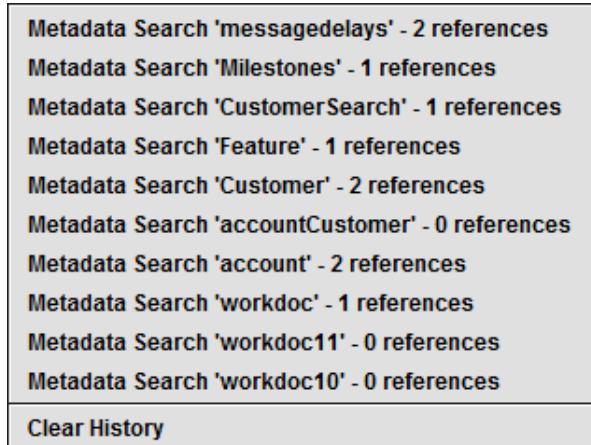
The search pane is located at the bottom-right of the Velocity Studio as the **Search** tab, which contains all matched results of a [search command](#). The search results are metadata objects listed in namespace hierarchy. You can double-click a metadata object to jump to that object in the navigation and detail panes. You can also use the Expand all and Collapse all buttons to expand or collapse the search results.



If the search pane is closed, click **Views > Search** from the menu bar to open it again. Alternatively, click [Search > Search...](#) from the menu bar to perform another search, and to open the search pane.

### Show History of Previous Searches

You can click the **Show Previous Searches** icon to display the last ten searches. The search history also includes the number of successful search references. To clear the search history, click the **Clear History** option.



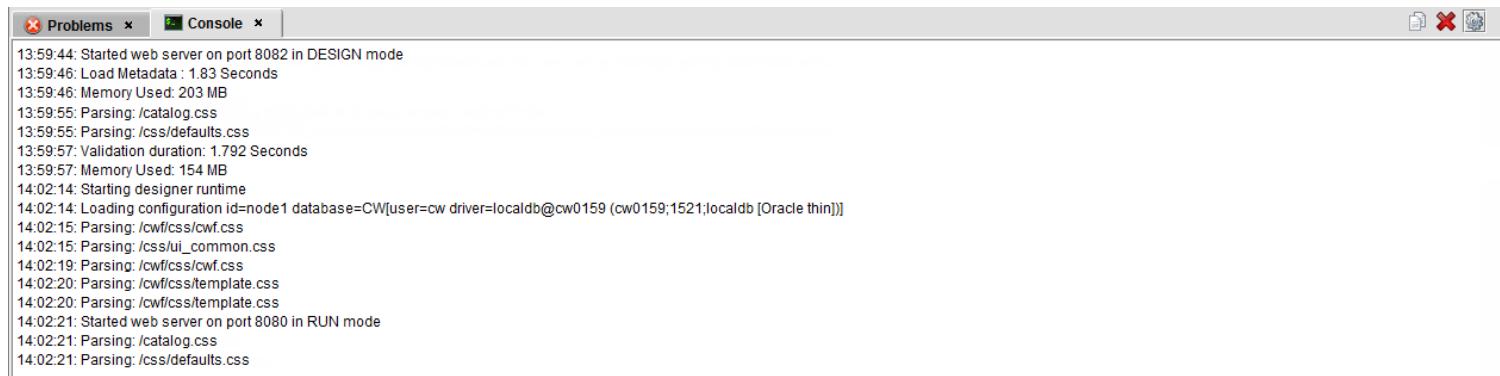
### Export Search Results to the .CSV File

To export the search results as a comma-separated values (CSV) file, do the following:

1. Right-click in the search pane and select the **Export** menu, or click the **Export results to .CSV** icon.
2. From the Export Search Results to File dialog, enter the name of the file.
3. Click the **Save** button.

## Console Pane

The Console pane is located at the bottom-right of Velocity Studio as the **Console** tab, which displays when you have opened a project, or shows system information of issued commands such as [Run](#) and [Debug](#).



A screenshot of the Velocity Studio interface showing the Console pane. The pane displays a log of events with timestamps and descriptions. The log includes:

```
13:59:44: Started web server on port 8082 in DESIGN mode
13:59:46: Load Metadata : 1.83 Seconds
13:59:46: Memory Used: 203 MB
13:59:55: Parsing: /catalog.css
13:59:55: Parsing: /css/default.css
13:59:57: Validation duration: 1.792 Seconds
13:59:57: Memory Used: 154 MB
14:02:14: Starting designer runtime
14:02:14: Loading configuration id=node1 database=CW[user=cw driver=localdb@cw0159 (cw0159;1521;localdb [Oracle thin])]
14:02:15: Parsing: /cwfcss/cwf.css
14:02:15: Parsing: /css/ui_common.css
14:02:19: Parsing: /cwfcss/cwf.css
14:02:20: Parsing: /cwfcss/template.css
14:02:20: Parsing: /cwfcss/template.css
14:02:21: Started web server on port 8080 in RUN mode
14:02:21: Parsing: /catalog.css
14:02:21: Parsing: /css/default.css
```

If a CSS parsing error occurs, Velocity Studio logs the error with its filepath in the Console pane. The Console pane also displays information about the duration of loading, validating, and starting runtime, and memory usage.

Velocity Studio has an embedded Web server to support itself as an Integrated Development Environment (IDE). One of the important system information details displayed in the **Console** is starting and stopping of the Web server in a specific *mode*, and on a particular port, as shown in the previous screenshot. There are several modes that a Web server port can be binded to:

Mode	Description
RUN mode	Indicates the project's metadata is running by AVM at this port. In RUN mode, the <a href="#">Configuration application</a> and <a href="#">System Administration application</a> are available.
CONFIG mode	Indicates the Configuration application is running at this port. System Administration App or project's metadata are not available.
HELP mode	Indicates the <a href="#">product documentation</a> is running on this port.

If the Console pane is closed, use the menu **Views > Console** to reopen the Console pane.

### Console pane buttons

The following table describes each **Console pane** button and its action when you click it:

Button	Description
	Click the <b>Copy</b> button to copy all information in the Console pane. You can then paste the information into an editor, such as Notepad, and save the information as a file.
	Click the <b>Clear All</b> button to clear all information in the Console pane.
	Click the <b>Auto-Scroll</b> button to have the information in the Console pane automatically scroll whenever new information appears.

**Note:** You can also right-click the Console pane area to access these actions. By default, the Auto-Scroll feature is on.

## Database Mapping Pane

The Database Mapping Pane is located at the bottom-right of Velocity Studio as the **DB Mapping** tab, which displays all warnings and errors of [Check DB Mapping command](#). You can expand the list of errors/warnings using the expand icon, and then double-click the individual error or warning to jump to the metadata object in question. You can then fix the DB mapping problem at hand.

The screenshot shows the Velocity Studio interface with the DB Mapping pane open. The pane has three tabs: Problems, Console, and DB Mapping. The DB Mapping tab is selected. It contains a table with two columns: Description and Object. The Description column lists errors and warnings, and the Object column lists the corresponding metadata objects. A specific warning about the 'Name' field being unmapped is highlighted with a blue selection bar.

Description	Object
Errors(1 items) DE0509: Table "CWPUSRPRTCPNTS" does not exist	CWPUSRPRTCPNTS (Participant Users)
Warnings(20 items) DW0515: Database column is larger than field "Event External Cc" DW0513: Field "Group Name" is not mapped to database DW0513: Field "Name" is not mapped to database DW0513: Field "Active" is not mapped to database DW0513: Field "Manager" is not mapped to database DW0513: Field "YEAR" is not mapped to database DW0513: Field "CALENDAR" is not mapped to database DW0513: Field "DAY" is not mapped to database	mappingForProductProperties (Mapping for product properties) user_group (User or Group) user_group (User or Group) user_group (User or Group) user_group (User or Group) calendarYearByType (Calendar Year By Type) calendarYearByType (Calendar Year By Type) calendarYearByType (Calendar Year By Type)

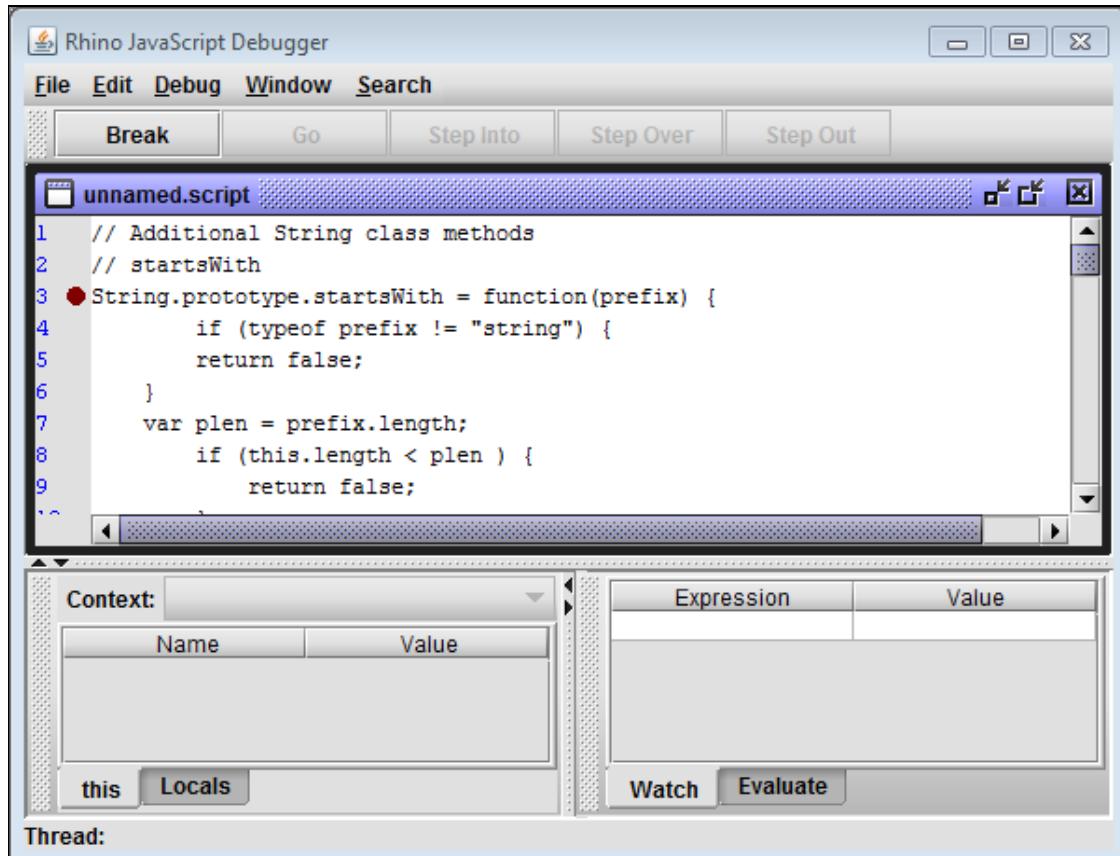
Column	Description
Description	The description of the mapping error
Object	The metadata object that contains the mapping error

If the DB Mapping Pane is closed, use the menu **Views > DB Mapping** to open the Search Pane again. Or, click [Database > Check DB Mapping](#) from the menu bar to perform another database mapping check to open the pane.

## Breakpoints Pane

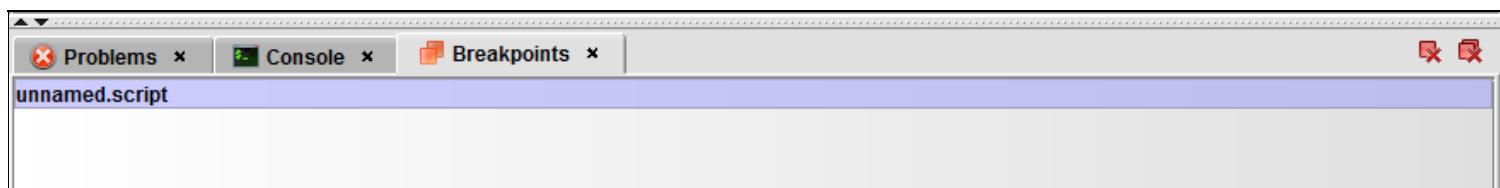
The Breakpoints Pane is located at the bottom-right of Velocity Studio as the **Breakpoints** tab, which provides a list of all scripts in Velocity Studio that have breakpoints set.

In this example, breakpoints have been set in `unnamed.script` using the JavaScript debugger.



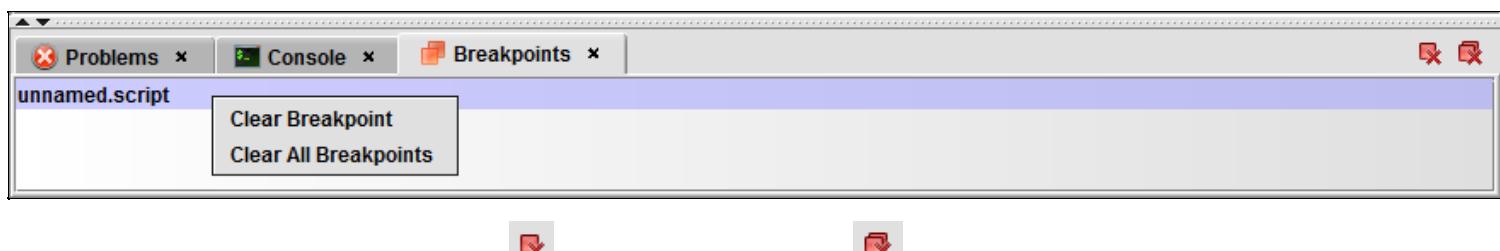
The script name also appears in the Breakpoint pane, listing it as a script that contains at least one breakpoint.

**Note:** You can double-click the script link in the Breakpoints tab, to open the script and set the focus to the corresponding object or function.



When you close the JavaScript debugger, you can clear breakpoints in a script. Right-click the script that you want and select from one of the following options:

- **Clear Breakpoint** to clear a specific breakpoint set in your script.
- **Clear All Breakpoints** to clear all breakpoints set in your script.



Alternatively, you can click the **Clear Breakpoint** ( ) and **Clear All Breakpoints** ( ) buttons.

For more information on using breakpoints and debugging your metadata, see the [Debug](#) section.

# Help

---

The Velocity Studio **Help** menu contains the following menu items and brief descriptions of each menu item:

**Help Contents**

Access the online help documentation

**About Velocity Studio**

Display product and license details

**Quality Centre**

Generate reports about the quality of your metadata

**Open Quality Report**

Open an existing quality report pertaining to your metadata

**Warning/Errors**

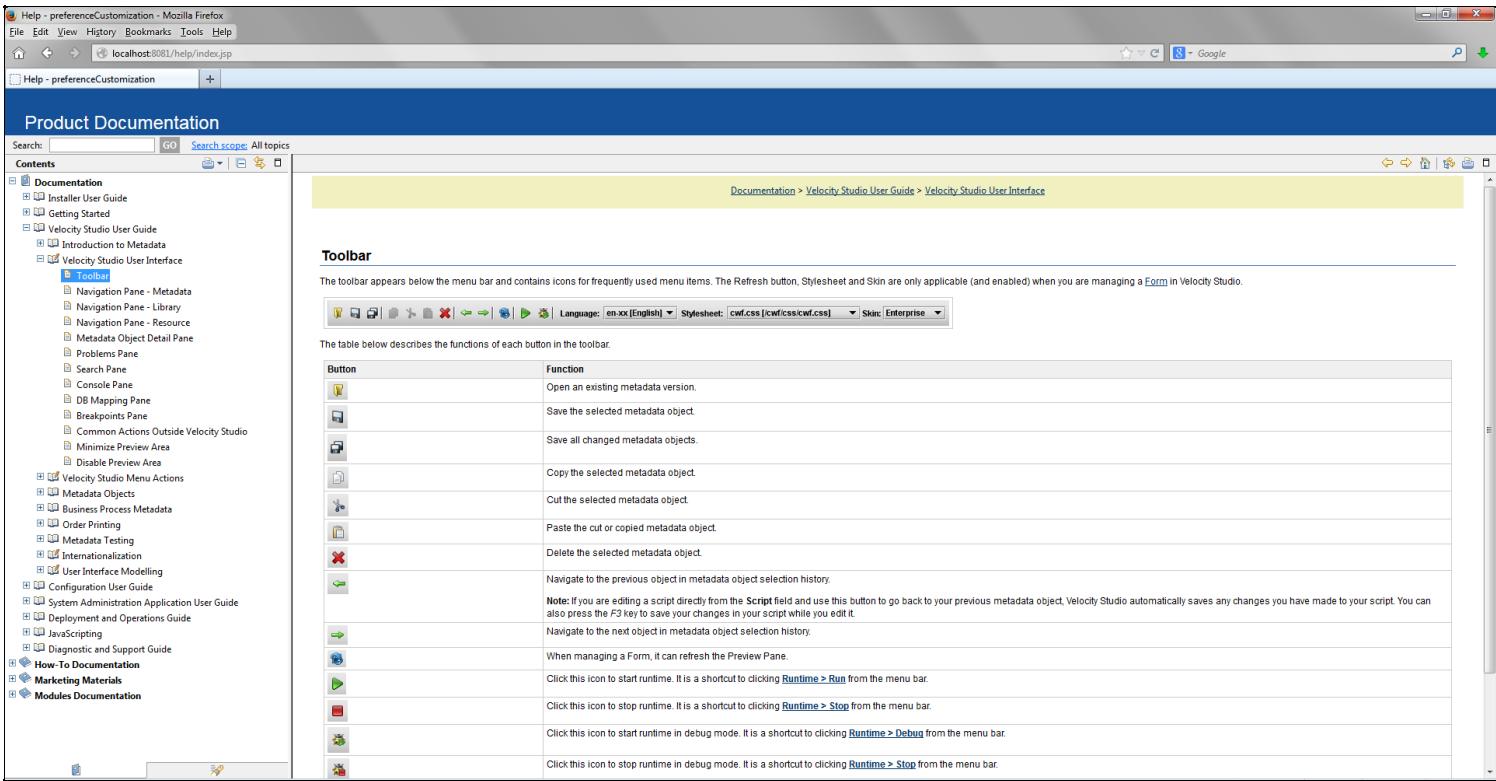
Warning Configuration in the Velocity Studio

**Show Log**

View the latest log file, which contains output and error information for the Velocity Studio

## Help Contents

By clicking the **Help > Help Contents** menu command, this product documentation is presented as Web pages in your default browser.



The screenshot shows a Mozilla Firefox browser window displaying the 'Product Documentation' page. The address bar shows 'localhost:8081/help/index.jsp'. The left sidebar contains a tree view of documentation topics under 'Documentation'. The right pane displays the 'Velocity Studio User Interface' documentation for the 'Toolbar' section. It includes a toolbar screenshot with labels for each button and a detailed table describing the function of each button. The table has two columns: 'Button' and 'Function'.

Button	Function
	Open an existing metadata version.
	Save the selected metadata object.
	Save all changed metadata objects.
	Copy the selected metadata object.
	Cut the selected metadata object.
	Paste the cut or copied metadata object.
	Delete the selected metadata object.
	Navigate to the previous object in metadata object selection history. <small>Note: If you are editing a script directly from the Script field and use this button to go back to your previous metadata object, Velocity Studio automatically saves any changes you have made to your script. You can also press the F3 key to save your changes in your script while you edit it.</small>
	Navigate to the next object in metadata object selection history.
	When managing a Form, it can refresh the Preview Pane.
	Click this icon to start runtime. It is a shortcut to clicking Runtime > Run from the menu bar.
	Click this icon to stop runtime. It is a shortcut to clicking Runtime > Stop from the menu bar.
	Click this icon to start runtime in debug mode. It is a shortcut to clicking Runtime > Debug from the menu bar.
	Click this icon to stop runtime in debug mode. It is a shortcut to clicking Runtime > Stop from the menu bar.

Browse the available topics on the left side in the topics pane. Once a topic is clicked, its documentation is shown at the right pane.

See section [Using Help](#) on how to navigate and search on this documentation.

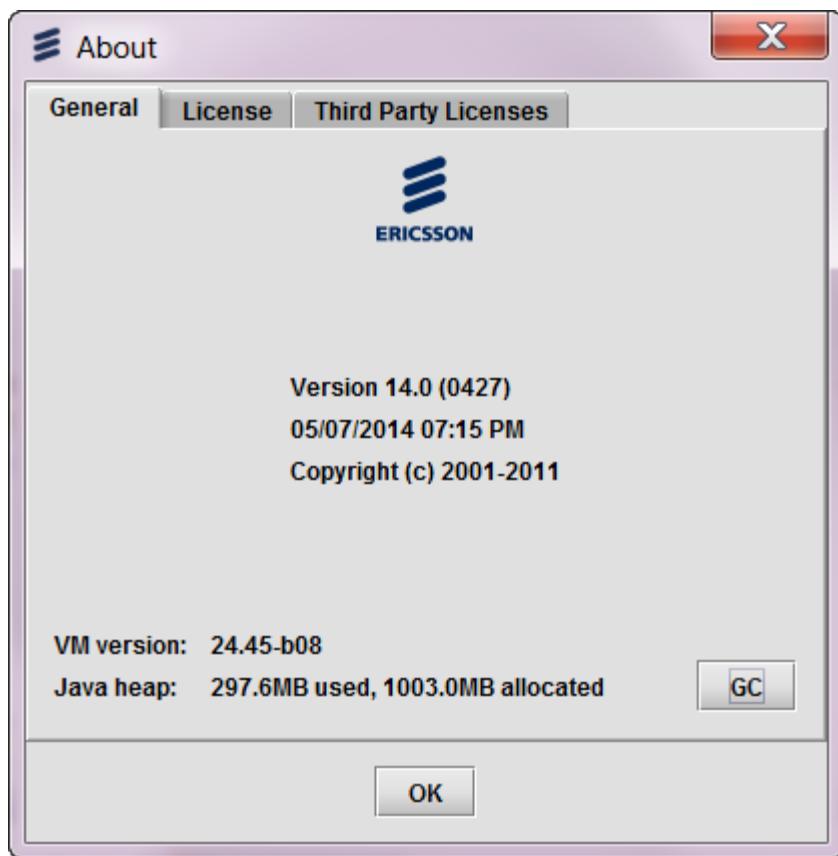
## Help - About Velocity Studio

The About dialog has the following tabbed sections:

- [General](#)
- [License](#)
- [Third-Party Licenses](#)

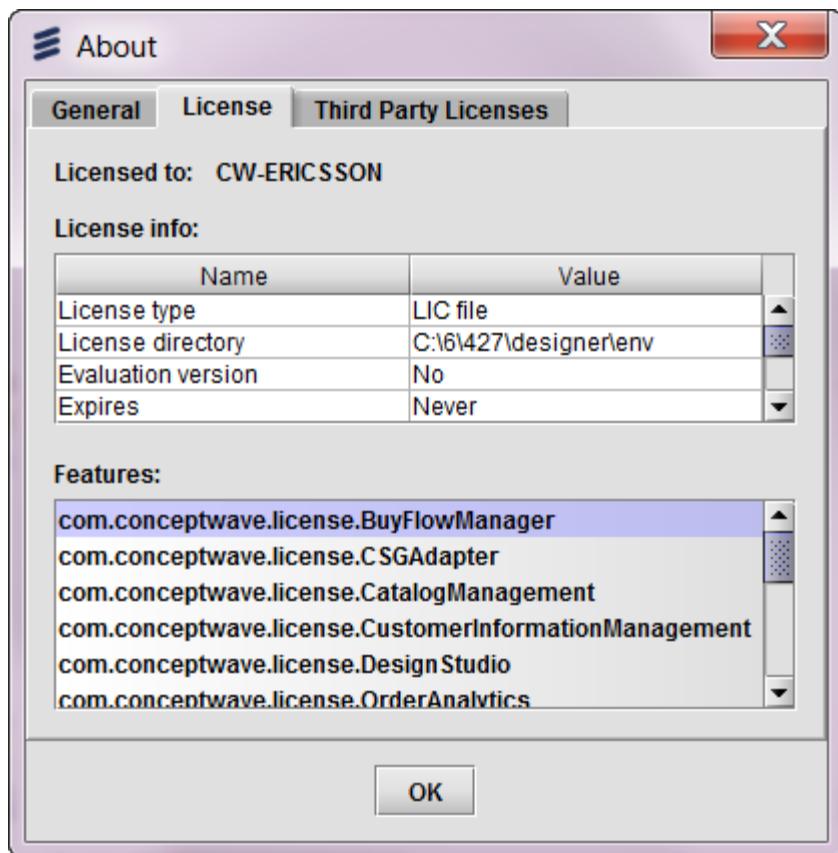
### General

This section displays the version number of your product installation. This tab also displays the VM version and Java heap info. You can click the **GC** button to force garbage collection in Velocity Studio.



### License

This section displays license information and features enabled by the licenses recognized in Velocity Studio. See section [License File Explained](#) in the *Installer Guide* for details on licenses.



### Third-Party Licenses

This section shows a list of third-party licenses used within the product. Double-click the **Information** icon next to a license, where available, for important information about the third-party license.

License	Important Information
nekohtml-2.0.4	
nekohtml-1.9.7	
openid4java	
options	
oraclejdbcdriver	
poi-3.7-20101029	
relaxngDatatype	
rhino	(i)
selenium	
serializer-2.7.0	
servlet-api-sun	
slf4j	(i)
velocity-1.6.1	
woden-api-1.0M8	
woden-impl-dom-1.0M8	
wsdl4j	
wxwidgets-2.2.4	

OK

You can also consult the ConceptWave3rdPartyPrograms.xls file, which is a part of the product installation files, to get detailed

information about each third-party program used within the product.

## Quality Centre

---

The Quality Centre command allows you to generate and review reports about the quality of your metadata. The reports contain information about the metadata elements, scripts, global processes, and user processes, as well as information about potential problems with the metadata, such as the use of deprecated APIs and older templates. Velocity Studio generates the report file as an HTML file. To access this command, click **Help > Quality Centre** from the menu bar.

The Quality Centre command offers two options:

- [Generate Default Statistics Report](#)
- [Generate Health Report](#)

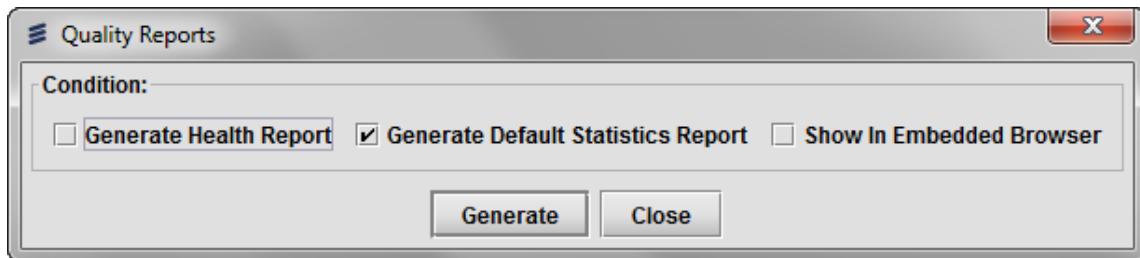
## Generate Default Statistics Report

When you use the Quality Centre command, the default option that is selected is the **Generate Default Statistics Report** option. The report that you generate by using the **Generate Default Statistics Report** option displays information about the metadata elements, scripts, global processes, and user processes.

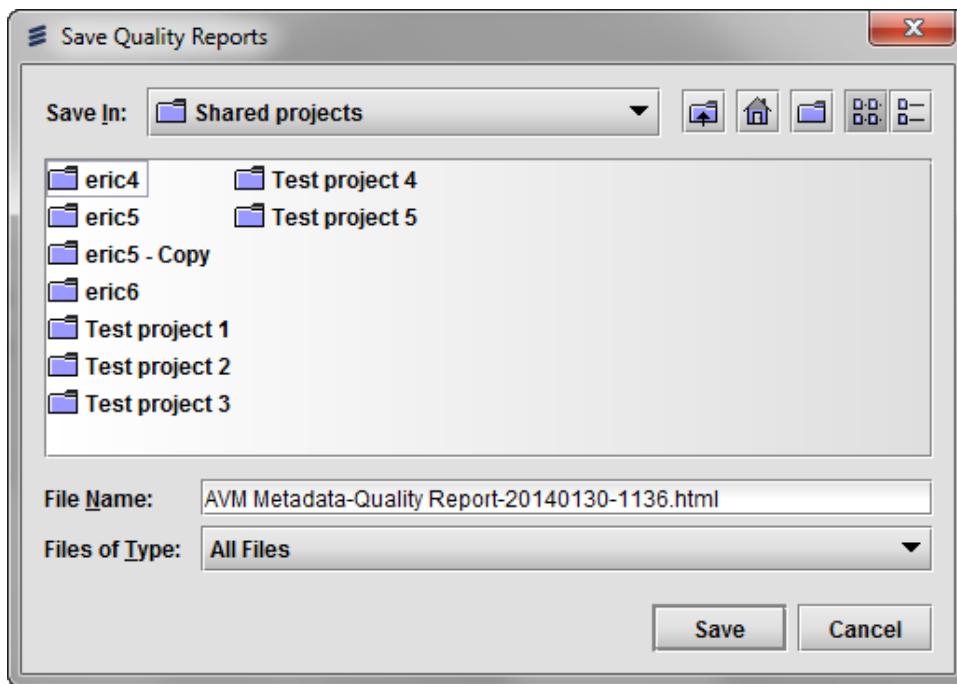
### Generate the Report

To generate the default Statistics Report, do the following:

1. [Open your metadata project](#) in Velocity Studio.
2. From the menu bar, click **Help > Quality Centre** to open the Quality Reports dialog.



3. Click the **Generate** button to open the Save Quality Reports dialog, which allows you to specify the location and the filename of the report that is to be generated. By default, the report is saved in the metadata root folder, and has a filename in the form of **AVM Metadata-Quality Report-<date>-<time>.html**, as shown in the following figure.



4. Specify the desired filename and location for the report, and then click the **Save** button.

### View the Report

Once generated, the Statistics Report is saved as an HTML file and has an associated folder with the same name.

To view the Statistics Report, double-click the HTML file. The report opens in your default Web browser, as shown in the following figure.

AVM Metadata Quality Centre Statistics Report

**Metadata Application Name : AVM Metadata**

2013/09/03

6.0.0.0

## Metadata Element

Metadata element type	Number In Application	% In Application	Number In Template	% In Template	Number In Customer Template	% In Customer Template	Total
Alert	7	100%	0	0%	0	0%	7
Binding	9	69.23%	4	30.77%	0	0%	13
Conversion Map	0	0%	62	86.11%	10	13.89%	72
Data Structure	8	4.12%	168	86.6%	18	9.28%	194
Datatype	36	4.59%	393	50.06%	356	45.35%	785
Decision Tree	3	100%	0	0%	0	0%	3
Document	39	12.46%	124	39.62%	150	47.92%	313
Exception	6	42.86%	0	0%	8	57.14%	14
External Service	6	75%	2	25%	0	0%	8
Finder	9	4.27%	106	50.24%	96	45.5%	211
Interface	36	90%	2	5%	2	5%	40
Navigation Bar	0	0%	0	0%	1	100%	1
Navigation Tree	0	0%	8	50%	8	50%	16
Order	6	85.71%	0	0%	1	14.29%	7
Participant	48	70.59%	0	0%	20	29.41%	68
Permission	4	21.05%	7	36.84%	8	42.11%	19
Signal	9	100%	0	0%	0	0%	9
Test Case	155	46.27%	0	0%	180	53.73%	335
Test Suite	10	71.43%	0	0%	4	28.57%	14
User Interface	12	4.23%	237	83.45%	35	12.32%	284

## Script

Script Type	Application Script	%	Template Script	%	Customer Template Script	%	Application Script Lines	%	Template Script Lines	%	Customer Template Script Lines	%
Global	103	14.39%	485	67.74%	128	17.88%	1395	9.81%	10277	72.25%	2552	17.94%
Element	491	8.49%	4640	80.28%	649	11.23%	2382	7.31%	26230	80.51%	3967	12.18%
Total	594	9.14%	5125	78.89%	777	11.96%	3777	8.07%	36507	78%	6519	13.93%

Global Process

Process Type	Application Processes	%	Template Processes	%	Customer Processes	%	Application Process activities	%	Template Process activities	%	Customer Template Process activities	%
Process	4	80%	0	0%	1	20%	12	80%	0	0%	3	20%
Subflows	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A	0	N/A
Total	4	80%	0	0%	1	20%	12	80%	0	0%	3	20%

## User Process

## Generate Health Report

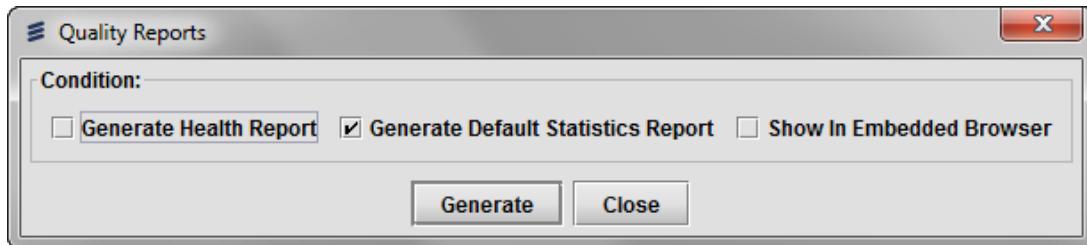
The **Generate Health Report** option of the Quality Centre command provides information on the following:

- Elements in the metadata considered extreme
- Deprecated APIs used in the metadata
- Warnings about the metadata
- Older templates used in the metadata

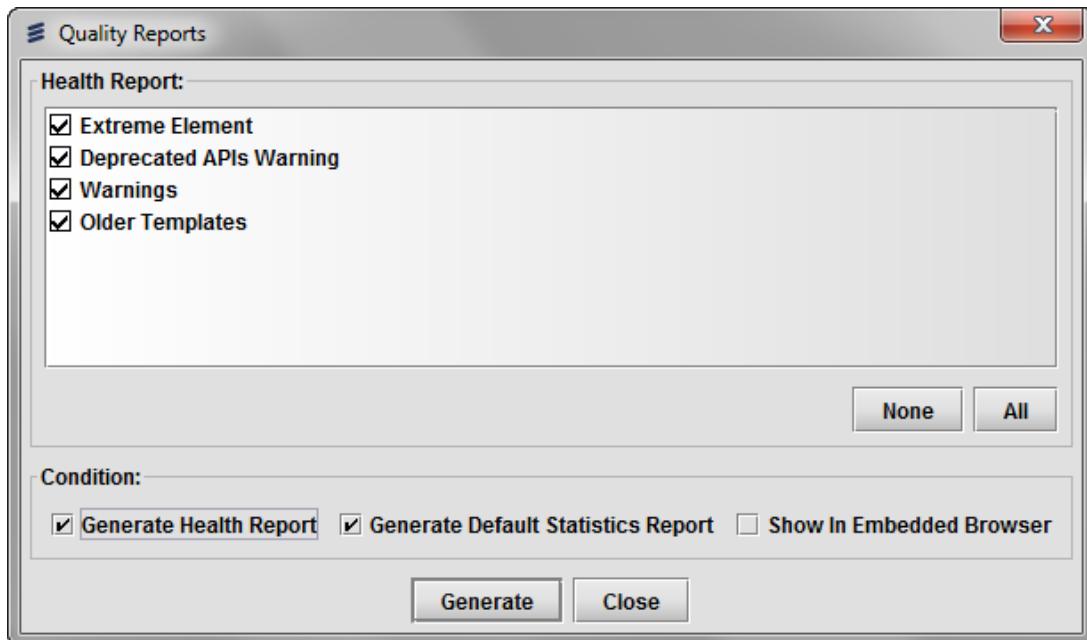
### Generate the Report

To generate a report using the **Generate Health Report** option of the Quality Centre command, do the following:

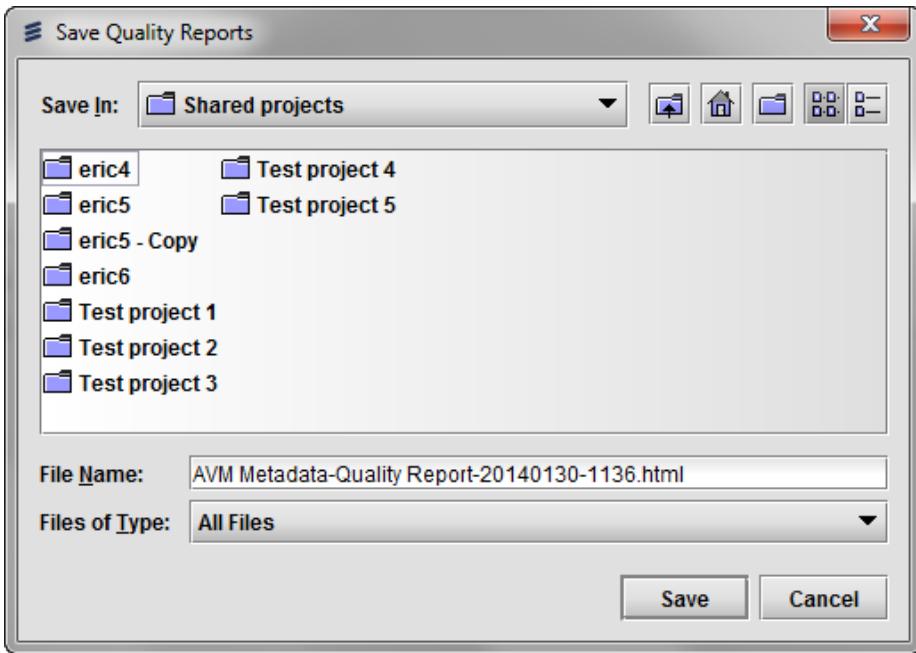
1. [Open your metadata project](#) in Velocity Studio.
2. From the menu bar, click **Help > Quality Centre** to open the Quality Reports dialog.



3. Click the **Generate Health Report** checkbox in the Quality Reports dialog, which shows the options available for the Health Report.



4. Check the desired options under the Health Report section. By default, the Health Report includes information on all the available options.
5. Click the **Generate** button to open the Save Quality Reports dialog, which allows you to specify the location and the filename of the report that is to be generated. By default, the report is saved in the metadata root folder, and has a filename in the form of **AVM Metadata-Quality Report-<date>-<time>.html**, as shown in the following figure.



- Specify the desired filename and location for the report, and then click the **Save** button.

## View the Report

Once generated, the Health Report is saved as an HTML file and has an associated folder with the same name.

To view the Health Report, do the following:

- Double-click the HTML file to open the report in your default Web browser.
- Click the **Health Report** link from the left pane to display the Health Report, as shown in the following figure.

[Statistics Report](#)  
[Health Report](#)

## AVM Metadata Quality Centre Health Report

**Metadata Application Name : AVM Metadata**

**Mon Sep 16 12:50:38 EDT 2013**

**6.0.0.0**

[Extreme Element](#)  
[Metadata Warnings](#)  
[Deprecated API Warnings](#)  
[Older Templates](#)

### Extreme Element

#### Process with resume after activity

Element Name	Number
worklistNS.subprocess_withResume	1
worklistNS.subflow_withResume	1
worklistNS.handler_process_withResume	1
worklistNS.TT_13459_nestedOnException_resumeInother	1

### Warning for subflow

Element Name	Number
worklistNS.TT12990_topLevel_subflow	1
ns13032.pr2	1
ns13032.sub	1
ns13032.subTestF	2
ns13032.prTest	2
ns13032.prTestF	1
worklistNS.topLevel_SF	1
worklistNS.TT3977_Seq_SubFlow	1
ns13032.mPr	2

3. Click the links in the middle pane, such as **Metadata Warnings**, **Deprecated API Warnings**, and **Older Templates**, to view the different sections of the report.

## Open Quality Report

---

Clicking **Help > Open Quality Report** from the menu bar opens that last generated [quality report about your metadata](#) using your default Web browser. This option is only available after you have generated a quality report.

**Note:** This action only opens the last report that you generated. To open previous reports, you must go to the folder where you saved them and open the pertinent HTML file that you want.

### Relocate Quality Report Files to Another Directory

The quality report location appears in the .setting file of your metadata project. The following is an example:

```
C:\Users\testuser\Documents\Ericsson Metadata-Quality Report-20140331-0826.html
```

When you relocate your quality report files, you must change this property to open the report in Velocity Studio.

When you generate a quality report, one folder contains all HTML, image and .js files. Additionally, an index.html is created, which links to all necessary files. This HTML file has the same name as the folder. The index.html file with a relative path to the report is generated inside the report folder. As a result, when you relocate your report to other folder locations, you can still access the report without performing any changes to the report.

## Warning / Errors

In the Velocity Studio, warning messages and errors appear on the [Problems pane](#). You cannot run a project that contains errors, whereas you can when your metadata contains warnings.

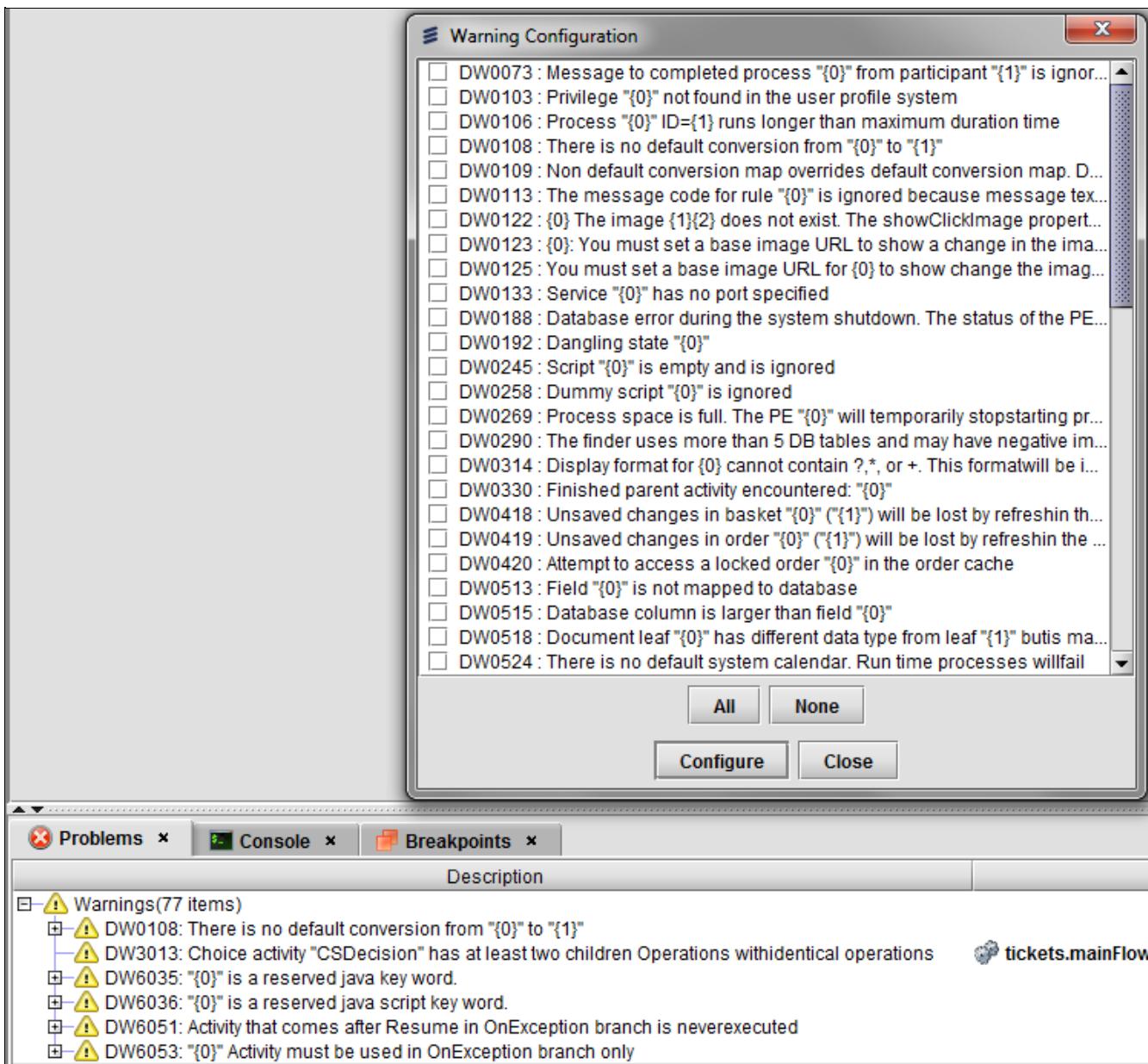
Clicking **Help > Warning/Errors** menu allows you to state whether you want the warning to show as just warning or as an error. For example, invalid dates in metadata appear as warnings and are ignored. With this command, you can upgrade the warning message to appear as an error message that prevents starting runtime until all errors are fixed.

### Change Warning Message into an Error

To change warning message into an error, follow these steps:

1. From the menu bar, click the **Help > Warning/Errors**.
2. The **Warning Configuration** dialog appears with a list of warnings.

**Note:** The same warnings also appear in the Problems pane.



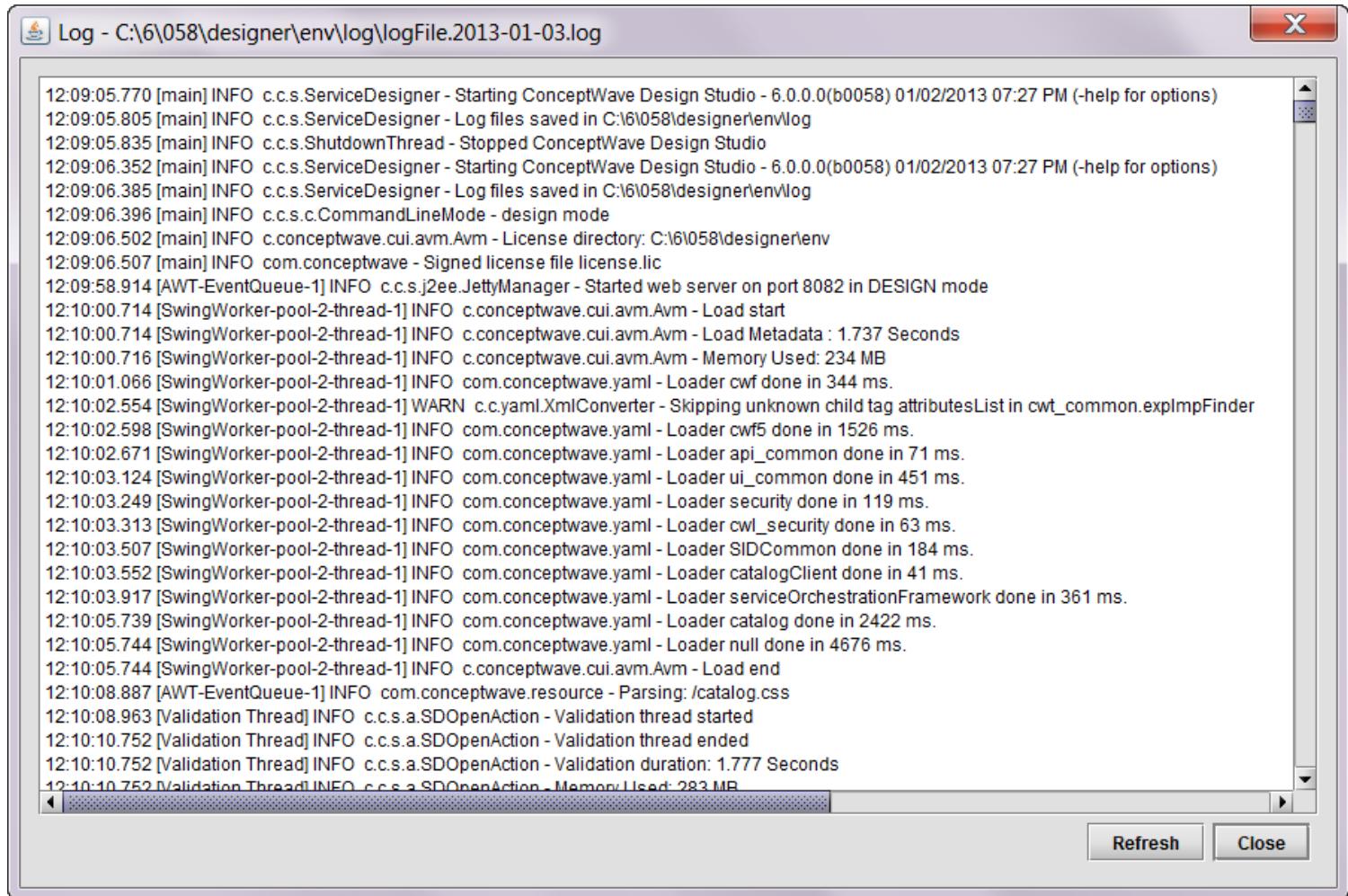
3. Select a warning from the list (for example, *DW0108: There is no default conversion from "{0}" to "{1}"*), and click the **Configure** button.
4. The selected warning appears as an error in the Problems pane.

Problems		Console
	Description	Object
<input type="checkbox"/>	Errors(26 items)	
<input checked="" type="checkbox"/>	DW0108: There is no default conversion from "{0}" to "{1}"	
<input type="checkbox"/>	Warnings(3 items)	

You can click the **All** button to select all warnings in the list. To deselect or change the error message back to warning, click the **None** button.

## Show Log

From the menu bar, click the **Help > Show Log** menu command to view the latest log file, which contains output and error information for the Velocity Studio. This information is useful to review when you are troubleshooting problems with your metadata.



The screenshot shows a Windows-style application window titled "Log - C:\6\058\designer\env\log\logFile.2013-01-03.log". The window contains a large text area displaying a log file with numerous entries. The log entries are timestamped and show various informational messages from the ConceptWave Design Studio application. At the bottom right of the window, there are two buttons: "Refresh" and "Close".

```
12:09:05.770 [main] INFO c.c.s.ServiceDesigner - Starting ConceptWave Design Studio - 6.0.0.0(b0058) 01/02/2013 07:27 PM (-help for options)
12:09:05.805 [main] INFO c.c.s.ServiceDesigner - Log files saved in C:\6\058\designer\env\log
12:09:05.835 [main] INFO c.c.s.ShutdownThread - Stopped ConceptWave Design Studio
12:09:06.352 [main] INFO c.c.s.ServiceDesigner - Starting ConceptWave Design Studio - 6.0.0.0(b0058) 01/02/2013 07:27 PM (-help for options)
12:09:06.385 [main] INFO c.c.s.ServiceDesigner - Log files saved in C:\6\058\designer\env\log
12:09:06.396 [main] INFO c.c.s.c.CommandLineMode - design mode
12:09:06.502 [main] INFO c.conceptwave.cui.avm.Avm - License directory: C:\6\058\designer\env
12:09:06.507 [main] INFO com.conceptwave - Signed license file license.lic
12:09:58.914 [AWT-EventQueue-1] INFO c.c.s.j2ee.JettyManager - Started web server on port 8082 in DESIGN mode
12:10:00.714 [SwingWorker-pool-2-thread-1] INFO c.conceptwave.cui.avm.Avm - Load start
12:10:00.714 [SwingWorker-pool-2-thread-1] INFO c.conceptwave.cui.avm.Avm - Load Metadata : 1.737 Seconds
12:10:00.716 [SwingWorker-pool-2-thread-1] INFO c.conceptwave.cui.avm.Avm - Memory Used: 234 MB
12:10:01.066 [SwingWorker-pool-2-thread-1] INFO com.conceptwave.yaml - Loader cwf done in 344 ms.
12:10:02.554 [SwingWorker-pool-2-thread-1] WARN c.c.yaml.XmlConverter - Skipping unknown child tag attributesList in cwt_common.explmpFinder
12:10:02.598 [SwingWorker-pool-2-thread-1] INFO com.conceptwave.yaml - Loader cwf5 done in 1526 ms.
12:10:02.671 [SwingWorker-pool-2-thread-1] INFO com.conceptwave.yaml - Loader api_common done in 71 ms.
12:10:03.124 [SwingWorker-pool-2-thread-1] INFO com.conceptwave.yaml - Loader ui_common done in 451 ms.
12:10:03.249 [SwingWorker-pool-2-thread-1] INFO com.conceptwave.yaml - Loader security done in 119 ms.
12:10:03.313 [SwingWorker-pool-2-thread-1] INFO com.conceptwave.yaml - Loader cwl_security done in 63 ms.
12:10:03.507 [SwingWorker-pool-2-thread-1] INFO com.conceptwave.yaml - Loader SIDCommon done in 184 ms.
12:10:03.552 [SwingWorker-pool-2-thread-1] INFO com.conceptwave.yaml - Loader catalogClient done in 41 ms.
12:10:03.917 [SwingWorker-pool-2-thread-1] INFO com.conceptwave.yaml - Loader serviceOrchestrationFramework done in 361 ms.
12:10:05.739 [SwingWorker-pool-2-thread-1] INFO com.conceptwave.yaml - Loader catalog done in 2422 ms.
12:10:05.744 [SwingWorker-pool-2-thread-1] INFO com.conceptwave.yaml - Loader null done in 4676 ms.
12:10:05.744 [SwingWorker-pool-2-thread-1] INFO c.conceptwave.cui.avm.Avm - Load end
12:10:08.887 [AWT-EventQueue-1] INFO com.conceptwave.resource - Parsing: /catalog.css
12:10:08.963 [Validation Thread] INFO c.c.s.a.SDOpenAction - Validation thread started
12:10:10.752 [Validation Thread] INFO c.c.s.a.SDOpenAction - Validation thread ended
12:10:10.752 [Validation Thread] INFO c.c.s.a.SDOpenAction - Validation duration: 1.777 Seconds
12:10:10.752 [Validation Thread] INFO c.c.s.a.SDOpenAction - Memory Used: 283 MB
```

If you require an older version of a log file, you can access all logs from the `<installation_folder>\designer\env\log` folder.

## **Validation of Metadata Element Names**

---

Validation of metadata element names does not allow Java and JavaScript reserved words, such as *instanceof*, Java keywords, and names of product APIs. A warning message appears for metadata that has used a reserved word.

## JavaScript Dot Notation

You can use JavaScript dot notation to access both standard JavaScript objects and metadata objects within the product. Additionally, element scripts can be called by name as object methods.

As an example, under the Ns namespace, the Ds data structure has a DsElement datatype element. In your Velocity Studio JavaScript, you can access the element as follows:

```
var ds = new DataStructure("Ns.Ds");
ds.DsElement="test";
```

If the element is an array, accessing it is as follows:

```
var ds = new DataStructure("Ns.Ds");
ds.DsElement[0]="test1";
ds.DsElement[1]="test2";
```

If the DsElement element has method1() and method2() as defined methods, in your Velocity Studio JavaScript, invoke these methods:

```
var ds = new DataStructure("Ns.Ds");
ds.DsElement="test";
ds.DsElement.method1();
```

If the element is an array, do the following:

```
var ds = new DataStructure("Ns.Ds");
ds.DsElement[0]="test1";
ds.DsElement[0].method1();
```

## Virtual Field

Metadata objects and methods can be marked as **Virtual**. On the General tab, of a metadata object a **Virtual** field appears as



If checked, it provides the ability to create methods in template metadata that does not have an implementation, but can be used in a template script and be bound to user interface elements. The implementation is done in user metadata.

## Top-Level Virtual Objects

The following metadata objects can be virtual:

Root Metadata Object	Metadata Objects
Data Dictionaries	<ul style="list-style-type: none"><li>• Data Structures</li><li>• Documents</li></ul>
Finders	<ul style="list-style-type: none"><li>• Document Finder</li><li>• Interface Finder</li><li>• Order Finder</li><li>• Rule Finder</li><li>• Script Finder</li><li>• SQL Finder</li></ul>
Global Scripts	Scripts
Orders	Orders
Presentation	<ul style="list-style-type: none"><li>• User Interface Contributor</li><li>• User Interface</li></ul>
Security	Permission

Top-level virtual metadata elements that belong to templates must be overridden by another metadata element, or be in the metadata header's replacement map, otherwise a Velocity Studio validation error occurs. Top-level virtual metadata elements cannot be instantiated at runtime through script. An error is thrown if it is attempted. Virtual objects cannot be **Final** or **Private**.

Metadata objects that are not top-level automatically inherit the virtual property of their parent and cannot specify their own virtual property. If the parent object is virtual, then the child object is also virtual. If the parent object is not virtual, then the child object is also not virtual.

## Virtual Methods

**Virtual** methods must exist under virtual top-level metadata elements. If a non-virtual top-level metadata element contains a virtual method, it results in a Velocity Studio validation error. If a non-virtual metadata element extends a virtual metadata element, then it must override those virtual methods that are defined in virtual metadata but has not been overridden in the chain of inheritance. For example:

- A virtual Data Structure *Person* contains two virtual *getName* and *setName* methods.
- A virtual Data Structure *ReadPerson* extends *Person* and override *getName* method.
- If a non-virtual Data Structure *UpdatePerson* extends *ReadPerson*, then Data Structure *UpdatePerson* only has to override the *setName* method since *getName* method has already been overridden.

Any virtual method that has not been overridden produces a validation error. Virtual methods cannot be invoked through the *Super* call in the overriding method. It results in a *method not found* error.

Virtual Methods include all types of object methods including script methods, permissions, user action methods, etc. Virtual methods do not have scripts. In the case of permission scripts, the states and privileges, virtual methods cannot be **Final** or **Private**.

## Runtime Access to Selected Metadata Objects

---

You have access to selected metadata elements at [runtime](#), such as Finder, DataType, DatabaseConnection, and more. A number of methods from the following classes have been changed to support this enhancement:

- MetadataProcess class:
  - getAllGlobalProcesses()
  - getAllUserProcesses()
  - processDocument()
  - activities()
  - topActivity()
  - priority()
  - memoryProcess()
  - global()
- MetadataActivity class
  - activityType()
  - children()
  - milestone()
  - nextActivity()
  - milestoneConditional()
  - position()
  - duration()
  - getExpectedDueDate()
- MetadataFinder class (scriptable):
  - selectInputDocument()
  - selectOutputDocument()
  - getInputDocument()
  - getOutputDocument()
  - detailFormName()
  - maxRows()
  - finderType()
  - views()
- MetadataFinderView class (scriptable):
  - searchFormName()
  - resultFormName()
  - searchUI()
  - resultUI()
  - searchImmediately()
  - showSearchForm()
  - runInitScript()

For more information, refer to the [JavaScript Documentation](#).

## Metadata Root

---

Metadata Root is the top-level node in **Metadata** tab of **Navigation** pane. Unlike other nodes in the pane, this node cannot be removed. It enables you to specify metadata application-wide settings, as well as adding namespaces to the application metadata.

The User Interface is used to display data from metadata objects at runtime. There are two different types of User Interfaces available:

- *Included User Interface*
- *Top-level User Interface*

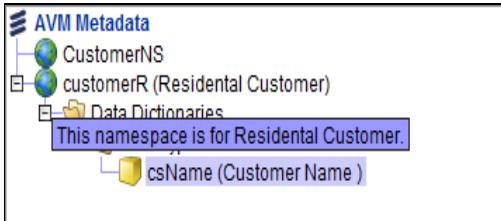
Included User Interfaces are defined within a metadata object that may contain User Interface objects such as **Documents**, **Orders**, and **Finders**. With an included User Interface, the *model* in MVC pattern connects with the controller. Top-level User Interfaces are defined as a standalone metadata object, although their model is attached to the data source. For more information, see [User Interface](#).

### Notes:

- Performing a **Save** function on the metadata root does not save all the metadata underneath the node; it only saves the changes in the metadata root.
- The top-level metadata object has a maximum Type ID limit of 32,000.

## Hover Metadata Node for Tooltip Description

You can hover your mouse pointer over any metadata node to see its description in a tooltip. The **Description** field for your metadata node must contain information for this tooltip to appear.



## Metadata Root General Properties

Metadata properties can be defined or updated by clicking the metadata root object. The following figure shows a sample **General** tab.

General	Library	Languages	Aliases	Metadata Replacement	Application Page	Contributors	Feature Management
Label:	AVM Metadata						
Author:	KT						
Version:							
Description:	<input type="button" value="Edit"/>						
Created:	Thu Jan 09 11:26:58 EST 2014						
Updated:	Thu Jan 09 11:37:12 EST 2014						
Internal Name:	OF						
Internal Version:	1						
Product Version:	6.0.0.0						
Auto-Show Application:	<NONE>						
Template Group:							
Style Sheet:	<NONE> <input type="checkbox"/> Use size from css <input type="checkbox"/> Debug UI Path						
	<input type="checkbox"/> Enforce						

The **General** tab contains the following fields:

Field	Description
Label	Mandatory. Label of the current metadata version. This field defaults to the product's root metadata name.
Author	Mandatory. Author of the metadata. This field defaults to Unknown.
Version	Metadata version ID.
Description	Description text.
Created	System-assigned metadata creation date.
Updated	System-assigned metadata update date.
Internal Name	Specify the internal name of the metadata.
Internal Version	Specify the <a href="#">metadata internal version</a> . This version number is eight digits long and is editable.
Product Version	Indicates the product version number of Velocity Studio.
Auto-Show Application	This field defaults to NONE. Select an application that you want to automatically show from the drop-down menu. You can view a dynamic, full list of applications by creating a user action method that returns the selectApp page. When you call this action, the entire list of applications appears.
Template Group	This field indicates the template group that your metadata belongs to, if applicable. When your metadata is used as a template (that is, .jar file), it appears under a group folder that you have specified in this field in Velocity Studio's Library tab. To define multiple group names in this field, separate each name with a comma.
Style Sheet	Click the drop-down menu and select the style sheet that you want to use.
Enforce	By default, this property is unchecked. Select this checkbox to indicate that the current metadata is to be used as a template. Other metadata can use the current metadata as a template, allowing you to show objects, such as finders, with proper CSS, unless a different styling on the finder is provided. This property

	<p>promotes the reuse of template objects, while keeping the same style that the objects were originally designed and ensuring that they display properly at runtime.</p> <p>See the example that follows this table on how to use this property.</p>
<b>Use size from CSS</b>	Select this option to use the size specified in your style sheet.
<b>Debug UI Path</b>	When you select this property's checkbox, the full form path is shown at runtime.

## Example: Using the **Enforce** property

The following is an example using using the **Enforce** property:

1. Create a metadata with one application page. Ensure that you also create a URL mapping.
2. In the header property, set a stylesheet and select the **Enforce** property.
3. Create a user interface that uses styles from this stylesheet.
4. Display this user interface during runtime. Make sure that you see the styles that you have set.
5. Create a .jar file for this metadata.
6. Create a new metadata and add the .jar file from the Library tab.
7. Create an application that displays the user interface that you have previously created.
8. Save and run your metadata. This user interface can still be styled by the styles that were originally used.

## Metadata Internal Version

In Velocity Studio, clicking **Database > Upgrade System** from the menu bar ensures that your database is compatible with the current version of your product release. After invoking the Upgrade System command, the system creates a .sql file that contains an upgrade SQL script.

Creating the .sql file consists of two parts:

1. Generating SQL statements according to changes in mapping the database documents.
2. Appending SQL statements to the .sql file, which are retrieved from a predefined .txt file according to the current metadata internal version.  
This file, <*InternalName*>SQLUpgrades.txt, includes changes required for the current release.

For the second part of creating the .sql file, system, template, and user metadata can be upgraded, with each set of metadata having its own predefined file (that is, its own <*InternalName*>SQLUpgrades.txt file).

### Internal Name and Internal Version fields for User Metadata

In Velocity Studio, clicking your metadata root node displays the **General** tab in the right pane. The **Internal Name** and **Internal Version** fields are mandatory to perform the upgrade.

General		Library	Languages	Aliases	Metadata Replacement	Application Page	Contributors	Feature Management
Label:	AVM Metadata							
Author:	KT							
Version:								
Description:	<input style="width: 15px; height: 15px;" type="button" value="..."/>							
Created:	Thu Jan 09 11:26:58 EST 2014							
Updated:	Thu Jan 09 11:37:12 EST 2014							
Internal Name:	OF							
Internal Version:	1							
Product Version:	6.0.0.0							
Auto-Show Application:	<NONE>							
Template Group:								
Style Sheet:	<NONE>							
<input type="checkbox"/> Use size from css								
<input type="checkbox"/> Debug UI Path								

When you first view the information on the **General** tab, only the **Internal Name** field can be edited. When you save your metadata, this field is grayed out and the **Internal Version** field is then available for input. The **Internal Version** field is an eight-digit number (for example, 12345678) and is editable.

### Locations of the <*InternalName*>SQLUpgrades.txt file

The <*InternalName*>SQLUpgrades.txt file for each type of metadata can be found in these locations:

- For system metadata, its .txt file is located under **Resources**.
- For template metadata, its .txt file is located under the template metadata resources directory.
- For user metadata, its .txt file is located under the user metadata resources directory

If the <*InternalName*>SQLUpgrades.txt files exist for both templates and user metadata, and there are upgrades for the current product release, selecting **Database > Upgrade System** from the menu bar generates a .sql file containing SQL upgrading scripts for system, templates, and user

metadata.

## CWPRODUCTPROPERTIES Table

The CWPRODUCTPROPERTIES table contains a record for system metadata, every template, and user metadata in which the column ID receives the following values:

- ID="CW" for system metadata
- ID=<value of **Internal Name**> for template metadata
- ID=<value of **Internal Name**> for user metadata

When you click **Database > Upgrade System** from the menu bar, a .sql file is generated and processed, with the CWPRODUCTPROPERTIES table updated accordingly.

**Note:** If a template is removed from the metadata templates list, the record containing an ID equal to the removed template name is also removed from the CWPRODUCTPROPERTIES table.

## Add a New Template to Metadata

Suppose that you have clicked **Database > Upgrade System** from the menu bar and have run the .sql file. When you add a new template to metadata that contains the <*InternalName*>SQLUpgrades.txt file, according to the current metadata internal version, an upgrade is required.

## Add and Remove the Same Template in Metadata

Assume that you have a template that requires an upgrade, as indicated by the current metadata internal version. You are required to click **Database > Upgrade System** from the menu bar and then run the .sql file. If you first remove the template from your metadata and then decide to add the same template, or the same template with a different name, you are required to click **Database > Upgrade System** once again.

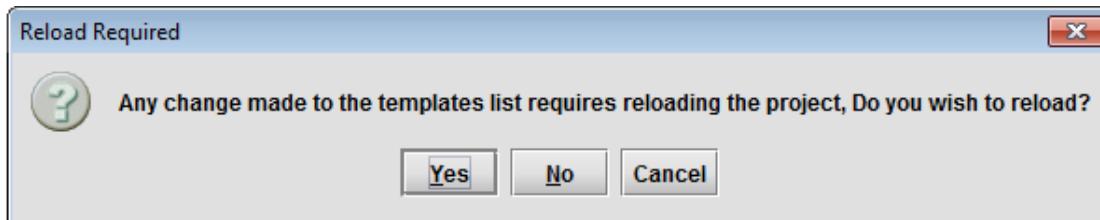
## Metadata Root Libraries

The **Library** tab in the top-level metadata object manages all project-specific libraries included in the project.

### Add a Library JAR File

Click the **Add** icon () to include a library by adding a JAR file using the opened dialog box. The JAR file must be a library based on another project that was packaged using the [Build Library JAR](#) menu command by Velocity Studio. Once the library inclusion is successful, the name of the library JAR file is added to the list, and the JAR file is stored in the *templates* folder of project's root directory.

A (manual) project reload is necessary to have the added libraries to appear in Velocity Studio. In fact, Velocity Studio prompts you to do so once you save or navigate to another metadata object. Simply open the same project again to reload.



After the project is reloaded, you can find and access metadata of the included library in the [Library](#) tab of [Navigation](#) pane.

### Remove a Library JAR File

To remove a library from the list, highlight the library that you want and then click the **Remove** icon ()

### Sort Library JAR Files

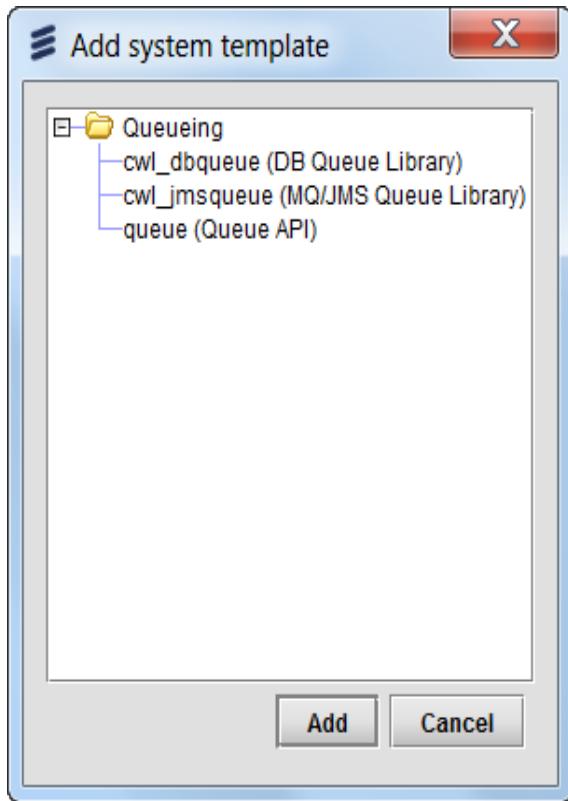
To sort a library JAR files list alphabetically, click the **Sort** icon ()

### Add a System Template to a Project

To add an system template to your metadata project, complete these steps:

1. Click the **Add** button () to launch the Add system template dialog.

2. Highlight the optional templates that you want to add and then click the **Add** button.



3. The system templates appear in the Library tab. Click the **Save** button to save your changes.

**Notes:**

- The included metadata files are also subject to [metadata validation](#) as well as [DB mapping check](#) before you can run your project.
- After you have added all optional system templates, clicking the **Add** button results in a message appearing, indicating that you have added all available system templates.
- The SystemTemplateSettings.java file defines the list of optional system templates, which includes those that are not mandatory.

#### Automatically Loaded Libraries

The following mandatory libraries are loaded by default:

- api\_common
- ui\_common
- security
- cwl\_security

If your project metadata does not include these libraries in your metadata header, these libraries are always loaded. If your project metadata contains these JAR files within its own *template* folder, those JAR files are used. Otherwise, the JAR files in the installation's *modules* folder are used instead.

#### System JAR Files

The following JAR files contain the product system metadata, which are located in your <*installation\_folder*>\modules folder:

- cwf.jar
- cwf5.jar

**Note:** The cwf5.jar is optional. None of the system templates depend on this .jar file. Instead, cwf5.jar is to be used for backward compatibility with your metadata. New metadata projects need to use the other mandatory and pertinent templates, as needed.

## Metadata Root Languages

The **Languages** tab in top-level metadata object defines the list of languages to be supported by the application runtime of the project, and the default language to use when no language is specified in the [browser language setting](#).

This tab, along with [Translation Import](#) and [Translation Export](#) functionality, provides runtime support for [internationalization](#) in applications.

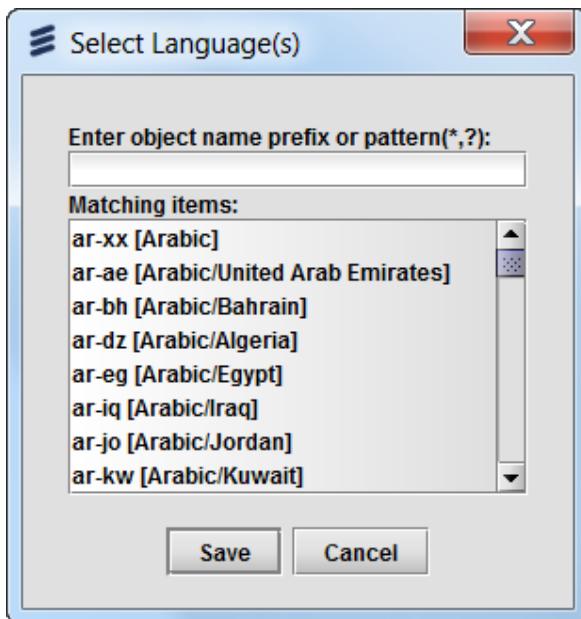
To ensure that you have access to the language of your choice while configuring the metadata:

- You need to add the list of supported languages under the Metadata Header object in the Languages tab.
- You must have a label for the default language (en-xx) for translations of other languages to work. If you do not specify a label in English, other languages do not appear in the list.

The screenshot shows the 'Languages' tab selected within a larger configuration window. The 'Languages' tab is highlighted in blue. Below the tabs, there is a list titled 'Available:' containing two entries: 'en-xx [English]' and 'fr-ca [French/Canada]'. To the right of this list are two small icons: a blue square with a white plus sign and a blue square with a white minus sign. The rest of the interface is mostly empty and light gray.

The following table describes the fields in the Languages tab.

Field	Description
<b>Available</b>	The list of languages to be available for runtime presentation. Click the <b>Add</b> button to add a language using the <b>Select Language</b> dialog box:



Enter a search key to search by prefix, or use asterisk (\*) or question mark (?) for wildcard string or character match respectively, or simply click to select from the list of defined language codes. The language codes are in format of either:

- *aa-BB*: (for example, "en-US") where *aa* is the language and *BB* is the country or locale
- *aa*: (for example, en) where *aa* is the language. This is without locale denotation which is a more generic specification than *aa-BB*. It is equivalent to attaching -*xx* at the end. For example, "en" is equivalent to "en-xx".

See how the [browser language setting](#) would match against this list upon browser request for pages.

<b>Default</b>	The default language to serve Web application pages when the browser request does not specify a preferred language, or the language is not in the <b>Available</b> language list. Only languages in the <b>Available</b> language list may be chosen in the drop-down listbox here. For convenience, the English language <i>en-xx</i> is always included as a choice and as the default upon project creation.
----------------	---

## Metadata Root Aliases

The **Aliases** tab allows you to specify a string alias for a metadata element. When you load your metadata, Velocity Studio replaces all references by using an aliases map.

Original	Replacement	Source
cwf.Decimal10_2	cwf_up.Decimal10_2	ConceptWave Metadata
cwf.Integer3	cwf_up.Integer3	ConceptWave Metadata
cwf.Integer9	cwf_up.Integer9	ConceptWave Metadata
cwf.UserEmail	cwf_up.UserEmail (User Email)	ConceptWave Metadata
cwf.UsersForNode	cwf_up.UsersForNode	ConceptWave Metadata
cwf.nodeFinder	cwf_up.nodeFinder (Node Finder)	ConceptWave Metadata
cwf.userActionFinder	cwf_up.userActionFinder (User Actions)	ConceptWave Metadata
cwf.userFinder	cwf_up.LoginUserFinder (Logged Users)	ConceptWave Metadata
cwf.userLogDocument	cwf_up.userLogDocument	ConceptWave Metadata

At runtime, any script referring to the string is deferred to the metadata element. This string must take the *namespace\_name.objectname* format.

**Note:** If the string is same as the full name of any existing metadata element, an error occurs in either design or runtime.

To use the aliases function, perform the following steps:

1. From the Aliases tab, click the **Add** button.

Original	Replacement	Source
cwf.Decimal10_2	cwf_up.Decimal10_2	ConceptWave Metadata
cwf.Integer3	cwf_up.Integer3	ConceptWave Metadata
cwf.Integer9	cwf_up.Integer9	ConceptWave Metadata
cwf.UserEmail	cwf_up.UserEmail (User Email)	ConceptWave Metadata
cwf.UsersForNode	cwf_up.UsersForNode	ConceptWave Metadata
cwf.nodeFinder	cwf_up.nodeFinder (Node Finder)	ConceptWave Metadata
cwf.userActionFinder	cwf_up.userActionFinder (User Actions)	ConceptWave Metadata
cwf.userFinder	cwf_up.LoginUserFinder (Logged Users)	ConceptWave Metadata
cwf.userLogDocument	cwf_up.userLogDocument	ConceptWave Metadata

2. Double-click the **Original** field and specify the replacement element's alias. Press the **Enter** key to continue.

Original	Replacement	Source
cwf.Decimal10_2	cwf_up.Decimal10_2	ConceptWave Metadata
cwf.Integer3	cwf_up.Integer3	ConceptWave Metadata
cwf.Integer9	cwf_up.Integer9	ConceptWave Metadata
cwf.UserEmail	cwf_up.UserEmail (User Email)	ConceptWave Metadata
cwf.UsersForNode	cwf_up.UsersForNode	ConceptWave Metadata
cwf.nodeFinder	cwf_up.nodeFinder (Node Finder)	ConceptWave Metadata
cwf.userActionFinder	cwf_up.userActionFinder (User Actions)	ConceptWave Metadata
cwf.userFinder	cwf_up.LoginUserFinder (Logged Users)	ConceptWave Metadata
cwf.userLogDocument	cwf_up.userLogDocument	ConceptWave Metadata
cwf.fileAttachment	<NONE>	<local>

3. Click the **Replacement** field of the new row that you have added and select the element that you want from the list. Alternatively, you can click the  Select Element button and enter the object name that you want in the field provided.

Original	Replacement	Source
cwf.Decimal10_2	cwf_up.Decimal10_2	ConceptWave Metadata
cwf.Integer3	cwf_up.Integer3	ConceptWave Metadata
cwf.Integer9	cwf_up.Integer9	ConceptWave Metadata
cwf.UserEmail	cwf_up.UserEmail (User Email)	ConceptWave Metadata
cwf.UsersForNode	cwf_up.UsersForNode	ConceptWave Metadata
cwf.nodeFinder	cwf_up.nodeFinder (Node Finder)	ConceptWave Metadata
cwf.userActionFinder	cwf_up.userActionFinder (User Actions)	ConceptWave Metadata
cwf.userFinder	cwf_up.LoginUserFinder (Logged Users)	ConceptWave Metadata
cwf.userLogDocument	cwf_up.userLogDocument	ConceptWave Metadata
cwf.fileAttachment	<NONE>	<local>

4. When you save and reload your metadata, all original elements in your script are replaced with the replacement elements using this aliases map.

#### Delete an Alias

To delete an alias from the Aliases tab, highlight the alias that you want and then click the **Delete** button.

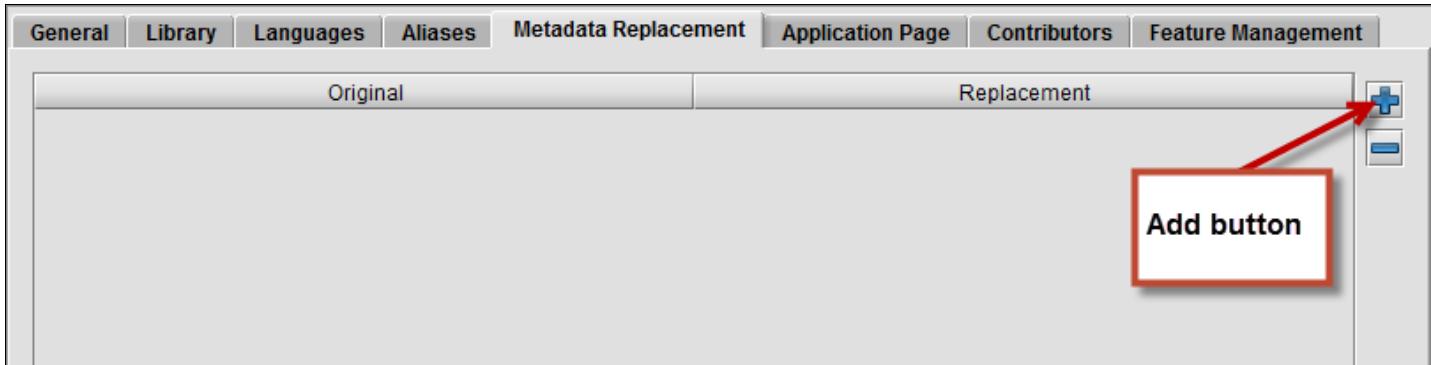
Original	Replacement	Source
cwf.Decimal10_2	cwf_up.Decimal10_2	ConceptWave Metadata
cwf.Integer3	cwf_up.Integer3	ConceptWave Metadata
cwf.Integer9	cwf_up.Integer9	ConceptWave Metadata
cwf.UserEmail	cwf_up.UserEmail (User Email)	ConceptWave Metadata
cwf.UsersForNode	cwf_up.UsersForNode	ConceptWave Metadata
cwf.nodeFinder	cwf_up.nodeFinder (Node Finder)	ConceptWave Metadata
cwf.userActionFinder	cwf_up.userActionFinder (User Actions)	ConceptWave Metadata
cwf.userFinder	cwf_up.LoginUserFinder (Logged Users)	ConceptWave Metadata
cwf.userLogDocument	cwf_up.userLogDocument	ConceptWave Metadata

Delete  
button

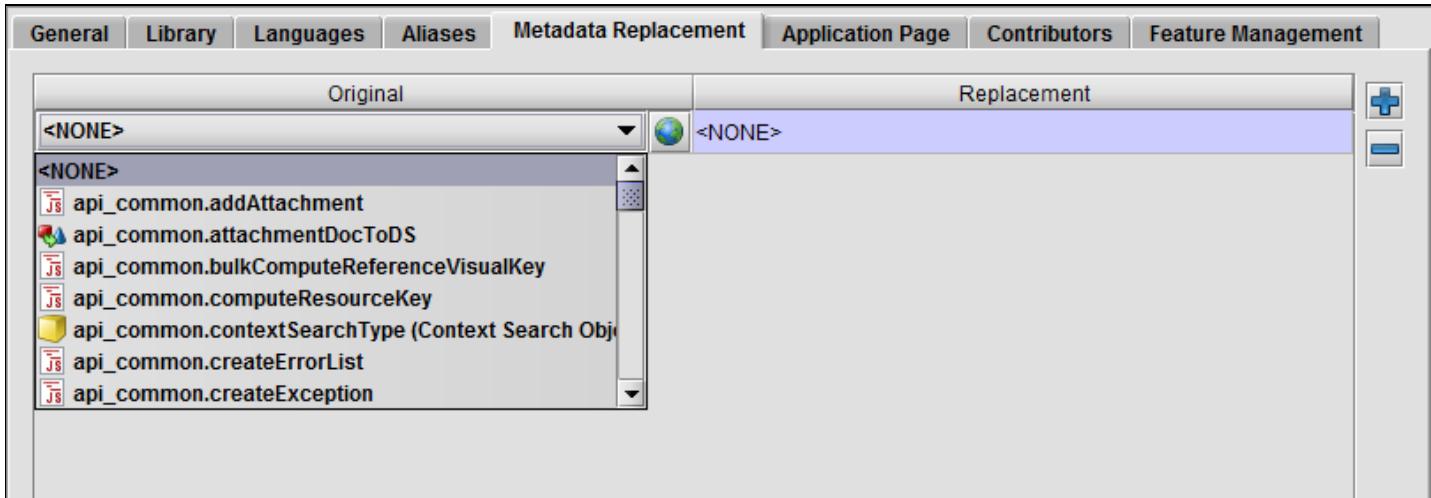
## Metadata Root Metadata Replacement

The **Metadata Replacement** tab allows you to replace a metadata object in your project with another one. To use this function, perform the following steps:

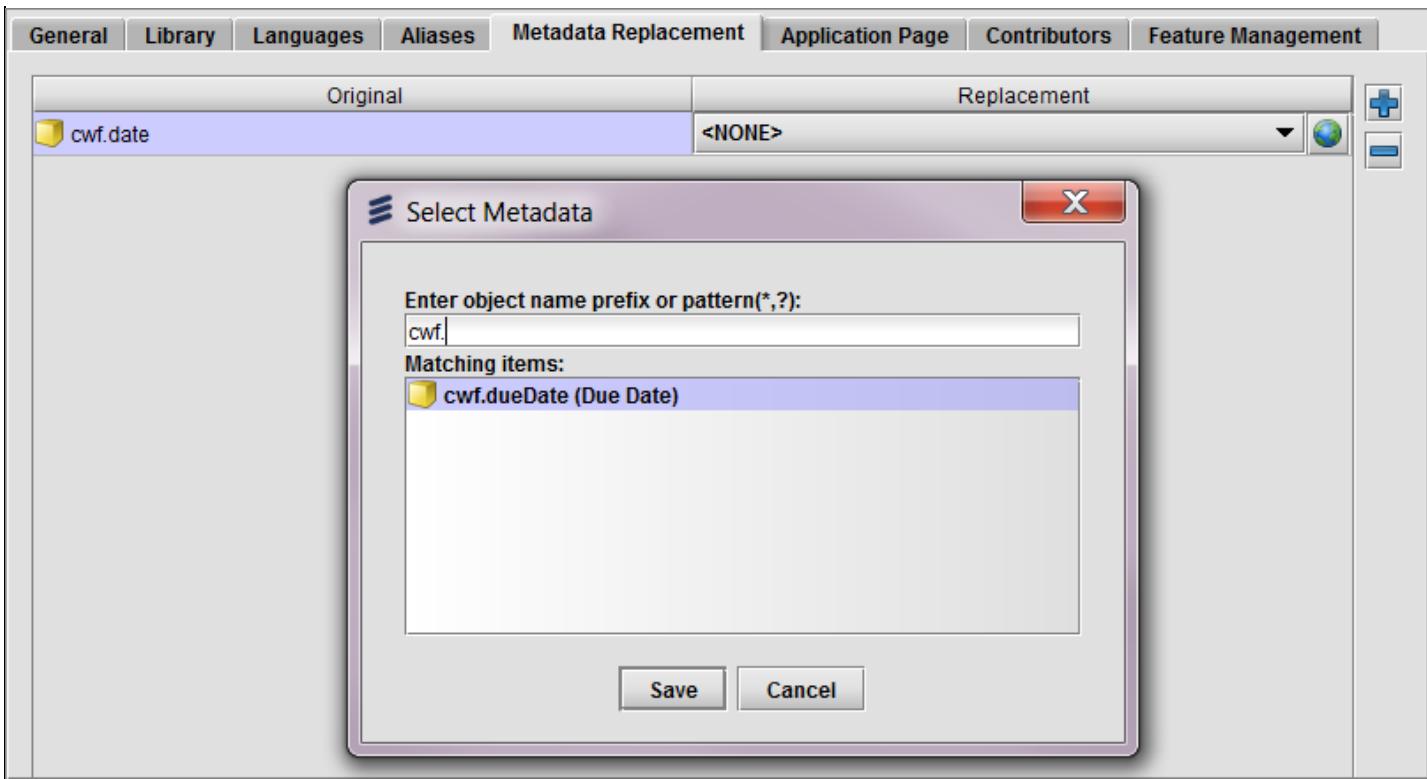
1. From the Metadata Replacement tab, click the **Add** button.



2. Click the **Element** field of the new row that you have added and select the element that you want from the list. Alternatively, you can click the **Select Element** button ( ) and enter the object name that you want in the field provided.



3. Repeat the previous step, but for the **Replacement** field, to specify the element that will replace the one that you specified in the **Original** field.



4. If you have more elements that you want to add, repeat the first three steps of this procedure.
5. When you have finished adding your elements and their corresponding replacement elements, click the **OK** button. The elements that you have specified to be replaced have been applied to your metadata.

## Metadata Root Application Page

The **Application Page** tab automatically populates with all application metadata objects, including templates. It also contains properties to customize your experience using the Select Application page.

Show	Application	Security	URL	Login UI	Group	Source
Yes (<Default>)	cwf_pm.WorkflowAdministration		/WorkflowAdministration		Security	ConceptWave Metadat...
Yes (<Default>)	cwt_sof.SUSMenu (Old App Menu)		/SUSMenu			Service Orchestration ...
Yes (<Default>)	applicationUI.CRMLite (CRM Lite)		/testApp		General	<Local>
Yes (<Default>)	cwa_worklist.worklistManagement (Worklis...)		/worklistMgr		Security	CWA - Worklist Manag...
Yes (<Default>)	cwa_config.systemConfiguration (System ...)		/configApp		System	CWA - Config
Yes (<Default>)	cwa_admin.systemAdministration (System ...)		/systemAdministration...		System	CWA - System Admini...
Yes (<Default>)	cwt_pc.catalogMain (Catalog Management ...)		/catalogMain5 (<Defau...		Other	ConceptWave Metadata
No (<Default>)	ui_common.ApplicationPage		/selectApp (<Default>)			UI - Common
No (<Default>)	ui_common.baseApplication		/baseApplication (<De...			UI - Common
No (<Default>)	ui_common.loginUI		/loginUI (<Default>)			UI - Common
No (<Default>)	ui_common.logoutUI		/logoutUI (<Default>)			UI - Common
Yes (<Default>)	ui_common.sessionErrorUI		/session_error (<Defa...			UI - Common
Yes (<Default>)	cwt_pcim.LifeCycleManagement		/LifeCycleManagemen...			ConceptWave Metadata
Yes (<Default>)	cwt_pcim.plmConfiguration		/plmConfiguration (<D...			ConceptWave Metadata
Yes (<Default>)	cwt_sof.softMenu (Service Orchestration)		/softMenu (<Default>)			Service Orchestration ...
Yes (<Default>)	cwa_security.securityManagement (User Pr...		/securityManagement ...			CWA - Security
Yes (<Default>)	cwt_pc.catalogMainApp (Catalog Managem...		/catalogMain (<Default...			ConceptWave Metadata

The page has the following properties:

Field	Description
<b>Group Source</b>	This field defaults to null and works with the <b>Group</b> field located in the table area. To select a group source, you can either click the drop-down menu and select a data type with enumerations as your group from the list, or click the <b>Globe</b> button to select your metadata data type with enumerations as your group object from the list. Each enumeration appears as a header that separate the available applications into groups.
<b>Show Application Page before login</b>	<p>This property's checkbox is unchecked by default. Selecting this property indicates that the Application Page appears first. In other words, the <b>Show Application Page before Login</b> property determines whether the /login or the /selectApp URL will be directed to when the root / URL is accessed.</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>If you have the <b>Classic Select Application</b> property selected, the login screen appears first, regardless of whether you have selected the <b>Show Application Page before login</b> property.</li> <li>If the Application Page table has more than one security configuration name (see the Security field in the following table), this property is selected and editing is disabled.</li> </ul> <p>When using the icon version of the Select Application with this property selected, the application icons appear with lock icons. Clicking an application icon shows the login screen, prompting you to enter your username and password. Click the <b>Login</b> button to log on to the application. Otherwise, clicking the <b>Cancel</b> button returns you to the Select Application page.</p>
<b>Classic Select Application</b>	Select this property to use the classic Select Application screen at runtime. Otherwise, leave this property unchecked to use the icon version of the Select Application screen.

This page contains the following fields:

Field	Description
-------	-------------

<b>Show</b>	This field indicates whether to show the application on the Application Page at runtime and allows you to override the <b>Not Valid for Application Selection</b> property. This field takes one of the following values: <ul style="list-style-type: none"> <li>• &lt;Default&gt; means that the application's <b>Not Valid for Application Selection</b> property is used to determine whether it displays</li> <li>• <b>Yes</b> indicates that the application page displays</li> <li>• <b>No</b> means that the application does not display</li> </ul> To change the value of this field, click the drop-down menu and select the value that you want from the list.
<b>Application</b>	This field contains the application's element name or label, depending on your Velocity Studio display preferences. You cannot edit this field.
<b>Security</b>	<p>This field denotes the security configuration name to be used for the application. The field's default value is empty, which indicates that the default security provider is used for the application.</p> <p>To change select a security provider the value of this field, click the drop-down menu and select a value from the list.</p> <p><b>Note:</b> Editing this field requires you to connect to the database, to retrieve the list of possible security providers.</p>
<b>URL</b>	<p>This field indicates the URL property to directly access each application.</p> <p>See the <i>URL Mapping</i> section that follows for more information. When &lt;Default&gt; appears in this field, it indicates that the application's <b>Url Mapping</b> field value is used. By specifying a different value in this field means that you are overriding the application's <b>Url Mapping</b> property.</p>
<b>Login UI</b>	<p>This field allows the selection of a UI to be used for the login page. This UI displays once you have selected an application and you are accessing the <i>login</i> URL.</p> <p>By default, if no UI is selected, the login UI specified under the selected provider is used. If there is no login UI specified under the selected provider or if no security provider is selected, the ui_common.loginUI is automatically used as the login UI. To specify a login UI, click the drop-down menu and select the login page that you want to use.</p>
<b>Group</b>	<p>This field denotes the application group specified from the <b>Group Source</b> field, which shows a list of enumerated values. The application is grouped under its <b>Group</b> in the Application Page. Click the drop-down menu and select the group from the list provided. This field supports internationalization.</p> <p><b>Note:</b> If you do not specify a group, the application appears in the General group.</p>
<b>Source</b>	<p>This field indicates the template (library) source of the application. This field is read-only. If the application originates from user project-specific metadata, &lt;Local&gt; appears in this field. This field is not editable.</p>

## URL Mapping

You can expose (top-level) User Interface objects to Web clients by assigning them to a relative URL path, relative to

`http://<hostname>:<port>/cwf`

For example, if your project has an application page (say, for checking order status) with URL mapping `/orderstatus`, the page can be directly accessed by users with the URL `http://<hostname>:<port>/cwf/orderstatus`. At runtime, the User Interface (controller) is invoked to render according to the UI object's **Page** when a Web browser sends a request with this relative URL path.

See [Modifying URL Mappings and Style Sheets](#) for more information about URL mapping configuration.

Each top-level application controller has the **URL Mapping** string property.

The screenshot shows the AVM Metadata interface with the General tab selected. On the left, there's a tree view of metadata objects. In the center, the CRM Lite application controller is being edited. The URL Mapping field is highlighted with a red box and contains the value /CRMLite.

This property's string value appears in the **URL** field on the Application Page's table.

The following rules apply to the **URL Mapping** property:

- This property is mandatory and must be unique within the entire application.

**Note:** Any migrated 5.x application without a mapped URL automatically receives a URL set to the application's name.

- If more than one application has the same URL, you must resolve this conflict in your metadata project through the metadata header's Application Page tab.
- The URL is case-insensitive. The URL may not be the same as a list of reserved system URLs or match the wildcard pattern of system URLs. This means for a given system URL /v, no URL preceded by /v/ can be used. The list of reserved system URLs is the following:

- /v
- /r
- /t
- /cd
- /design
- /a
- /c
- /login

By default, many system-defined User Interface objects exist in the mapping, and they can neither be removed, nor edited. They are informational mappings that let you know which relative URLs are exposed in your Web application.

For your project's User Interface objects, you can assign multiple relative URL paths to the same User Interface object. As an example, the system login page is accessible by both / and /login, as set in default setting.

Only top-level User Interfaces are allowed to be URL mapped. Included User Interfaces, such as the User Interface of a finder, are not allowed to be mapped.

### Not Valid for Application Selection Property

Each application controller has the **Not Valid for Application Selection** Boolean property.

General Variables Methods

**Name:** CRMLite

**Label:** CRM Lite

**Description:**

**Extends:** com.conceptwave.system.Application

**Overrides:** <NONE>

**Help:**

**Worklist:** cwf\_pm.BaseWorklistFinder (Worklist)

**Timer duration (minutes):** 0 (<Inherited>)

**Url Mapping:** /CRMLite

Clear validation messages on editing    Show Errors Inline  
 Not Valid for Application Selection    Block UI on page update from server

By default, this property is false. When you select this property, the application is not visible on the Select Application page. However, direct URL access to the application through its **URL Mapping** property is allowed at runtime.

## Application Style Property

The Page overlay of each application contains the **Application Style** property, which specifies the style to be used for the icon representing the application on the icon version of the Select Application page.

Customers Orders Logout

**Page**

**Name:** Page

**Extends:** com.conceptwave.system.Application.Forms.Page

**Title:**

**Visible:** <NONE>

**Icon:** <NONE>

**Skin:** <NONE>

**Stylesheet:** /cwf/css/cwf.css

**Body Style:** <NONE>

**Application Style:** <NONE>

Dialog opacity (0-100):

By default, if no style is provided, the system automatically attempts to find a appPage<applicationName>Label (for example, appPageCustomerLabel, where Customer is the name of the application).

To use the default styling, follow this style:

```
.cwAppPageLabel, .cwAppPageLabelOver, .cwAppPageLabelDown, .cwAppPageLabelSelected, .cwAppPageLabelSelectedOver, .cwAppPageLabelSelectedDown {  
    word-wrap : break-word;  
    background-color: #009999;  
    background-image: url("../images/sample_icon.png");  
    background-position: 60px 70px;  
    -webkit-box-shadow: 2px 2px 2px 2px rgba(0, 0, 0, 0.8);  
    box-shadow: 2px 2px 2px 2px rgba(0, 0, 0, 0.8);  
    background-repeat: no-repeat;  
    color: #FFFFFF;  
    padding-left: 5px;  
    padding-right: 25px;  
    padding-top:10px;  
    font-family: Arial,Helvetica,sans-serif;  
    font-size: 12px;  
    height: 110px;  
}
```

```
width: 70px;  
vertical-align: top;  
text-align:left;  
}
```

## Select Application Page: Classic and Icon

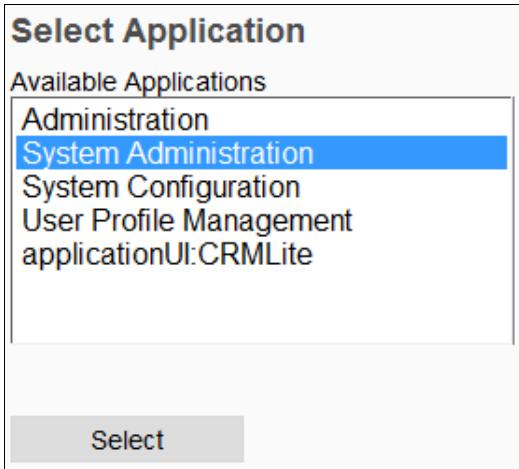
The Select Application page comes in two varieties:

- [Classic](#)
- [Icon](#)

The [Classic Select Application](#) property allows you to choose which version of the Select Application page that you want to view and use.

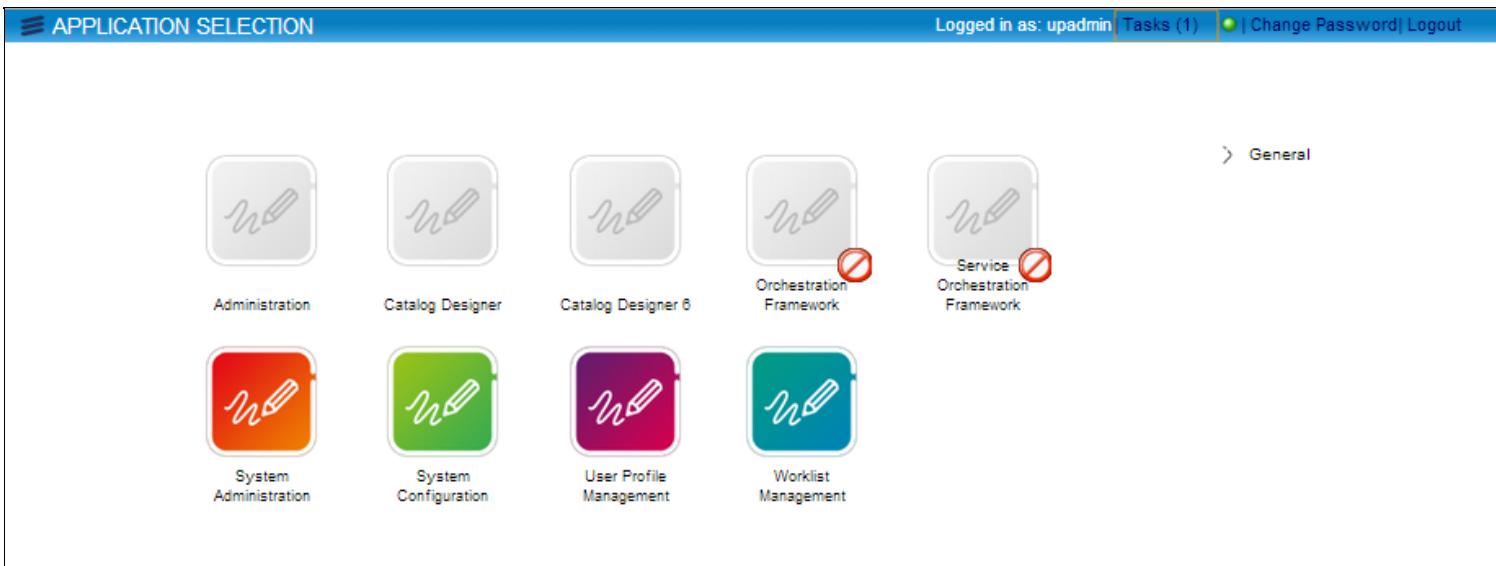
### Classic Select Application Page

The classic version of the Select Application page lists applications that you have access to. To open an application, select the application that you want to open from the list and click the **Select** button.



### Icon Select Application Page

The icon version of the Select Application page shows each icon as an application.



Any application in your metadata that is not specifically told not to show itself through the application's [Not Valid for Application Selection](#) property or through the metadata header's Application Page tab table's [Show](#) property appears on the Select Application icon page.

The icons appear in groups defined by the **Group** property on the metadata header's Application Page tab of the metadata header. The application's Page overlay's Application Style property determines the icon displayed for each application. The icon's label comes from the application's **Label** Page form property.

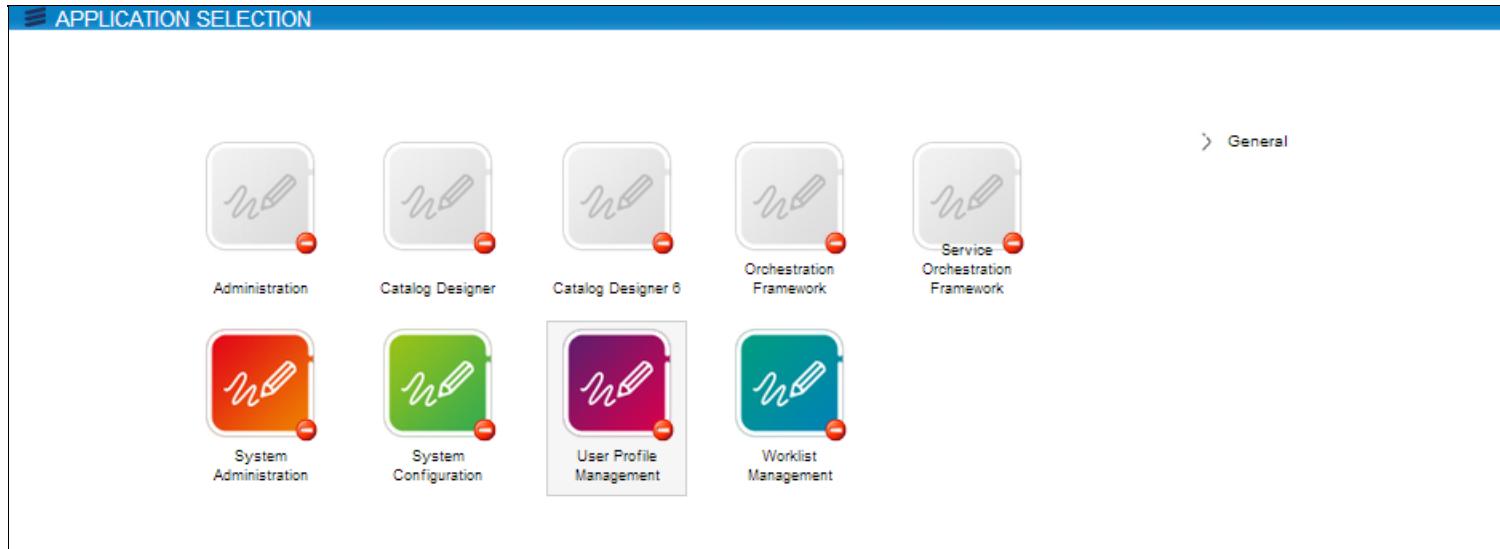
The top right side of each application icon contains a smaller icon that represents the current permission status:

- No icon indicates that the user currently logged in has the correct security provider
- No entry sign indicates that the user has the right security provider, but the wrong permissions
- A lock icon indicates that either the user does not have the right security provider or is not logged in

The following actions occur when clicking an application icon:

- When you click an application with no icon, the application launches
- When you click an application with either a no entry or a locked icon, you remain on the Select Application icon page.
- If you have selected the **Show Application Page before Login** property, and you click an application with either a no entry or a locked icon, the application's login page appears.
- If you enter incorrect login information for the security provider, an error message appears on the login page.
- If you enter your login information correctly for the security provider, but your login information does not have the proper permission for the selected application, the Select Application page appears and an error message displays.

**Note:** When your metadata project has the **Show Application Page before Login** property selected, at runtime, all icons on the Select Application page have smaller lock icons.



Clicking the application that you want to access launches the application's login page.

## Metadata Root Feature Management

The **Feature Management** tab allows you to specify a set of features in your metadata project's header. For each [global script](#), [event handler](#), and [permission](#) method, you can select a subset of features from a combined list of features defined in the project header and all dependent library headers.

By default, if a global script, event handler, or permission method does not specify any feature, it is assumed to always be accessible. For the specified subset of features, if your user [licence](#) contains any one of the features in the subset, it is assumed that you have access to the global script, the event handler is active, and the permission method is set to true.

At runtime, when a global script is invoked and your user licence does not entitle access to any of the features required for the global script, an exception occurs. For event handlers, no access to the necessary feature means that the event handler does not handle the event being published. For permission methods, it returns false if your user licence does not support any of the features required by the permission method.

### Notes:

- This feature does not deal with enabling or disabling features available in Velocity Studio, or loading metadata template libraries.
- This feature does not handle licence management or feature information management in the licences.

To use this function, perform the following steps:

1. From the Feature Management tab, click the **Add** button.

Features:	Name	Description

Restricted By:

2. Click the **Name** field of the new row that you have added and enter the feature's name. The name must match the feature name registered with Sentinel.
3. Click the **Description** field and enter a summary of what the feature entails and how it is to be used.
4. For the **Restricted By** field, click the field's **Checkbox** button () to launch the Select Feature dialog. You can select all features that you want to restrict and then click the **Save** button.
5. Save your metadata project.

## AVM Licencing with Sentinel RMS

Ericsson products use the Sentinel server-client approach, which the Ericsson licencing system supports. **Sentinel RMS** is a robust licence enforcement and enablement solution that allows you to control and see how you deploy and use your applications.

The client has a licenced application. To use the application, the client must contact the licence manager server and have it fetch the licence file. As a result, your environment must have a Sentinel licence manager server running in your environment. This server much be installed either in your network or on the same machine where the application is running. In the extreme case, the Sentinel licence manager server can be a public Web service available through the Internet.

Customer orders created by SAP flow to the Ericsson licencing system where it generates a licence file based on the order details. The Ericsson licensing system provides a user interface for you to maintain your licences. It also keeps a list of fingerprints for each licence. All fingerprints must be collected upfront, so they can be inserted in the licence file.

A licence file that the Ericsson licensing system generates contains [value packs and extra features codes](#) defined in PRIM (that is, the database used by SAP). No feature hierarchy is provided. Applications must recognize and map value packs to actual feature names used by both AVM and metadata API.

The following licence types are supported by Ericsson Order Care and Ericsson Catalog Manager:

- System – a server licence that a machine fingerprint locks
- User – a licence locked to a user by a client machine fingerprint
- Floating – locked by a licence server fingerprint with  $n$  number of subscribers
- Capacity – a feature may have a capacity and issues a warning when the maximum item count is achieved

### Notes:

- The following licence types are unsupported:
  - Named user licence, which is locked by the username.
  - Concurrent user, which is locked to a maximum number of open sessions within the application.
  - Standalone licences, when all licencing is self-contained on the individual computer on which the licenced application is installed. However, it is possible to have the licence server run as a separate service on the same machine.
- This implementation does not use the Execution Environment API for licencing when running in this environment. A special licence handler is to be implemented once the Execution Environment's cloud API is available.

This section covers the following topics:

- [Installation and packaging](#)
- [Licence handlers](#)
- [Value pack and feature codes mapping](#)
- [Support for the existing licence model](#)
- [Licencing scriptable APIs](#)

## **Installation and Packaging**

When you install the product through the wizard, the [Check Licence](#) page prompts whether you want to use a licence file or the Sentinel licence server address. If you select the Sentinel server type and a connection to the server is successfully established, the product checks out the pertinent licence information to local persistence storage and the server continues working offline. When you open the Velocity Studio application and start the AVM, the application uses the information from this persistence storage.

The Sentinel server address is saved in the properties file in the Designer installation directory. You can add the server IP address and change it by clicking [Help > About Velocity Studio - License](#) tab from the menu bar.

## **Supported Operating Systems**

The Sentinel licence manager supports the following operating systems:

- Windows 32- and 64-bit versions
- Windows XP
- Windows Vista
- Windows 7
- Windows 8 (Desktop mode)
- Windows Server 2003, 2008, 2008 R2 (64-bit only), and 2012
- Solaris Sparc
- Solaris x86
- Linux
- Macintosh

## **AVM under a Web Container**

The WAR file that is used to deploy the application under a Web container includes DDLs and SO (shared object) files that the Sentinel API library requires. At startup, it extracts the libraries to the user home directory.

The IP address to the server is defined as a Java system option. Alternatively, it may look for a local properties file with the licence server information. If the AVM is unable to connect to the licence server, the AVM does not start and logs the error.

## **SUF Installation**

The SUF installation prompts for the licence type (that is, either the licence file or the licence server). If you specify the latter, it prompts for the IP address. A special playlist command then runs, and checks whether the server is available and the licence can be obtained. The IP address is then saved as Java options in the JBoss standalone configuration file.

## **Upgrade Licences**

Upgrading licences is currently unsupported, but will be supported in future releases. To use additional licences, you must install additional licence servers.

## Licence Handlers

The AVM provides the following licence handlers, which are also known as connectors:

- [Legacy handler](#)
- [Sentinel connector under a Web container](#)
- [Sentinel offline repository handler for Velocity Studio](#)
- [Testing handler for feature developer licence](#)

### Legacy handler

An adapter is available to support the existing file format and existing module-based licencing.

### Sentinel Connector under a Web Container

This licence handler connects to the Sentinel license manager server. It is assumed that the connection to the server must be available at startup, and when the floating licence is used or licence synchronization is requested.

If the AVM is using a floating licence for itself or for a feature (for example, floating license for  $n$  of AVM instances), it requires using a special request or release API. The request is called with a specified timeout parameter. The licence defines this timeout value, which can be set from 1 to 900 minutes.

The AVM heartbeat thread needs to call the update API to continue using the licence. Otherwise, the licence is released when the timeout occurs. If the connection to the server is lost, or at runtime, or the licence has expired, it starts logging errors every update period.

### Sentinel Offline Repository Handler for Velocity Studio

You should have the ability to work offline. A special type of license, called a repository, is used. The licence generation defines the type of licence. A special Sentinel utility tool is called during the installation to initialize the local licence storage (that is, licence cache) and check out the licence from the server.

**Note:** The Velocity Studio licence handler *does not* support floating licence behaviour for features when running the application from Velocity Studio. API calls to request or release features are ignored. To test the floating licence, it is recommended that the application be deployed on a Web container and use the floating network licence type.

### Testing Handler for Feature Developer Licence

If your [Velocity Studio licence](#) has a special DeveloperFeature feature code in the licence, you can open any protected template and start AVM using the special testing handler. When starting AVM from Velocity Studio, the application creates a special runtime handler that substitutes the actual licence handler at runtime. All licence API calls go to the test handler.

You can find the features that are currently defined and activated for testing in [Preferences > Runtime Test Features](#). You can turn these features on and off on this page.

## Value Pack and Feature Codes Mapping

A special .ini file contains mapping between the value pack and features code, to features names that both the AVM and metadata use. Codes may change from one product version to another.

The following is a sample .ini file showing its file format:

```
FAT1010304/4=AVM  
FAT1010303/12=CatalogDesigner  
FAT1010303/5=VelocityStudio  
...
```

To protect the file from being changed, its contents require being encrypted.

**Note:** The .ini file with the mapping is packaged with the product installation (that is, either with Install Anywhere within the product wizard installation or System UF).

### Edit the .ini File

In Velocity Studio, there is an additional menu that allows you to create a new .ini file, and open and save an existing .ini file with codes mapping. This part of the user interface contains a table with key-value pairs. When saving the document, it automatically gets signed.

**Note:** To see and use this Velocity Studio menu, you must have the FeatureDeveloper licence.

## Support for Existing Licence Model

The licence model implemented with Sentinel RMS fully supports the existing licencing approach for these areas:

- Fingerprinted by machine
- Module-based licence
- Development licence

### Fingerprinted by Machine

Currently, the AVM supports its own licence file format, which contains a set of modules (that is, features). The file is fingerprinted with the MAC address so that the licence is generated for each machine or server. The capacity is unsupported. The number of subscribers is controlled manually by limiting the number of licence files to  $n$ .

### Module-based Licence

Both Ericsson Order Care and Ericsson Catalog Manager require licences for different product types (for example, Product Catalog, Service Registry, and so on). To install the product and start the application, you must have a valid licence file (that is, the base AVM licence at a minimum).

### Development Licence

The Velocity Studio development tool requires a valid licence file with the VelocityStudio feature included. When installing Velocity Studio, it checks that the licence is valid, and then copies only the licenced modules. To run the application through Velocity Studio or a command line, you need a valid licence for all modules to be used.

### Special Cases with Licences

There are two special cases when it comes to licences:

- [Your system running Velocity Studio fails or you forget to return a key](#)
- [You add incremental Velocity Studio users](#)

**Scenario 1:** Your system running Velocity Studio fails or you forget to return a key.

If your system that has Velocity Studio fails or if you forget to return a key, follow these steps:

1. Have all Velocity Studio users return their keys.
2. Run a cleanup script on the licence server.
3. Send a new fingerprint to Sweden, asking for a new file.
4. Install this file on the licence server. Velocity Studio users are down at this stage.

**Scenario 2:** Add incremental Velocity Studio users

If you need to add more Velocity Studio users (for example, you buy five licences and, at a later date, you add five more), do the following:

1. Have all users return their Velocity Studio keys.
2. Delete the Velocity Studio component from the licence file.
3. Add the new licence file (five + five licences) to the licence server.

**Note:** This step can be done in a maintenance window, provided that Sweden receives this new file.

## Licencing Scriptable APIs

The License scriptable class includes the following methods to support AVM licencing with Sentinel RMS:

Method	Description
Boolean isFeatureActive(String feature);	This method returns true if the feature is currently activated.
String[] getActiveFeatures();	This method returns all features currently activated.
String[] getAllFeatures();	This method returns a list of all features defined in the system.
Date getStartDate(String feature);	This method returns the start date of the feature, including the grace period.
Date getExpirationDate(String feature);	This method returns the expiration date of the feature, including the grace period.

The following table describes capacity and usage methods:

Method	Description
Integer getCapacity(String feature);	This method returns the capacity value for the given feature or null if the feature does not have a capacity.
Boolean requestUsage(String feature, String userID, Integer capacity);	You can use this method to support floating capacity licences. It requests feature usage for the userID (can be null, meaning that the capacity is requested for each AVM instance) and with a specified capacity (1, if not specified).  If this method calls requestUsage() multiple times, the counter gets summarized. This method returns true if the requested usage is granted.
Boolean releaseUsage(String feature, String userID, Integer capacity);	This method releases the feature capacity. The userID can be null, meaning that the AVM instance is considered the user. Otherwise, the capacity value is equal to 1 if not specified.  The releaseUsage() method can be called if its capacity is less than requestUsage(). Then, it is allowed to call releaseUsage() later to release the rest of the capacity. This method returns true if it is unable to release the capacity.
void reportUsage(String feature, String userID, Integer capacity);	This method reports feature usage (that is, the current capacity).  <b>Note:</b> This method is expected to be implemented for the Cloud API to report the feature's current usage.  Using this method is the equivalent to calling the following sequence:  requestUsage(feature, userID, capacity); releaseUsage(feature, userID, capacity);

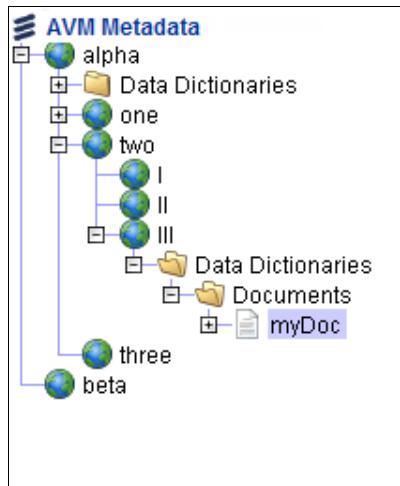
For more information about these methods, consult the [JavaScripting documentation](#).

## Namespaces

Application metadata in a project is organized into Namespaces. The namespace object organizes its metadata objects by displaying them in a static tree, with each sub-folder in the tree to contain a type of metadata object.

The namespace is the first item you create under the metadata root element. There is no restriction on the number and types of the metadata objects created within the namespace. Objects in one namespace may refer to objects in other namespaces.

Namespaces can be created within another namespace as well, in effect, creating a namespace hierarchy. There is no restriction on the depth of the hierarchy. For example, in the diagram, below, Namespace *alpha* contains child namespaces *one*, *two* and *three*. Sub-namespace *two* has further child namespaces *I*, *II* and *III*. In this example, the Document object *myDoc* in namespace *III* can be addressed by its full pathname, *alpha.two.III.myDoc*.



For easy navigation of the project, it is a good practice to have different namespaces to contain metadata of different aspects of the application. For example, you may have a namespace for each external interface-related metadata, or for each functional component/module of your application.

**Note:** All system namespaces are prefixed with *cwf*. User namespaces should not begin with this prefix.

### Create New Namespace

To create a new Namespace, do the following:

1. In the **Metadata** tab of **Navigation** pane, right-click a namespace or the on Metadata Root and select **New Namespace**.
2. The **New Namespace** wizard appears. Enter the **Name** of your namespace (must conform to JavaScript naming conventions), and fill in the other fields, as necessary.
3. Click the **Finish** button to create your namespace.

After the namespace is created, the namespace node appears under the node in the **Metadata** tab of the navigation pane.

### Namespace Properties

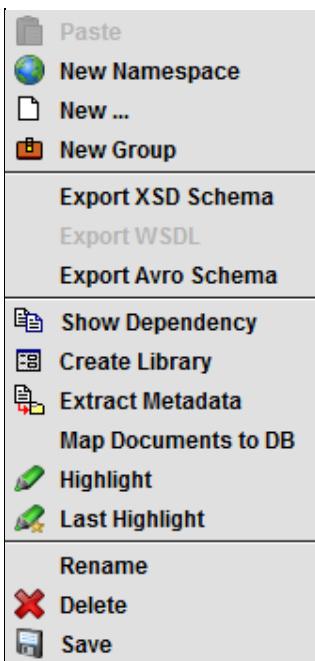
The namespace properties are listed in the table that follows.

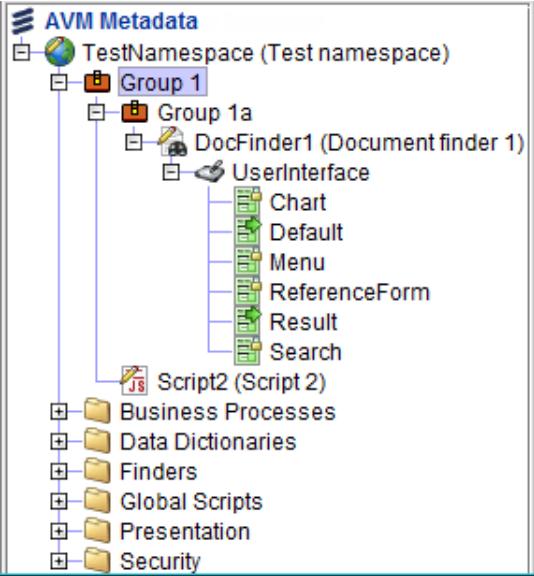
Name:	<input type="text" value="Alpha"/>	<input type="checkbox"/> Private	<input type="checkbox"/> Restricted	<input type="checkbox"/> Deprecated
Label:	<input type="text" value="Alpha"/>			
Description:	<input type="text"/>			
XSD Namespace:	<input type="text" value="http://www.conceptwave.com/alpha"/>			
License protection namespace:	<input type="text"/>			
<input type="checkbox"/> Skip DB schema generation <input type="checkbox"/> Qualified				

Field	Description
Name	Name of the Namespace (used in scripts; must conform to JavaScript naming conventions).
Private	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
Restricted	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
Deprecated	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
Label	<p>Namespace display label.</p> <p><b>Note:</b> If you do not specify a label for your namespace, the name of the namespace appears in the Configuration application, under the <b>Services &gt; Services</b> tab.</p>
Description	Description of the Namespace for documentation.
XSD Namespace	<p>Defines a unique URL for this Namespace, for the purpose of defining XSD Namespace when the namespace is exported (for example, <a href="http://www.conceptwave.com/template/oe">http://www.conceptwave.com/template/oe</a>). See the section that follows for more details.</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>Velocity Studio does not allow two different namespaces to have the same XSD name. A validation warning appears under such circumstances.</li> <li>If the <b>XSD Namespace</b> field is not defined and you are generating a WSDL, the namespace name is used as the target namespace URL instead.</li> </ul>
License protection namespace	This field is used in the metadata to identify each namespace used for <a href="#">namespace protection using partner licences</a> .
Skip DB schema generation	If selected, skips SQL generation for all Documents from this Namespace.
Qualified	<p>When this checkbox is checked, the generated XSD schema for this namespace sets the schema attribute <i>elementFormDefault</i> to <i>qualified</i>. This attribute determines whether all the children element or type in this namespace need to be namespace-qualified, meaning all elements or types in the XML instance should have be prefixed with its namespace.</p> <p>As an example, if you have a namespace, tvNamespace, that has the <b>Qualified</b> checkbox selected, the namespace is qualified in the generated XML. To set the XML name for the header child nodes (for example, Version is a child node), you need to include the namespace as a prefix (for example, tvNamespace:Version).</p> <p>To specify an attribute with a namespace qualification, create a data structure attribute of type &lt;Any&gt;. You can then set its value to the qualified attribute and the value (for example, ns1:attr1="xyz").</p>

## Namespace Operations

The figure below shows the right-click menu for a namespace. The table below contains descriptions of menu items that are unique to namespaces; description of common menu items can be found [here](#).



Menu Item	Description
<b>New Group</b>	<p>You can organize the metadata within your namespace by selecting <b>New Group</b> from the right-click menu to create a new group. Nested groups are allowed, as in the following example. Group 1 contains a script called Script2. Group 1 also has a subfolder, Group 1a, which contains a document finder called DocFinder1.</p>  <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• You can delete a group node by right-clicking it and selecting <b>Delete</b>. However, you cannot copy and paste a group node.</li> <li>• The <b>Copy</b>, <b>Cut</b>, and <b>Paste</b> functions are available for metadata under a group node by right-clicking your metadata and selecting the function that you want to perform.</li> </ul>
<b>Export XSD Schema</b>	Exports XSD schema for the namespace. The system will create an XSD by the same name as the namespace in the directory specified.
<b>Export WSDL</b>	Exports WSDL for the namespace. This command is only visible when the namespace has a SOAP interface. The system will create the WSDL in the directory specified. See below section for details.
<b>Export Avro Schema</b>	<p>Exports Avro schema for the namespace. The system creates Avro schema files (.avsc files) in the specified directory. The file-naming convention for these files is as follows:</p> <p><code>&lt;namespace&gt;.&lt;data structure name&gt;.avsc</code></p> <p>To export Avro schema files for <i>only</i> referenced data structures, go to <b>File &gt; Preferences &gt; Export</b> and select the <b>Export Referenced Types Only</b> checkbox. If this checkbox is <i>not</i> selected, schema files for other data structures are also exported. You can then select any elements that you want to hide, if needed.</p>

For Java namespace CIL serialization, the Avro schema filename conforms to these rules:

- If the Java namespace is specified in the data structure, the .avsc file is named as [java\_namespace].avsc.
- Note:** You must re-export avsc and restart Velocity Studio to have CIL working.
- If the Java namespace is not specified, the file is named as the metadata object's [fullpath].avsc.

Show Dependency	Creates a list of dependent namespaces. Select one or multiple namespaces. The system will loop through all the child metadata to find a list of namespace dependencies. It will only return namespace dependencies - not dependencies with library files. Refer to the <a href="#">Namespace Dependency Example</a> for more details.
Create Library	Creates a library file for the selected namespace. Once the user enters a name for the library file, the system will check for metadata dependencies across namespaces. If there are metadata dependencies with other namespaces, then the system will them and stop the library creation process. If there are no dependencies across namespaces, then the system will create a library JAR file for the selected namespace metadata and delete the namespace metadata. Refer to the <a href="#">Namespace Dependency Example</a> for more details.
Extract Metadata	Creates a copy of the selected namespace metadata and its dependent library files to a new project. The system will check the selected namespace for dependent metadata across namespaces. If a dependent namespace is found, then the metadata is not copied to the new project. If there are no dependent metadata with other namespaces (with the exception of the library files), then a new project is created for the selected metadata within the namespace. Refer to the <a href="#">Namespace Dependency Example</a> for more details.
Map Documents to DB	Creates DB mapping for Documents in the namespace. A Document must have a DB schema assigned to it to be affected by this command. It is equivalent to <a href="#">Map Documents to DB</a> in main menu command.
Highlight	You can change the way your namespace looks within the tree, including its style (for example, bold, italic) and font colour. The <b>Highlighting Settings</b> dialog appears, allowing you to change the look of your namespace, use the <b>Default</b> highlighting, or use the <b>Last Used</b> highlighting.
Last Highlight	Select this option to use the last highlighting style used on a metadata object within the tree.

#### Notes:

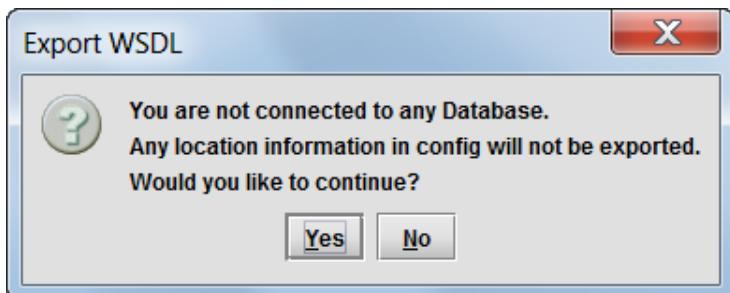
- For XSD and WSDL exports, some metadata documents will not be included as not all complex documents and types conform to the XSD and WSDL standards.
- Validation of generated XML directly against a schema is not available. Instead, the dataObject validate script API can be used to validate data content.

#### Export WSDL

You can export all [External Services](#) in a namespace whose ports are binded to SOAP Service Provider, with the **Export WSDL** right-click menu command. All Services and dependent objects such as Binding, PortType, Message, etc are generated into the WSDL file.

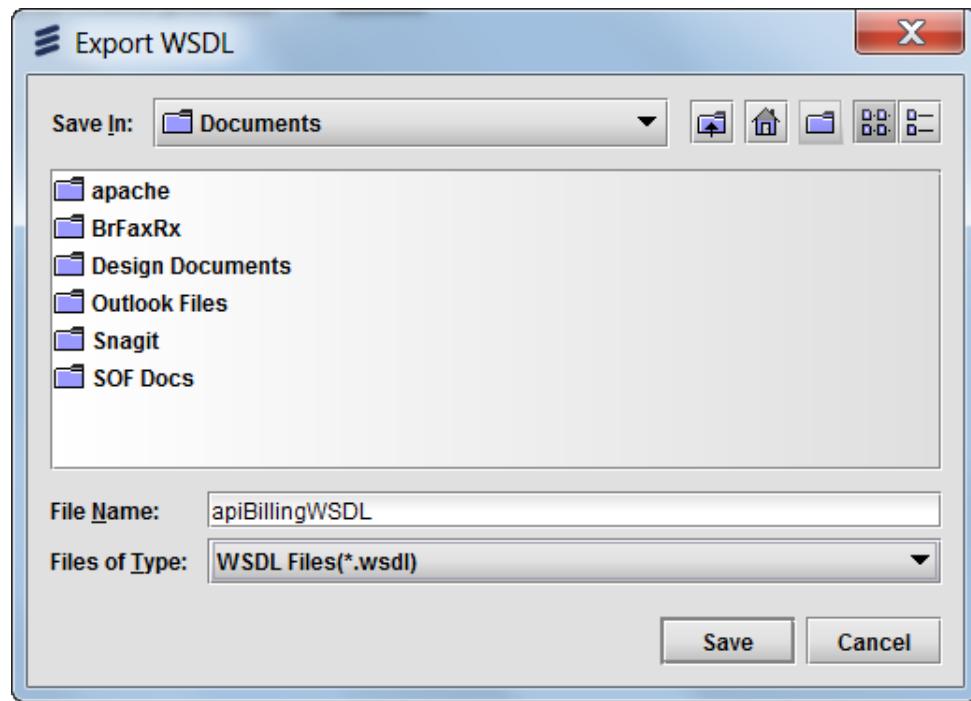
The availability of this menu command, and which SOAP service(s) is exported, depends on the currently selected node in the **Navigation Pane**. If the currently selected node is a metadata object that resides in a namespace that has a SOAP service (specifically, has a Service with at least one Port with a Binding that uses the SOAP service provider), this menu command becomes available. All SOAP services in the Namespace are exported in one WSDL file. Otherwise (for example, selected node has no SOAP service in the Namespace, or the selected node is an organizational folder node such as "Data Dictionaries", or no node is selected at all), the command is disabled.

You should connect to the Database before invoking this command, so that configurations set for the SOAP service in the Configuration application (for example, address information of Port) is properly exported as well. If the database is not connected when the menu command is invoked, the following prompt appears as a warning:



You can ignore the warning by selecting **Yes** to export the WSDL, or click **No** to cancel the operation.

The Export WSDL command opens the **Export WSDL** dialog. Specify the directory location to export the file as well as the filename. Click the **Save** button to export the file.



**Note:** you can choose whether to export schema within the WSDL or not, by selecting such setting in [File > Preferences](#) menu.

## Namespace Protection Using Partner Licences

---

A situation may arise in which you want to protect your namespace as intellectual property. In Velocity Studio, you can protect your namespace through licences.

To protect your namespace using partner licences, perform these steps:

1. Contact the Support team to advise that you want to protect your namespace. For each partner, the team generates a new private and public key pair:
  - o The public key is included in the product software
  - o The Support team ships the private key to you
2. Once you have both the private key and a release that contains the public key, you can then use namespace licences for your respective customers.
3. You must provide the Support team with a unique identifier. This identifier must be used on the **License protection namespace** line in the metadata for each namespace. The format is as follows:

```
com.unique identifier.NS1
```

4. Manually create the `<template_licence>.xml`. The following sample template licence .xml file is for two namespaces defined by:
  - o com.unique identifier.NS1
  - o com.unique identifier.NS2

In this example, NS1 and NS2 are names selected by you for your partner namespace.

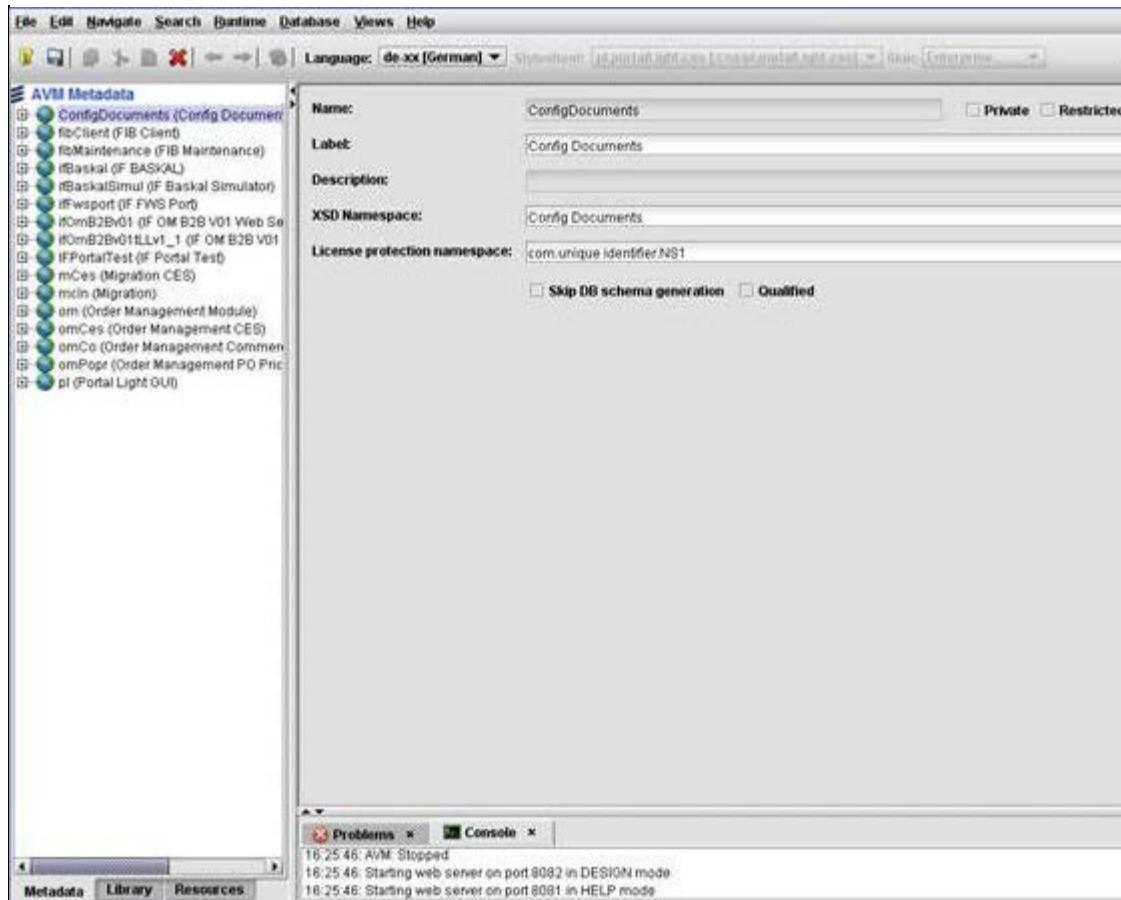
```
<?xml version="1.0" encoding="UTF-8"?>
<license>
    <comments>Partner name template licence</comments>
    <organization>
        <name>Partner Name</name>
        <address>Partner address</address>
    </organization>
    <namespaces>
        <namespace>com.unique identifier.NS1</namespace>
        <namespace>com.unique identifier.NS2</namespace>
    </namespaces>
</license>
```

5. Use the `signLicense.cmd` command, which is available in the product. This command creates the template licences for the end-customer. The syntax for this command is as follows:

```
signLicense.cmd [logfile] <template_licence>.xml privateKey
```

This command requires `<template_licence>.xml` and the private key as input. After running this command, the `<temp_licence>.lic` template licence file is created in the same folder where `<template_licence>.xml` resides. The `logFile` parameter allows you to specify the log file for a command line. It is an optional parameter. However, when used, it must be the first parameter.

6. The namespaces defined in the .xml is same as those defined in the metadata in the licence protection namespace. The following is a sample screenshot:



The licence-protected namespaces must always be prefixed with `com.unique.identifier`. The **License Protection namespace** field is available for namespaces only. When set, it forces runtime to check if this namespace is indeed included in the licence files.

7. After putting the template licence file into the Designer licence directory, you can use licence-protected namespaces for development. At runtime, in the application server, the namespace licence file must reside in same place as the product license.lic file.

## Namespace Dependency Example

Below is an example of namespace dependency and a description of how the Create Library, Show Dependency and Extract Metadata behaves with the two namespaces. The example contains two namespaces: **ns1** and **ns2**.

The screenshot shows the AVM Metadata interface. On the left, the tree view displays two namespaces: **ns1** and **ns2**. **ns1** contains Data Dictionaries, Data Types (with items **Ns1Decimal (Ns1)** and **ns1Integer**), and Documents (with item **ns1Doc**, which has UserInterface, Default, and Menu). **ns2** contains Presentation, User Interfaces, and NSApplication (Application), which includes Default, Help, Menu, and Page. The right panel shows the **ns1 Default** form. It contains a **Menu** node with children **HorizontalLayout**, **HorizontalLayoutForMenu**, **HorizontalLayoutForUser**, **WrkList**, and **MenuItem**. To the right is a table of properties:

Name	Value
Accelerator	
Action Auditor	
Click method	ns1
Dynamic Label Style	
Enabled	

**ns1** has two local data types that are referenced as variables in the namespace **ns1** document. These variables are further referenced in the Document user interface's default form.

The screenshot shows the AVM Metadata interface. The tree view on the left shows namespaces **ns1** and **ns2**. The **ns1** section is expanded, showing Data Dictionaries, Data Types (with items **Ns1Decimal (Ns1)** and **ns1Integer**), and Documents (with item **ns1Doc**, which has UserInterface, Default, and Menu). The right panel shows the **ns1** document's default form. It has three fields: **Ns1** (String type), **ns1Integer** (String type), and **ns1Integer** (String type). Below the form is a table of properties for the **Default** node:

Name	Value
Run Trigger	
Show Label	<input checked="" type="checkbox"/>
Start Row	<input type="checkbox"/>
Style	
Text Align	
Textbox Style	
Tooltip	
Variable	model.Ns1Decimal
Visible	
Width	

**ns2** contains a user interface menu that simply has a user action method that calls the **ns1** document default form. This menu depends on **ns1** for the form and the associated variables.

## Show Dependency

By performing this function on **ns1** results in "no dependency found". However, performing this function on **ns2** results in a Dependency List that contains **ns1**.

## Create Library

When the user tries to create a library file for the selected namespace, **ns2**, the process is interrupted as there are dependencies. The system displays the Dependency List showing **ns1**.

When the user tries to create a library file for namespace **ns1**, since there are no dependencies, the system will create a library JAR file for the **ns1** metadata and then delete the associated **ns1** metadata.

Now that the ns1 is no longer a namespace (as it is a library file), it is possible to create a library file for ns2 because ns1 is no longer a dependent namespace.

## Extract Metadata

By performing Extract Metadata on **ns1**, the system creates a copy of the **ns1** metadata and its dependent library files (**ns2**) to a new project.

## Data Types

Data Type is a fundamental type on top of which more complex structures, such as Documents and Data Structures, are built. In the **Navigation** pane they are located in **Data Dictionaries** folder in each namespace.

Velocity Studio offers the following system-defined primitive Data Types:

Primitive Data Type	Hierarchy Path	Description
<b>Boolean</b>	<code>com.conceptwave.system.Boolean</code>	Boolean; true or false.
<b>Date</b>	<code>com.conceptwave.system.Date</code>	Date.
<b>DateTime</b>	<code>com.conceptwave.system.DateTime</code>	Date and time of day, together. The time is represented in seconds.
<b>Decimal</b>	<code>com.conceptwave.system.Decimal</code>	Decimal, in format <code>[whole number].[scale]</code> (for example, 178.8734). The scale part has maximum size of 9 (decimal) digits. The decimal length consists of the maximum size of 38 (decimal) digits plus the scale length. For example, if the scale is 9, the maximum length for the decimal is $38 + 9 = 47$ digits.
<b>Integer</b>	<code>com.conceptwave.system.Integer</code>	Integer; can be negative or positive up to 9 (decimal) digits.
<b>String</b>	<code>com.conceptwave.system.String</code>	String of unlimited length.  <b>Note:</b> When you define a string data type without any length, the Velocity Studio generate an NCLOB. An NCLOB generally leads to slower performance and a larger database size.
<b>Timestamp</b>	<code>com.conceptwave.system.Timestamp</code>	System time. It has precision to nanoseconds, but can only be presented up to milliseconds.
<b>Timezone</b>	<code>api_common.data.timezone</code>	Timezone, which inherits a String data type. This data type works with a DateTime Element that binds to a DateTime data type of <b>Absolute Time</b> . The Timezone data type work with a combination box that binds to this data type.
<b>Translation</b>	<code>com.conceptwave.system.Translation</code>	String of unlimited length. Works with the Translation element.

**Note:** There is also a **Void** primitive Data Type, but it is a special Data Type and cannot be extended to other Data Types. All six primitive Data Types and the Void Data Type extends from base Data Type object **Data Type**.

New Data Types are defined by extending from these Data Types, or other Data Types that extend from these Data Types. For example, there are system Data Types such as **Translation** that are extended from primitive Data Types.

To create a new Data Type, complete these steps:

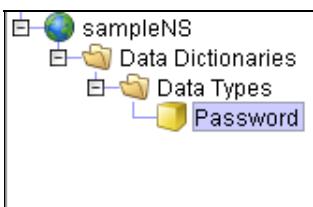
In the **Metadata** tab of **Navigation** pane, either:

1. Right-click a Namespace. Select **New ...**
2. **New Metadata Object** wizard appears as pop-up. Expand **Data Dictionaries**, select **Data Type**, and then click the **Next** button.
3. Step two of wizard appears. Enter the name of Data Type (must conform to JavaScript naming conventions). For the **Extends** field, either choose a primitive Data Type or another defined Data Type. Then, click the **Finish** button.

OR

1. Right-click the **Data Types** folder or the **Data Dictionaries** folder in a Namespace, if it is present. Select **New Data Type**.
2. Follow step 3 in the previous section.

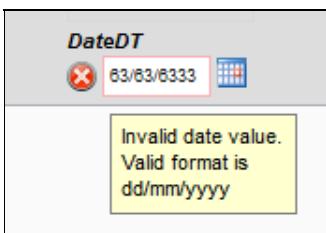
After the Data Type is created by the wizard, you can no longer change the base Data Type that extends it. The newly created Data Type is added under **Data Types** folder.



Other than the primitive Data Types defined in system, there are many predefined data types that are used as base types to build user-defined Data Types and System Types. Their definition can be found in `cwf`, `cwf_ue`, `cwf_pm`, `cwf_up` Namespaces in the **Library** tab of the Navigation Pane.

## Validation of Data Types

Validation ensures that provided value is valid for their intended data types. The Length, Scale, Min, Max, and Default value fields on the [data type general tab](#) are used for the validation. The validation is done at the runtime. An error occurs if the data type value or the format is wrong. For example, the format for the date is set to DD/MM/YYYY, you can enter the value only in this format 12/12/2013. A wrong value, such as 63/63/6333 produces an error.



With Documents or Data structures, use the `CwObject.validate()` in a script for full validation on data types Date, Datetime, Timestamp, Integer, and Decimal . This method returns a list of validation errors.

**Note:** Validation for number data types (Integer and Decimal) and dates (Date, Datetime and Timestamp) is not complete. Data type validation is available for min and max on numbers and for length on dates.

With the implementation of full validation for the data types (Integer and Decimal) and dates (Date, Datetime and Timestamp), validation will occur for content as well. Invalid content will generate errors which will be part of the error list returned when invoking validation on Documents and Data structures containing leaves with the above data types.

### Examples:

- 1) A data structure with a variable named `start_date` and a Date data type represents the order start date. If `start_date` is assigned with content such as "Hello", invoking validate results in no errors. After the modifications in the validation logic are implemented, validating the data structure results in an error indicating that the content for `start_date` is invalid.
- 2) A Document containing a variable named `days` with an Integer data type is used to store the number days for order fulfillment. If the content "ABCD" is assigned to the variable `days`, the validate logic will return no errors as the content will not be validated. After the changes in the validation logic, an error message indicates that "ABCD" is not a valid content for the variable `days`.

## Data Reference Types

A particular set of predefined Data Types is worth special attention -- the **Data Reference Types** (or *Reference Type* in short). A Reference Type is a Data Type of *Reference Key* -- an object that references a Document or an Order object.

There are three types of base Reference Types:

Data Reference Type	Hierarchy Path	Description
String Reference Type	<code>cwf.sReference</code>	Reference Key is a String.
Integer Reference Type	<code>cwf.iReference</code>	Reference Key is an Integer.
System Reference Type	<code>cwf.nReference</code>	Reference Key is of system-defined type. Typically used for referencing an object with composite keys.

The Reference Type is associated with a Finder. The Reference Keys of this Reference Type refers objects that is of the Finder's output.

Using the reference type you can display data in several ways:

- a drop-down menu containing a list of results documents (see [Reference Element](#))
- a text field with a reference icon. Clicking the icon displays a finder that contains a list of results (see [Reference Element](#))
- an editable reference field. Clicking the icon displays document details (see [Reference Element](#))
- a [Large Text](#) Element type that displays a text box containing a user-specified amount of data. Clicking the icon displays the full text

## Create a Reference Type Data Type

Creating a reference data type allows a maximum length of 256 characters. The following is an example of creating a reference data type with this maximum length:

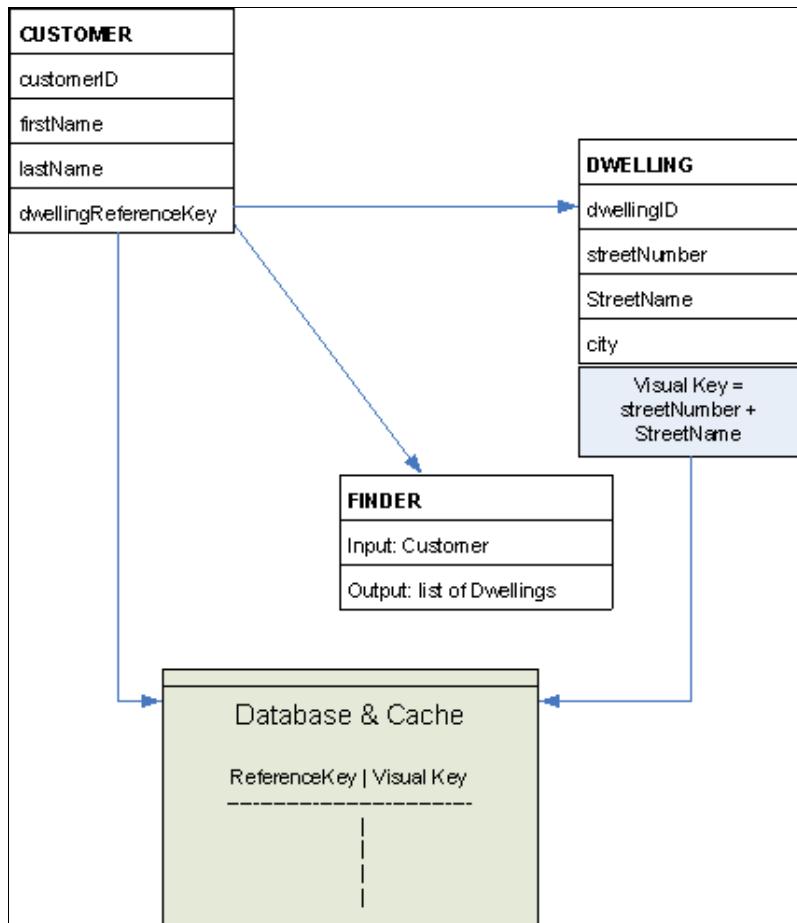
1. Create a new document with a key of String type and a length of 256 characters.
2. Create a finder that returns the new document from the previous step.
3. Create a new reference data type with type `sRef`, a length of 256 characters, and uses the finder created in the previous step.

4. Create a user interface form.
5. Add a new layout. Under the layout, add a new Select Field element.
6. Use the new reference data type in Step 3 for the enumeration reference.

After saving your changes and running your project, no errors appear in the Velocity Studio console. The Select Field element lists available options returned from the finder.

### Reference Type Example

The following example show you how to incorporate a Reference Key element into model. A customer includes a document that includes *dwelling* information in their model. The customer has also included the document *Customer* and it has a *Dwelling* Reference Key that points to the document *Dwelling*, and specifically, to the unique key *dwellingID* which is a String. The *Dwelling* Reference Key is of a user-defined Data Type that is extended from the String Reference Type. The Reference Type has an associated Finder which includes input as *Customer* and output as *Dwelling*.



In the interface presentation, the Reference Key is displayed as the referenced object's Visual Key by default, which is defined in a script at the referenced object.

For example, you can create *cwOnDocVisualKey* Visual Key that returns a value, in this case, the *streetNumber* in the Reference Key field.

Dwelling

cwOnDocVisualKey

Name: cwOnDocVisualKey

Description:

Script:

```
function cwOnDocVisualKey() { // Computes the document visual key when requested
    var document=this; // Deprecated - use 'this'
    return this.streetNumber;
}
```

Any DwellingIDs that are entered on the Customer form can be retrieved using the dwelling ReferenceKey field.

Internally, the Framework has a system table in the database that keeps the correspondence between the reference key and the visual key. This correspondence is in addition to existing functionality of the Finder that is referenced by the Referenced Key. For performance reasons, a cache of the most recently accessed references is maintained in the runtime server.

**Note:** References that are defined to have Visual Keys are stored in an internal table. Visual Keys in this table are updated automatically by the framework as the referenced Documents are changed, but be aware that data in this table may be stale under the following conditions:

- Referenced Documents are changed external to the framework.
- Data residing in a foreign system is changed.
- Visual key script is changed.

## Data Type General Properties

The set of properties available in a Data Type varies depending on the Base Data Type that it **Extends** upon. Some properties are inherited from the Base Data Type, and depending on circumstances, may or may not change from the given inheritance. The **General** tab of a Data Type is as follows:

The screenshot shows the 'General' tab of the Data Type configuration interface. It includes fields for Name, Label, Description, Help, Extends, Overrides, Length, Min, Max, Default value, XML name, JSON name, Text format, and Element properties. There are also sections for Pattern, Scale, and Text length. A table at the bottom lists element properties with columns for Name and Value.

Name	Value

The following table describes the fields:

Field	Description
<b>Name</b>	Name of the metadata object (used in scripts; must conform to JavaScript naming conventions). Read-only after creation. To change it, use the <b>Rename</b> command in pop-up menu by right-clicking the metadata object.
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Final</b>	Metadata elements can be marked as final, which includes any Boolean final property that is available on any metadata element. Additionally, any metadata element that has the <b>Final</b> property set to true can neither be extended, nor overridden.
<b>Label</b>	Metadata object display label.
<b>Description</b>	Description of the metadata object for documentation.
<b>Help</b>	Context sensitive user help for the framework.
<b>Extends</b>	Base Data Type to derive from. Unchangeable after Data Type creation.
<b>Visual key</b>	Appears only when Data Type is a <a href="#">Reference Type</a> . A boolean that, if true, specifies that visual key of the referenced Document is to be displayed in the UI. If false, the reference key itself is displayed instead.
<b>Overrides</b>	Metadata object to override. Used to customize system or library metadata objects.

<b>Finder</b>	Only applicable to <a href="#">Reference</a> types. This is the Finder that defines the Reference type. Specifically, the output document of the Finder is the reference target of the Reference type. The select operation of the Finder returns the dataset of the reference target available to be chosen by user.
	<b>Note:</b> If a Script Finder is chosen, the Script Finder must have <code>cwOnFinderGet</code> Method defined to access <code>.refData</code> and <code>.reference</code> in the corresponding <a href="#">ReferenceValue</a> object in script, unless the output document is mapped to the database.
<b>Event</b>	<p>Only applicable to <a href="#">Reference</a> types. This field represents the event that defines the Reference type. See the <a href="#">Reference</a> property for details.</p> <p>This field acts as a substitute for the <b>Finder</b> field. Events in which the handler returns either the full name of the finder or an instance of the finder object itself to be used as reference must be specified for this field.</p> <p><b>Note:</b> A data type must have either the <b>Finder</b> or the <b>Event</b> field specified, but not both fields.</p> <p>The following is an example of using the <b>Event</b> field:</p> <ol style="list-style-type: none"> <li>1. Create an event.</li> <li>2. Create an event handler for the event that returns the full name of the finder to use (for example, return "cwa_security.userFinder").</li> <li>3. Create a data type and set its <b>Event</b> field to the event that you have just created.</li> <li>4. Use this data type as a variable's data type, bind the variable to a <b>Select</b> field, and display the input field in a user interface.</li> <li>5. At runtime, when you click the drop-down icon, the list shown comes from the finder.</li> </ol>
<b>Length</b>	Only applicable to Decimal, Integer or String Data Types. For Strings, it specifies allowable max character length. For Decimals and Integers, it specifies allowable max number of digits (that is, negative sign and decimal point do not count). Select from drop-down listbox the inherited choice (displayed with <b>(&lt;Inherited&gt;)</b> appended) or <b>&lt;Unlimited&gt;</b> , or type in a number in the textbox to specify the length.
<b>Scale</b>	Only applicable to Decimal Data Types. Specifies the number of digits that appear to the right of the Decimal. Select from drop-down listbox the inherited choice (displayed with <b>(&lt;Inherited&gt;)</b> appended) or <b>&lt;Unlimited&gt;</b> , or type in a number in the textbox to specify the scale. Refer to the Notes section below for more details.
<b>Min</b>	Only applicable to Integer or Decimal Data Types. The inclusive minimum value for the Data Type. Select from drop-down listbox the inherited choice (displayed with <b>(&lt;Inherited&gt;)</b> appended), or type in a number in the textbox to specify the minimum value.
<b>Max</b>	Only applicable to Integer or Decimal Data Types. The inclusive maximum value for the Data Type. Select from drop-down listbox the inherited choice (displayed with <b>(&lt;Inherited&gt;)</b> appended), or type in a number in the textbox to specify the maximum value.
<b>Default value</b>	<p>This field is only applicable to Integer, Decimal, Boolean, String, Date, and DateTime data types. The data type's default value is assumed in initialization.</p> <p>The default value for Date and DateTime types can be specified either using the display format defined in the Element properties, or the default display format: <i>MM/dd/yyyy</i> for Date and <i>MM/dd/yyyy HH:mm</i> for DateTime.</p>
<b>Nullable</b>	If checked, this Data Type may have an empty/null value. If unchecked, a value must be specified. The default is checked.
<b>Encrypt</b>	If checked, this Data Type will be encrypted when saved and decrypted during loading procedures. This checkbox will be disabled if this datatype is inherited from another data type that is flagged as encrypted. <b>Note:</b> If the inherited datatype does not have this encrypted flag checked then the child datatype can be flagged for encryption.
<b>Auto format</b>	<p>This flag defaults to being checked, meaning that an integer or decimal data type contains commas (for example, 1,234,567). Unchecking this property sets it to false, indicating that an integer or decimal value does not contain any commas.</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• This checkbox is displayed only if the data type extends either integer or decimal.</li> <li>• When this checkbox is false, no commas appear in the data type. For example, if the display format is set to <code>\d\,\d\,\d\,\d\,\d\,\d\,\d</code>, the data type still formats with a decimal.</li> <li>• When numeric values having a length longer than the display format, a validation error is thrown in the Web browser.</li> </ul>
<b>Reference</b>	<p>This property is available for a data type if it extends a String, Integer, or Decimal system type, or any other type based on these system types. When this property is selected, the type is considered to be a reference. As a result, you need to specify the <b>Finder</b> or <b>Event</b> property to get a reference finder at runtime. A variable of that type can then be used in a reference field.</p> <p><b>Extending a type with the Reference property</b></p> <p>All extending child data types inherit the <b>Reference</b> property and become a reference type automatically. The checkbox displays selected, but disabled.</p> <p><b>Note:</b> Child types do not inherit the <b>Finder</b> or <b>Event</b> property. As a result, properties need to be set for each child separately.</p> <p><b>Parent-child collisions</b></p>

There is the potential for parent-child type collisions:

- If the base is not a reference when you create the child type, but becomes a reference later, the child type automatically becomes a reference, too. In this case, Velocity Studio displays an error for the child type, requiring you to specify either the **Finder** or **Event** property.
- If the base type was a reference, but this property's checkbox was deselected for the parent, the child type does not become a reference either. However, the **Reference** checkbox becomes enabled for the child type, so it can be corrected.

#### cwf.sReference, cwf.iReference, and cwf.nReference system types

By default, these system types are still reference types. This property's checkbox is selected and disabled for them, as well as for all child types based on them.

<b>XML name</b>	The XML tag name for this Data Type. Defaults to Name. This field is used when exporting data to XML.
<b>JSON name</b>	The JSON tag name for this Data Type. Defaults to Name. This field is used when exporting data to JSON.
<b>Pattern</b>	Text mapping format pattern, which is used when it appears on a document using a text map. This field is used when exporting data to XML.
<b>Text format</b>	Text mapping format (refer to section <a href="#">Data Formats</a> ), which is used when it appears on a document that uses a text map. This field is used when exporting data to XML.
<b>Text length</b>	Text mapping format length, which is used when it appears on a document using a text map. This field is used when exporting data to XML.
<b>Element</b>	The type of <a href="#">Form Element</a> that the Data Type shall be defaulted to present. Available choices depend on the base Data Type, and whether enumeration is defined.
<b>Element Properties</b>	<p>List of properties that the Form Element of a Variable of this Data Type shall inherit. These properties can be overridden at lower levels; see <a href="#">Permissions</a> for details. Available properties to be inherited depend on the selected Element Type; they are:</p> <ul style="list-style-type: none"> <li>• <b>Display Format:</b> Specifies, in regular expression, the format of how the field Element shall be displayed. The use of this property depends on the Element Type selected. See documentation of the selected Element Type for details.</li> <li>• <b>Visible:</b> Determines whether the field Element is visible in UI or not. True or false, or assign a Permission Method to determine the boolean result.</li> <li>• <b>Editable:</b> Determines whether the field Element is editable (true) or read-only (false). True or false, or assign a Permission Method to determine the boolean result.</li> </ul> <p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.</p> <ul style="list-style-type: none"> <li>• <b>Optional:</b> Determines whether the field Element can have empty value (true) or a mandatory field (false). True or false, or assign a Permission Method to determine the boolean result.</li> </ul>

#### Notes:

- Data Types may only specify more restrictive values than their base Data Types. For example, Length of an extended Data Type cannot be longer than its base Data Type.
- If you change the Element of the Data Type after Forms are created based on Variables of this Data Type, this change does not propagate to the Form. For example, if you change from Select Field to Radio Button Group for a String-based Data Type, the existing Forms that contain Form Elements assigned with Variable of this Data Type will not be automatically changed to Radio Button Group. If desired, you must change it yourself, for each such Form Element across all Forms.
- **Number Formatting:** Decimal and integer data types are strictly numeric by default. When using a Textfield element and assigning a Variable of the type Decimal, the scale and length property will determine how the digits are displayed. The Scale property determines how many digits appear to the right of the decimal. If the user enters decimal digits less than the Scale value, then the system will insert zeros at the end of the decimal number. The length and scale properties of the data type will dictate the number of digits that will appear to the left of the decimal. The difference between the Length and Scale values will determine the number of digits that appear to the left of the decimal.

Length	Scale	User Entry	Display Value
2	3	123	.123
4	3	123	1.230
5	2	123	123.00
5	2	12.3	12.30
5	2	1.23	1.23

- **Number Formatting (commas and decimals):** When displaying data of the type Decimals, the commas and decimals are reversed depending on the browser language settings. For example, 1.23 is displayed for US English and 1,23 is displayed for Portuguese.

- You can override variables with a data type user interface, as long as the data type is the same or extends the original variable's data type.

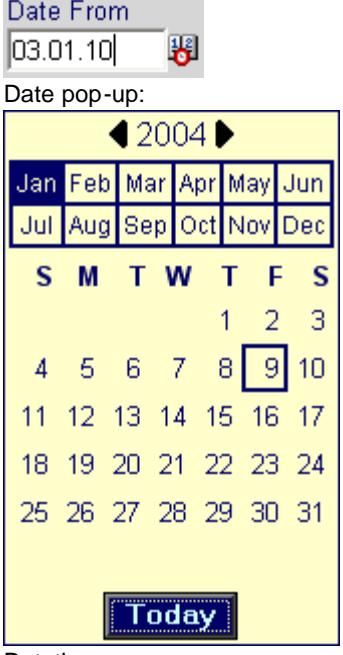
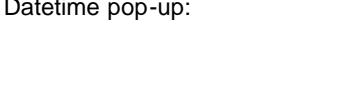
## Data Formats (Classic mode - Deprecated)

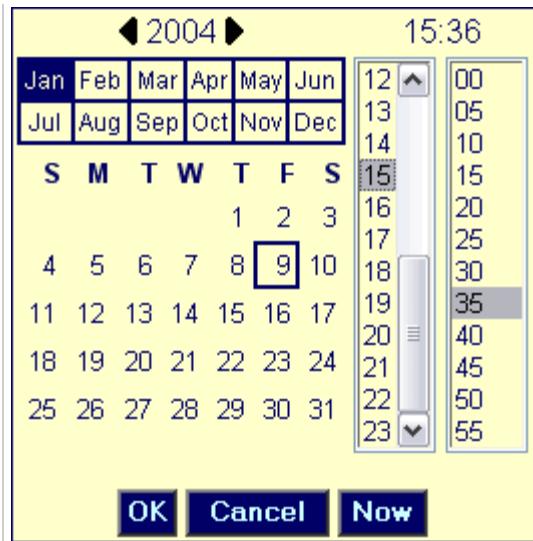
Data formats consist of action control codes and formatting masks. Format control codes are preceded by a backslash. Masks consist of formatting characters and inserted characters. The allowed format control codes, formatting and inserted characters depend on the data type, which is set in the **Extends** field on the **General** tab, for which the format is defined. These are the basic data types from a formatting prospective:

- Numbers
- Strings
- Booleans
- Dates, Times and Timestamps

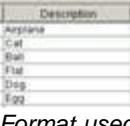
### Format Control Codes

Table below provides information for all supported format codes.

Format	Description	Data Type	UI example
\a	Display field actual width. The field will only occupy the space required rather than a full column.	All	
\b	Multiselect.		
\cf	Field is to be horizontally centred.	Image	
\c#	Number of columns for field (1-10).	All	
\d	Pop-up calendar field. Format may include any combination of yy (or yyyy), MM, and dd formats. Default format is yyyy/MM/dd.	Date, Time and Datetime.	<p>Date From 03.01.10 </p> <p>Date pop-up:</p>  <p>Datetime pop-up:</p> 



\di	Dynamic image. The field should contain a resource image ID. At runtime it is displayed as an icon.	String	
\ef	Editable visual key. Allows the visual key to be directly edited.  <b>Note:</b> Displaying visual keys for multi-select reference fields does not work with the editable visual key format.	Reference	
\en	Non verified.		
\h100%	Height percent.		
\he	Hides an empty selection in the select list.	Enumeration.	
\h#	Maximum height of the field in lines (1 to 25). For enumerations values > 1, will display field as a select list.	String or any Enumeration.	
\it	Immediate trigger. Not implemented.		
\l	Field label width (1-200).	All	
\L	Display reference as a drop-down list instead of a Finder. The reference Finder result list will be displayed in the drop-down list. If the reference Finder has a search Document, the current Document will be mapped to it before the search is performed.  <b>Note:</b> If there is no search form in the reference Finder, the number of DB results can not be restricted (that is, all records present in the DB results will be available in the drop-down list). This may introduce serious performance problems.	Reference	
\m	Multi-part field. The format mask defines how the field is divided into parts (for example, ###-####).	String	<p>string P1 <input type="text"/> P2 <input type="text"/> Format: \m'P1'**'P2'**)</p>
\n	No field label displayed in the UI.	All	
\P	Display as a password field. Cannot be used on multiline fields.	String	
\rv	Vertical radio button.	String and Enumeration	<p>selection ○ A ○ B ○ C</p>

\rh	Horizontal radio button.	String and Enumeration	
\ro	Read only field.	All	
\r	Horizontal radio button. Deprecated. \rh should be used instead.		
\sl	Search list always. Clicking the reference field displays the search list with the corresponding reference highlighted.	Reference	
\s	Sort drop down list items (Ascending).	String and Enumeration	<p>In UI:</p>  <p>In metadata</p>  <p>Format used: \s</p>
\ss	Auto-select in a drop down if there is only one item to select.	String and Enumeration	
\w100%	Width percent.		
\w#	Width of the field in characters (1 to 2000). Number of columns that the field occupies will be calculated using this width rather than the Data Type or Element Type width.	String	
\x	Display boolean as a checkbox.	Boolean	
\z	Default button.	Button	

## Boolean Formatting

The UI represents boolean as a check box if the **Nullable** field is unchecked, otherwise it will be two radio buttons.



If \rv is set in the format field, the **Yes** and **No** radio buttons will be displayed vertically.

A default value, 1=true, 0=false, can be assigned to a boolean Data Type in the **Default** field in the **General** tab.

## Number Formatting

Numeric formats represent the pattern by which the numbers are displayed. When numbers are entered, these rules are applied:

- Only the following characters are allowed: minus (-), comma (,), dot (.) and digits.
- All leading integer zeroes and trailing fractional zeroes are ignored.
- All commas in the integer part are ignored.

Numeric formats differ from the other formats in the sense that they do not specify what exact character should be in any position of the field (as the string formats and the fixed date formats do). The format pattern specifies the grouping size and maximum number of fractional digits. The grouping size is a constant number of digits between the grouping characters, such as 3 for 100,000,000 or 4 for 1,0000,0000. If you supply a pattern with multiple grouping characters, the interval between the last one and the end of the integer is the one that is used. For example #,##,###,### == #####,### == ##,###,###. Illegal patterns, such as #.#.# or #.###,###, will generate a message that describes the problem.

Table below summarizes allowable numeric formatting characters.

Symbol	Localized	Description
0	Y	Digit.
#	Y	Digit, zero shows as absent.
.	Y	Decimal separator or monetary decimal separator.
-	Y	Minus sign.
,	Y	Grouping separator.

### String Formatting

Below table summarizes allowable string formatting characters.

Symbol	Description
*	Any character.
#	Digit.
L	Uppercase letter.
I	Lowercase letter.
M	Mixed case letter.
A	Uppercase alphanumeric.
a	Lowercase alphanumeric.
N	Mixed case alphanumeric.

Any character that does not match the editing characters is considered an inserted character. If one of the formatted characters or the single quote character has to be inserted, they must be put in single quotes. Within the quoted group, the single quote is represented as two quotes.

For example: format \*\*\* "sharp" for data ABC will result in ABC sharp

### Date, Time and Timestamp Formatting

Date formats define a set of date/time fields that are either fixed length or variable length. If the format is variable length, it cannot be used for an updateable field.

To specify the time format, use a time pattern string. In this pattern, all ASCII letters are reserved as pattern letters.

Symbol	Description	Presentation	Example	Variable length
T	Era designator	Text	AD	Yes
y	Year.	Number	1996	No
M	Month in year	Text and Number	July & 07	Yes if more than 3 M characters
d	Day in month	Number	10	Yes if one d character
h	Hour in am/pm (1~12)	Number	12	Yes if one h character
H	Hour in day (0~23)	Number	0	Yes if one H character
m	Minute in hour	Number	30	Yes if one m character
s	Second in minute	Number	55	Yes if one s character
S	Millisecond	Number	978	Yes if less than 3 S characters
E	Day in week	Text	Tuesday	Yes
D	Day in year	Number	189	Yes if less than 3 D characters.
F	Day of week in month	Number.	2 (2nd Wed in July)	No

w	Week in year	Number	27	Yes if less than 2 w characters
W	Week in month	Number	2	No
a	AM/PM marker	Text	PM	Yes if less than 2 a characters
k	Hour in day (1~24)	Number	24	Yes if less than 2 k characters
K	Hour in am/pm (0~11)	Number	0	Yes if less than 2 K characters
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-05:00	Yes
'	Escape for text	Delimiter		No
"	Single quote	Literal	'	No

The count of pattern letters determines the format. If a format contains values that are out of the range specified in above table, the date/time values will be automatically adjusted by the parser to the closest valid combination. Such a usage may produce an unpredictable result and should be avoided.

For text with 4 or more pattern letters, use the full form. For less than 4 pattern letters, use the short or abbreviated form.

For numbers, use the minimum number of digits. Shorter numbers are zero-padded to this amount.

Year is handled uniquely. For example, if the count of y is 2, the Year will be truncated to 2 digits. When parsing a date string using the abbreviated year pattern (y or yy), the format must interpret the abbreviated year relative to some century. It does this by adjusting dates to be within 80 years before and 20 years after the time the format instance is created. For example, using a pattern of MM/dd/yy and a format instance created on Jan 1, 1997, the string 01/11/12 would be interpreted as Jan 11, 2012 while the string 05/04/64 would be interpreted as May 4, 1964.

During parsing, only strings consisting of exactly two digits will be parsed into the default century. Any other numeric string, such as a one digit string, a three or more digit string, or a two digit string that is not all digits (for example, -1), is interpreted literally. So 01/02/3 or 01/02/003 are parsed, using the same pattern, as Jan 2, 3 AD. Likewise, 01/02/-3 is parsed as Jan 2, 4 BC. If the year pattern has more than two y characters, the year is interpreted literally, regardless of the number of digits. So, using the pattern MM/dd/yyyy, 01/11/12 parses to Jan 11, 12 A.D.

For time zones that have no names, use strings GMT+hours:minutes or GMT-hours:minutes. The calendar defines the first day of the week, the first week of the year, whether hours are zero based or not (0 vs 12 or 24) and the time zone.

For text and numbers of 3 or more use text, otherwise use numbers.

Any characters in the pattern that are not in the range of '['a'..'z'] and '['A'..'Z'] will be treated as quoted text. For instance, characters like '!', ',', '#' and '@' will appear in the resulting time text even though they are not embraced within single quotes. A pattern containing any invalid pattern letter will result in a thrown exception during formatting or parsing. Examples of time formatting using the US Locale are shown in table below.

Format Pattern	Result
yyyy.MM.dd T 'at' hh:mm:ss z	1996.07.10 AD at 15:08:56 PDT
EEE,MMM d, "yy	Wed, July 10, '96
h:mm a	12:08 PM
hh 'o'clock' a, zzzz	12 o'clock PM, Pacific Daylight Time
K:mm a, z	0:00 PM, PST
yyyy.MMMMMdd TTT hh:mm aaa	1996.July.10 AD 12:08 PM

## Format editing

The **Format** field in the **General tab** consists of an editable text field and a button beside it. Click the button to open the **Format Properties** dialog (shown in diagram below), which provides a convenient way to create and modify format control codes.

**Format Properties**

Select	Description	Code	Value
<input type="checkbox"/>	Actual size	\a	
<input type="checkbox"/>	Multiselect	\b	
<input type="checkbox"/>	Centered	\cf	
<input type="checkbox"/>	Number of columns	\c	1
<input type="checkbox"/>	Dynamic image	\di	
<input type="checkbox"/>	Editable visual key	\ef	
<input type="checkbox"/>	Non verified	\en	
<input type="checkbox"/>	Height percent	\h100%	
<input type="checkbox"/>	Hide empty	\he	
<input type="checkbox"/>	Height lines	\h	1
<input type="checkbox"/>	Immediate trigger	\it	
<input type="checkbox"/>	Label length	\l	1
<input type="checkbox"/>	Reference drop-down	\L	
<input checked="" type="checkbox"/>	Multi-part field	\m	
<input type="checkbox"/>	No label	\n	
<input type="checkbox"/>	Password field	\P	
<input type="checkbox"/>	Vertical radio-button	\rv	
<input type="checkbox"/>	Horizontal radio-button	\rh	
<input type="checkbox"/>	Read only	\ro	
<input type="checkbox"/>	Horizontal radio-button	\r	
<input type="checkbox"/>	Search list always	\sl	
<input type="checkbox"/>	Sorted	\s	
<input type="checkbox"/>	Width 100%	\w100%	
<input type="checkbox"/>	Data length	\w	1
<input type="checkbox"/>	Checkbox	\x	
<input type="checkbox"/>	Default	\z	

**All****None****OK****Cancel**

The dialog shows a list of all format codes available for selection for this data type. Individual selection can be made using the check boxes in the **Select** column. Click the **All** button to select all format codes in the list. Click the **None** button to deselect all format codes. Click **OK** to close the dialog and convert the selected format codes into the format string. The generated string will be added to the **Format** field at the end of format masks if they exist.

<b>Format:</b>	***-**-****\m	<input type="button" value="..."/>
----------------	---------------	------------------------------------

## Data Type Methods

Use the **Methods** tab in Data Type to create data validation methods. The validation methods are user-defined JavaScript script that determine the validity of runtime data values. The result of the script execution determines the state of the Document that contains the Data Type. At runtime the user will be presented with the messages specified for the validation scripts that have fired. Further, the user will be able to navigate to the associated data element.

### Notes:

- Methods defined in derived types override the base type Methods.
- See the [phasedValidation configuration variable](#) for validating all variables and document rule scripts even when there are errors for mandatory fields or at the data type level.

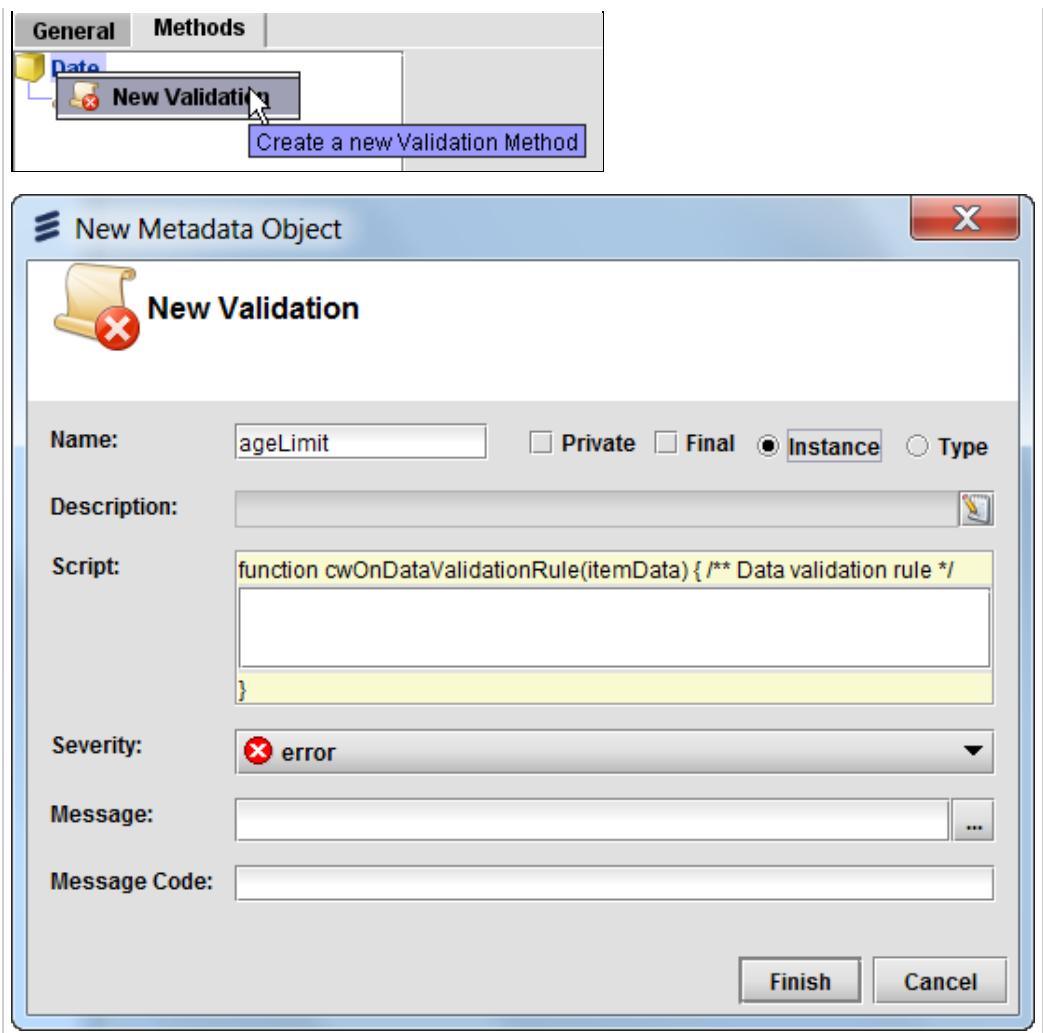
The screenshot shows the 'Methods' tab of the Data Type configuration interface. On the left, a tree view shows a 'Date' node with a 'validate' method under it. The main panel is titled 'General' and contains the following fields:

- Name:** validate
- Description:** (empty)
- Script:**

```
function cwOnDataValidationRule(itemData) { // Data validation rule
itemData > new Date(2008,6,1)
}
```
- Return:** com.conceptwave.system.Boolean
- Severity:** error
- Message:** Invalid Date Entered.
- Message Code:** (empty)

In the above example at runtime, if data entered into an element of the above Data Type is greater than the date June 1, 2008 (the script itemData > new Date(2008, 6, 1) will evaluate to true) the method will fire and the error message "Invalid date Entered." will be presented to the user.

A new method is added by right-clicking the Data Type root object in the Methods pane, and then click **New Validation**. The **New Validation Method** wizard appears; type in the method name there and click **Finish**.



Below is a description of properties in the Validation Method:

Field	Description
<b>Name</b>	Mandatory. Name of the method. Must be unique for all methods defined for this type.
<b>Description</b>	Description of the method for documentation.
<b>Script</b>	Mandatory. The validation JavaScript script: The data value is passed into the script via the <i>itemData</i> parameter. The script may be in one of the following forms: <ul style="list-style-type: none"> <li>An expression that evaluates to true to indicate that the <b>Message</b> specified is to be displayed. Otherwise it must evaluate to false.</li> <li>A script that returns true to indicate that the <b>Message</b> specified is to be displayed. Otherwise it must return false.</li> <li>A script that returns a resource identifier string or list of identifiers separated by semicolons to indicate the resource message(s) to be displayed. For example: AW0155 or AW0155;AE0156. The first character must be A and the second indicates the type: I=information, W=warning, and E=error.</li> </ul>
<b>Return</b>	The Object type to be returned by the script.
<b>Severity</b>	Indicates the severity of the method: information, warning or error.
<b>Message</b>	Defines the error message to display to the user when this method fires. It is mandatory that one of either the <b>Message</b> or <b>Message Code</b> fields are specified. If both the <b>Message</b> and <b>Message Code</b> fields are specified, the <b>Message Code</b> will be ignored.
<b>Message Code</b>	Defines the message code of the message to retrieve, from Resources, to display to the user. It is mandatory that one of either the <b>Message</b> or <b>Message Code</b> fields are specified. If both the <b>Message</b> and <b>Message Code</b> fields are specified, the <b>Message Code</b> will be ignored.

To delete a Method, click the node representing the method, then click the **Delete** button.

## Data Type Enumeration

The **Enumeration** tab in Data Type is only available to Data Types with Decimal, Integer and String primitive Data Types as base. If an enumeration restriction is needed, define in this tab, the list of valid choices in the enumeration. Otherwise, leave empty in this tab.

**Note:** Enumeration defined in derived types have the base Data Type enumeration upon creation, but the list of choices can be added and removed.

Code	Description	Active
AC	Active	<input checked="" type="checkbox"/>
IN	Inactive	<input checked="" type="checkbox"/>
SUS	Suspend	<input checked="" type="checkbox"/>

Below is a description of properties in the Enumeration tab:

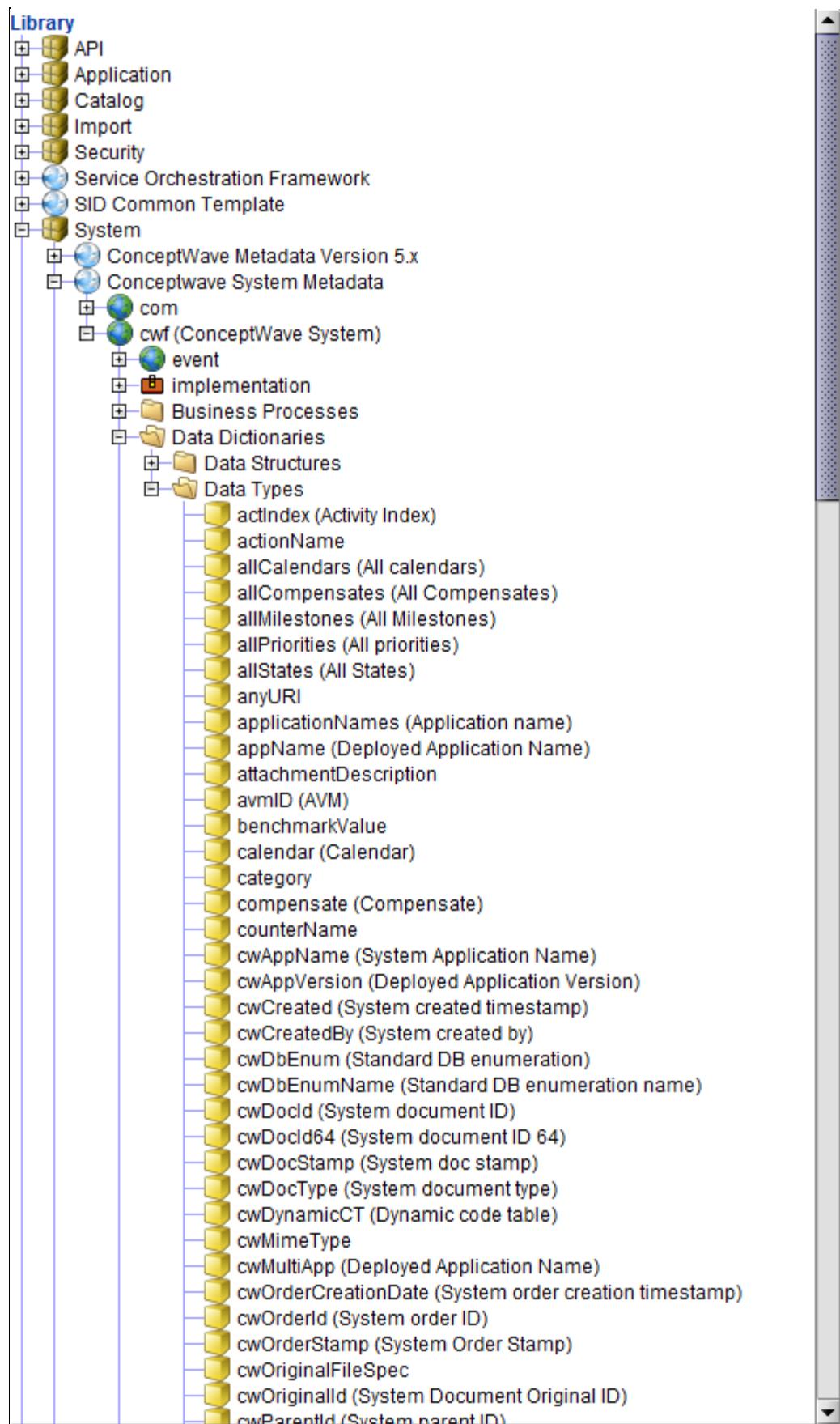
Field	Description
<b>Finder</b>	Only applicable to Reference Types. This field specifies the Finder to use the valid choices of the enumeration that can be assumed by the Reference Key. The Finder's input shall be of the same metadata object as the one that contains the Reference Key. The Finder's output determines the object type that the Reference Key is referencing. See <a href="#">documentation on Reference Types</a> for details.
<b>System DB enum</b>	Specifies the system database code table to use to find the valid choices of the enumeration (for example, provState_CA and iso6392 are valid system database code table names that can be used in this field). To create the table, use the <a href="#">Code Tables</a> option in the System Administration application.
<b>Enumeration</b>	List of valid values and corresponding descriptions An entry contains: <ul style="list-style-type: none"><li><b>Code:</b> The data value.</li><li><b>Description:</b> Descriptive text to present to the user.</li><li><b>Active:</b> If not checked, the value is not allowed to be selected from the UI, but may be displayed when viewing existing</li></ul>

| data.

## **Predefined Data Types**

Other than the primitive Data Types defined in system, there are other system predefined data types that are used as base types to build user-defined Data Types and System Types.

System data types are found in the **System > cwf** namespace, in the **Library** tab.



The following table describes a few predefined system data types:

Label	Name	Type	Description
Deployed Application Name	appName	com.conceptwave.system.String	This data type is used to get the data definition

			for application name.
<b>System Application Name</b>	<i>cwAppName</i>	cwf.appName	With this data type system automatically populate with current application name.
<b>Deployed Application Name</b>	<i>cwMultiApp</i>	cwf.appName	With this data type, you can get a system list of application names deployed on current database.  <b>Note:</b> The <a href="#">Multiple Application Deployment</a> checkbox must be enabled in the System Administration application.

## Translation Data Type

The Translation Data Type is an object that references a Document through a Finder. The Translation data type displays data in a Finder. By configuring the Translation data type and its associated element properties, you are able to change the language of text displayed in the Finder.

The data type contains the base type of Translation type:

Transaction Data Type	Hierarchy Path	Description
Translation Type	cwf.conceptwave.system.Translation	Translation Key is a String.

After creating a Data Type of Translation type, you set its Element properties to Translation.

The screenshot shows the 'Element properties' dialog for a 'Translation' element. The title bar says 'Translation'. The main area is a table with columns 'Name' and 'Value'. There are four rows: 'Optional' (checkbox), 'Editable' (checkbox), 'Visible' (checkbox), and 'Label' (text input field). The 'Optional' checkbox is checked.

Name	Value
Optional	<input checked="" type="checkbox"/>
Editable	<input type="checkbox"/>
Visible	<input type="checkbox"/>
Label	<input type="text"/>

To set the Translation properties in the UI display, configure the [Translation Element Properties](#). In addition to configuring the data type and element properties, you need to set the language setting in Velocity Studio.

The screenshot shows the 'Languages' tab in Velocity Studio. On the left, under 'Available:', 'en-xx [English]' and 'fr-ca [French/Canada]' are listed. A 'Select Language(s)' dialog is open in the center. It has a search bar 'Enter object name prefix or pattern(\*,?):' and a list 'Matching items:' containing Arabic language codes: 'ar-xx [Arabic]', 'ar-ae [Arabic/United Arab Emirates]', 'ar-bh [Arabic/Bahrain]', 'ar-dz [Arabic/Algeria]', 'ar-eg [Arabic/Egypt]', 'ar-iq [Arabic/Iraq]', 'ar-jo [Arabic/Jordan]', and 'ar-kw [Arabic/Kuwait]'. There are 'Save' and 'Cancel' buttons at the bottom. To the right of the dialog are '+' and '-' buttons for managing selected languages.

These language settings correspond to the language settings in your Web browser. Changing your Web browser's language display settings (placing it first in the selection list), enables you to display the Translation Finder's text in that language.



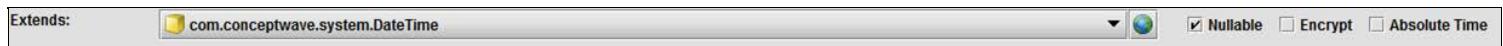
## Absolute Time and Timezone Support

The DateTime data type takes timezone conversion into consideration when working with the data type's value, which Java's Date object does naturally. However, there are cases where the DateTime datatype needs to be absolute. When it is marked as absolute, the timezone is not taken into consideration.

The api\_common.data.timezone system data type provides an enumeration of all valid and supported timezone names. This data type is available in the API\_Common system template.

### Absolute Time Property

In the Data Type General property tab, when the base data type is DateTime, the **Absolute Time** property becomes available. By default, the property is unchecked, meaning that the DateTime uses timezone conversion. Otherwise, selecting this property indicates that timezone conversion is disabled.



This data type works with a DateTime Element that binds to a DateTime data type of **Absolute Time**. The Timezone data type work with a combination box that binds to this data type.

### Runtime

At runtime, the difference between an absolute time DateTime data type and a non-absolute time one becomes evident when working with two different clients using different timezones.

For example, consider a scenario in which Client A works in a GMT-5 timezone and Client B works in a GMT+8 timezone. If Client A enters Jan 6, 2012 03:00 in a document's field that is bound to a non-absolute DateTime data type, when Client B views the same document, Client B sees Jan 6, 2012 16:00. In this case, the time has been converted to Client B's timezone. On the other hand, if Client A enters Jan 6, 2012 03:00 in a document's field that is bound to an absolute time DateTime data type, when Client B views the same document, Client B also sees Jan 6, 2012 03:00.

## Supported Time Zones

---

You can use the `Global.setDisplayTimeZone()` API to set the time zone that is used to display date-time values in the user interface. Refer to the [JavaScript documentation](#) for more information. The first parameter for this API method must be either the time zone ID (for example, `Antarctica/Davis`) or an integer offset in minutes calculated through the `Global.getClientTimeZoneOffset()` API.

The following [Google Web Toolkit \(GWT\)](#) supported time zone names can be used as first parameter of the `Global.setDisplayTimeZone()` API. Refer to the [Time Zone](#) for more information.

Africa	America	Antarctica	Asia	Atlantic	Australia	Europe	Indian	Pacific
Africa/Abidjan	America/Adak	Antarctica/Casey	Asia/Aden	Atlantic/Azores	Australia/Adelaide	Europe/Andorra	Indian/Antananarivo	Pacific/Apia
Africa/Accra	America/Anchorage	Antarctica/Davis	Asia/Almaty	Atlantic/Bermuda	Australia/Brisbane	Europe/Athens	Indian/Chagos	Pacific/Auckland
Africa/Addis_Ababa	America/Anguilla	Antarctica/DumontDUrville	Asia/Amman	Atlantic/Canary	Australia/Broken_Hill	Europe/Belgrade	Indian/Christmas	Pacific/Chatham
Africa/Algiers	America/Antigua	Antarctica/Mawson	Asia/Anadyr	Atlantic/Cape_Verde	Australia/Currie	Europe/Berlin	Indian/Cocos	Pacific/Easter
Africa/Asmara	America/Araguaina	Antarctica/McMurdo	Asia/Aqtau	Atlantic/Faeroe	Australia/Darwin	Europe/Bratislava	Indian/Comoro	Pacific/Efate
Africa/Bamako	America/Argentina/La_Rioja	Antarctica/Palmer	Asia/Aqtobe	Atlantic/Madeira	Australia/Eucla	Europe/Brussels	Indian/Kerguelen	Pacific/Enderbury
Africa/Bangui	America/Argentina/Rio_Gallegos	Antarctica/Rothera	Asia/Ashgabat	Atlantic/Reykjavik	Australia/Hobart	Europe/Bucharest	Indian/Mahe	Pacific/Fakaofa
Africa/Banjul	America/Argentina/San_Juan	Antarctica/Syowa	Asia/Baghdad	Atlantic/South_Georgia	Australia/Lindeman	Europe/Budapest	Indian/Maldives	Pacific/Fiji
Africa/Bissau	America/Argentina/Tucuman	Antarctica/Vostok	Asia/Bahrain	Atlantic/St_Helena	Australia/Lord_Howe	Europe/Chisinau	Indian/Mauritius	Pacific/Funafuti
Africa/Blantyre	America/Argentina/Ushuaia		Asia/Baku	Atlantic/Stanley	Australia/Melbourne	Europe/Copenhagen	Indian/Mayotte	Pacific/Galapagos
Africa/Brazzaville	America/Aruba		Asia/Bangkok		Australia/Perth	Europe/Dublin	Indian/Reunion	Pacific/Gambier
Africa/Bujumbura	America/Asuncion		Asia/Beirut		Australia/Sydney	Europe/Gibraltar		Pacific/Guadalcanal
Africa/Cairo	America/Bahia		Asia/Bishkek			Europe/Helsinki		Pacific/Guam
Africa/Casablanca	America/Barbados		Asia/Brunei			Europe/Istanbul		Pacific/Honolulu
Africa/Ceuta	America/Belem		Asia/Calcutta			Europe/Kaliningrad		Pacific/Johnston
Africa/Conakry	America/Belize		Asia/Choibalsan			Europe/Kiev		Pacific/Kiritimati
Africa/Dakar	America/Boa_Vista		Asia/Chongqing			Europe/Lisbon		Pacific/Kosrae
Africa/Dar_es_Salaam	America/Bogota		Asia/Colombo			Europe/Ljubljana		Pacific/Kwajalein
Africa/Djibouti	America/Boise		Asia/Damascus			Europe/London		Pacific/Majuro
Africa/Douala	America/Buenos_Aires		Asia/Dhaka			Europe/Luxembourg		Pacific/Marquesas
Africa/El_Aaiun	America/Cambridge_Bay		Asia/Dili			Europe/Madrid		Pacific/Midway
Africa/Freetown	America/Campo_Grande		Asia/Dubai			Europe/Malta		Pacific/Nauru
Africa/Gaborone	America/Cancun		Asia/Dushanbe			Europe/Minsk		Pacific/Niue
Africa/Harare	America/Caracas		Asia/Gaza			Europe/Monaco		Pacific/Norfolk
Africa/Johannesburg	America/Catamarca		Asia/Harbin			Europe/Moscow		Pacific/Noumea
Africa/Kampala	America/Cayenne		Asia/Hong_Kong			Europe/Oslo		Pacific/Pago_Pago
Africa/Khartoum	America/Cayman		Asia/Hovd			Europe/Paris		Pacific/Palau
Africa/Kigali	America/Chicago		Asia/Irkutsk			Europe/Podgorica		Pacific/Pitcairn
Africa/Kinshasa	America/Chihuahua		Asia/Jakarta			Europe/Prague		Pacific/Ponape

Africa/Lagos	America/Coral_Harbour		Asia/Jayapura			Europe/Riga	Pacific/Port_Moresby
Africa/Libreville	America/Cordoba		Asia/Jerusalem			Europe/Rome	Pacific/Rarotonga
Africa/Lome	America/Costa_Rica		Asia/Kabul			Europe/Samara	Pacific/Saipan
Africa/Luanda	America/Cuiaba		Asia/Kamchatka			Europe/Sarajevo	Pacific/Tahiti
Africa/Lubumbashi	America/Curacao		Asia/Karachi			Europe/Simferopol	Pacific/Tarawa
Africa/Lusaka	America/Danmarkshavn		Asia/Kashgar			Europe/Skopje	Pacific/Tongatapu
Africa/Malabo	America/Dawson		Asia/Katmandu			Europe/Sofia	Pacific/Truk
Africa/Maputo	America/Dawson_Creek		Asia/Krasnoyarsk			Europe/Stockholm	Pacific/Wake
Africa/Maseru	America/Denver		Asia/Kuala_Lumpur			Europe/Tallinn	Pacific/Wallis
Africa/Mbabane	America/Detroit		Asia/Kuching			Europe/Tirane	
Africa/Mogadishu	America/Dominica		Asia/Kuwait			Europe/Uzhgorod	
Africa/Monrovia	America/Edmonton		Asia/Macau			Europe/Vaduz	
Africa/Nairobi	America/Eirunepe		Asia/Magadan			Europe/Vienna	
Africa/Ndjamena	America/El_Salvador		Asia/Makassar			Europe/Vilnius	
Africa/Niamey	America/Fortaleza		Asia/Manila			Europe/Volgograd	
Africa/Nouakchott	America/Glace_Bay		Asia/Muscat			Europe/Warsaw	
Africa/Ouagadougou	America/Godthab		Asia/Nicosia			Europe/Zagreb	
Africa/Sao_Tome	America/Goose_Bay		Asia/Novosibirsk			Europe/Zaporozhye	
Africa/Tripoli	America/Grand_Turk		Asia/Omsk			Europe/Zurich	
Africa/Tunis	America/Grenada		Asia/Oral				
Africa/Windhoek	America/Guadeloupe		Asia/Phnom_Penh				
	America/Guatemala		Asia/Pontianak				
	America/Guayaquil		Asia/Pyongyang				
	America/Guyana		Asia/Qatar				
	America/Halifax		Asia/Qyzylorda				
	America/Havana		Asia/Rangoon				
	America/Hermosillo		Asia/Riyadh				
	America/Indiana/Knox		Asia/Saigon				
	America/Indiana/Marengo		Asia/Sakhalin				
	America/Indiana/Petersburg		Asia/Samarkand				
	America/Indiana/Vevay		Asia/Seoul				
	America/Indiana/Vincennes		Asia/Shanghai				
	America/Indianapolis		Asia/Singapore				
	America/Inuvik		Asia/Taipei				
	America/Iqaluit		Asia/Tashkent				

America/Jamaica	Asia/Tbilisi
America/Jujuy	Asia/Tehran
America/Juneau	Asia/Thimphu
America/Kentucky/Monticello	Asia/Tokyo
America/La_Paz	Asia/Ulaanbaatar
America/Lima	Asia/Urumqi
America/Los_Angeles	Asia/Vientiane
America/Louisville	Asia/Vladivostok
America/Maceio	Asia/Yakutsk
America/Managua	Asia/Yekaterinburg
America/Manaus	Asia/Yerevan
America/Martinique	
America/Mazatlan	
America/Mendoza	
America/Menominee	
America/Merida	
America/Mexico_City	
America/Miquelon	
America/Moncton	
America/Monterrey	
America/Montevideo	
America/Montreal	
America/Montserrat	
America/Nassau	
America/New_York	
America/Nipigon	
America/Nome	
America/Noronha	
America/North_Dakota/Center	
America/North_Dakota/New_Salem	
America/Panama	
America/Pangnirtung	
America/Paramaribo	
America/Phoenix	
America/Port_of_Spain	

America/Porto_Velho				
America/Puerto_Rico				
America/Rainy_River				
America/Rankin_Inlet				
America/Recife				
America/Regina				
America/Rio_Branco				
America/Santiago				
America/Santo_Domingo				
America/Sao_Paulo				
America/Scoresbysund				
America/Shiprock				
America/St_Johns				
America/St_Kitts				
America/St_Lucia				
America/St_Thomas				
America/St_Vincent				
America/Swift_Current				
America/Tegucigalpa				
America/Thule				
America/Thunder_Bay				
America/Tijuana				
America/Toronto				
America/Tortola				
America/Vancouver				
America/Whitehorse				
America/Winnipeg				
America/Yakutat				
America/Yellowknife				

# Creating User Interface Objects

---

## Top-Level User Interface Objects

A top-level User Interface is not included within a model-based metadata object. You can create a top-level User Interface in the Presentation folder within the Navigation Pane. For more information see [Creating User Interface Objects](#).

## Non (Included) User Interface

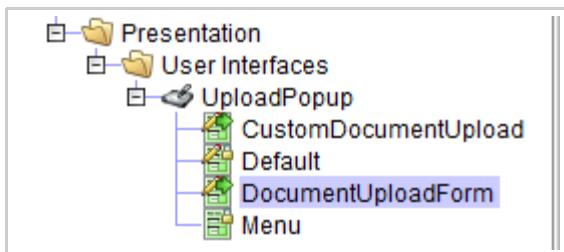
By adding an included (non-top-level) User Interface, the node **User Interface** is found beneath the including metadata object in the **Navigation** pane. Each of these metadata objects have exactly one User Interface; you cannot create multiple User Interface or remove it per each metadata object. However, you can create multiple [top-level User Interface](#) in the **Presentation** metadata type to create separate User Interfaces.

Included (non-top-level) User Interfaces are also available to create specific runtime interface appearances. The full pathname for each object is `com.conceptwave.<object_name>`.

User Interface Object	Description
<a href="#">BaseFinderUserInterfaces</a>	Contains base layout as well as functionality of a Finder
<a href="#">ConfirmDialog</a>	User Interface with default confirmation dialog
<a href="#">DocumentUserInterfaces</a>	Generally embedded under layouts, a document contains default menu for a document
<a href="#">Download</a>	Base class for download object
<a href="#">FinderUserInterface</a>	Contains default behavior User Interface for non-split finders
<a href="#">MasterDetailUserInterface</a>	Contains functionality for detail viewing for download object
<a href="#">SplitFinderUserInterface</a>	Contains a default User Interface for a SplitFinder
<a href="#">TabUserInterface</a>	Used for Navigation Tree
<a href="#">UploadPopup</a>	Contains functionality for uploading documents.
<a href="#">UserInterface</a>	Empty User Interface
<a href="#">Window</a>	Contains default functionality of a dialog and default User Interface for help content

## UploadPopup User Interface Objects

A com.conceptwave.system.UploadPopup provides default forms for uploading documents. There are two forms available that can be extended: CustomDocumentUpload and DocumentUploadForm.



### DocumentUpload Form

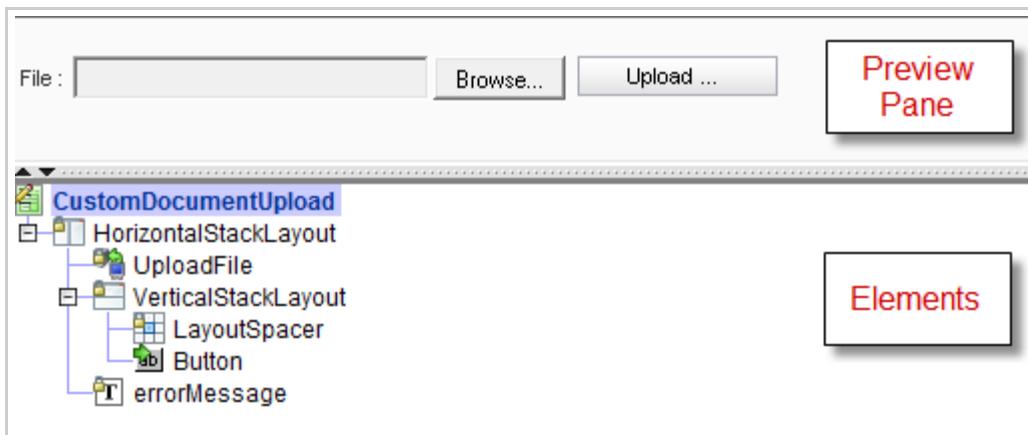
The DocumentUploadForm contains the UploadFile and errorMessage elements.



Each element can be replaced by "right-clicking" the element and selecting **Copy & Replace**. By performing a copy and replace, the properties of the elements are exposed and can be modified. The look and feel of the popup button can be changed by setting the **Button Style**. However, neither the **File** text nor **Browse** button can be changed as they are browser controls.

### CustomDocumentUpload Form

The CustomDocumentUploadForm contains the UploadFile and errorMessage elements. It also contains a separate Button element from the UploadFile element. The Button element is located beside the UploadFile element.



In this form, the **Show Upload** button property of the UploadFile element is set to *false* such that the **Upload** button associated with the UploadFile element is hidden. Instead, the Upload button that is displayed in the preview pane is the button associated with the Button element. The **Upload Control Name** property of the Button element can be set to the upload file element name that you want associated with this button.

## Documents

---

A Document is the container for Data Type objects called Document *Variables*. A Document may be extended from another Document and, in this case, may add more Variables to the base Document.

Essentially, a Document is used for the following purposes:

- To display or collect data from the UI (browser).
- To read or write data from database.
- To hold input or output parameters for external interface communication.
- To hold the search criteria and results for Finder search operations.
- To hold internal data for Decision Trees (wizards).
- To hold instance data of runtime process instances.

The most common use of Document is to present to the user in UI (that is, it can display itself in a browser and you can enter data in it from the browser). Additionally, a Document can be mapped to the following:

- The database.
- Another Document or Data Structure.
- A text file.

The UI presentation of a Document is defined through *Forms*. Forms allow different subsets of the Document data to be displayed in different presentation styles and to be used in different runtime conditions. This may depend on the user privileges, Order states, Document content, process operations and data from other runtime objects. The Document Variables displayed in the form are called *form fields*. Accessing Document data and its presentation in Forms are connected and controlled by the Document's *User Interface* object.

Document content can be manipulated via JavaScript scripts using pre-defined rules that are defined in **Methods** tab of the Document, such as Initialization Methods which are used to set initial values for any Variables in the current Document.

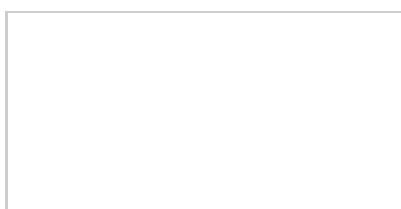
### Create a New Document

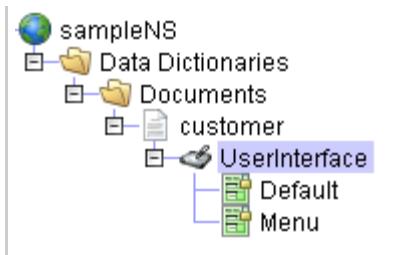
To create a new Document, do one of the following options:

- In the **Metadata** tab of **Navigation** pane, either:
  1. Right-click a Namespace and select **New ...**
  2. **New Metadata Object** wizard appears as a popup. Expand **Data Dictionaries**, select **Document**, and then click the **Next** button.
  3. Step two of the wizard appears. Enter the name of Document (must conform to JavaScript naming conventions). For **Extends**, choose **com.conceptwave.system.Document** to create a new Document. To create a user interface, select the **Create UI** option. Then, click the **Finish** button.
- OR
  1. Right-click **Documents** folder or the **Data Dictionaries** folder in a Namespace, if it is present. Select **New Document**.
  2. Follow step 3 from the previous procedure.

**Note:** To prevent problems from surfacing in JavaScript and SQL, checks occur to ensure that documents or document variables carrying names that match a set of reserved words are not used during development.

After the Document is created by the wizard, you can no longer change the base Document that extends it. The newly created Document is added under **Documents** folder. Beneath the Document node is the Document's User Interface node, which subsequently contains the Forms of the Document.





**Note:** To add or remove a user interface, right-click your document and select **Create UI** or **Delete UI**, respectively.

## Document General Properties

The general Document properties are on the **General** tab.

General   Variables   Methods   Mapping   Alias

Name:   Private  Restricted  Deprecated  Virtual  Final

Label:  ...

Description:

Extends:  ...

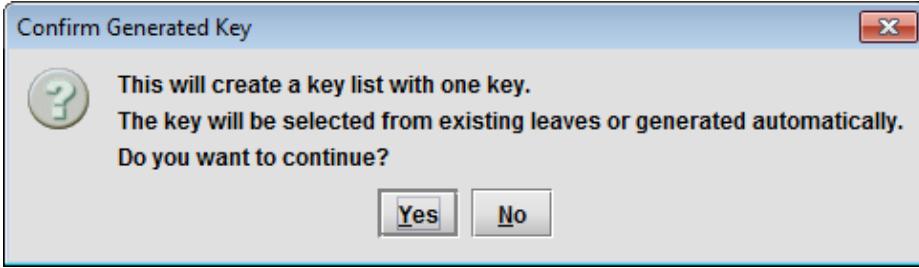
Overrides:  ...

Dynamic document:  Dynamic table name:   
 Audit trail  Attachments  Generated key  Timestamp sync

Keys:

The following table describes the fields for **General** tab of Document:

Field	Description
<b>Name</b>	Name of the metadata object (used in scripts; must conform to JavaScript naming conventions). Read-only after creation. To change it, use the <b>Rename</b> command in the popup menu by right-clicking the metadata object.
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Virtual</b>	If checked, it provides the ability to create methods in template metadata that does not have implementation, but can be used in template script and be bound to UI elements. The implementation is done in user metadata (see <a href="#">Virtual Field</a> for more information).
<b>Final</b>	Metadata elements can be marked as final, which includes any Boolean final property that is available on any metadata element. Additionally, any metadata element that has the <b>Final</b> property set to true can neither be extended, nor overridden.
<b>Label</b>	Metadata object display label.
<b>Description</b>	Description of the metadata object for documentation.
<b>Extends</b>	Base Document to derive from. Unchangeable after Document creation.

<b>Overrides</b>	Metadata object to override. Typically used to customize system or library metadata objects. Visible only if this Document extends from another Document. At runtime, the Document specified in <b>Overrides</b> field is completely replaced by this Document. A search for the overridden Document will return this Document as the result.
<b>Dynamic Document</b>	Boolean. If enabled, the Document becomes a dynamic Document, which may have dynamic Variables, capable to be added at runtime. See topic <a href="#">Dynamic Document</a> for details. Default to false.
<b>Dynamic table name</b>	The database table name of the name-value pair (NVP) table that stores values of dynamic Variables. Only relevant when <b>Dynamic Document</b> is checked. See topic <a href="#">Dynamic Document</a> for details.
<b>Audit trail</b>	If checked, the system will be triggered to record a history for any changes of the content of any runtime instance of this Document. Once it is checked, any change made to the Document will be recorded, providing the audit is requested in the Configuration application, and the <b>Audit</b> flag of the corresponding Document Variable is also checked. The changes are recorded into the <i>cwaudittrail</i> and <i>cwfieldaudittrail</i> database tables, which are system tables used as a history log in the database (refer to the note that follows this table).
	<p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• A maximum of 100 fields per document can be audited.</li> <li>• Translation fields are audited. The old and new values in the <i>CWFIELDAUDITTRAIL</i> table indicate the language of the change, and takes the format &lt;<i>language code</i>&gt; &lt;<i>translation text</i>&gt; (for example, en-xx HELLO).</li> <li>• For a dynamic document, a single operation creates one audit field record in the <i>CWAUDITTRAIL</i> table for both static and dynamic variables.</li> </ul>
<b>Attachments</b>	If checked, indicates that attachments to this Document are allowed at runtime. The attached files will be saved into the database. It is recommended that the attached files be smaller than 2 Mbytes to minimize performance degradation. Available only when Keys are defined.
<b>Generated key</b>	If checked, indicates that the system automatically generates the key value for this document. When checked, the Confirm Generated Key dialog appears.
	 <p>If you click the <b>Yes</b> button, Velocity Studio tries to locate, within existing variables, a variable with the <i>cwf.cwDocId</i> data type to use as a key. If such a variable is not found, the key is automatically created and added to the variable list of the document. This variable is added to the key list of the document and all other keys are removed.</p>
<b>Timestamp sync</b>	The runtime system uses an optimistic strategy to detect situations when a Document is updated simultaneously by two or more users and/or scripts, called <i>update conflicts</i> . To do this, each database transaction that saves the Document will check certain database field to decide if the Document has been changed from the moment it had been read into the memory. Usually the Variable containing the namespace type <i>System doc stamp</i> is used for update conflict detection. However, if such a Variable does not exist and this flag is checked, the Variable of the namespace type <i>System last updated timestamp</i> will be used for update conflict detection if it exists in the Document.
<b>Keys</b>	If the <b>Generated key</b> checkbox is unchecked, this field defines the list of Document Variables used to uniquely identify a Document Instance. To define keys, click the <b>Add</b> button and select Variables from the list in the <b>Document Tree</b> dialog that opens. To remove an existing key, select the key and click the <b>Remove</b> button. When the <b>Generated key</b> checkbox is checked, this field contains the system-generated key and is disabled so that it can neither be changed, nor removed. A key must be defined for the following types of Documents: <ul style="list-style-type: none"> <li>• A Document that is mapped to a database.</li> <li>• A Document that is result of a Finder.</li> <li>• A Document that is result Document or rule Document of a Finder rule.</li> </ul>

**Note:** Setting up an **Audit trail** for a Document will slow system performance, since extra information will be written to the database if necessary. If an audit trail is set up, to view the log content you can either use database tools or create a Finder object for table *cwaudittrail*, table *cwfieldaudittrail* or table view *AuditDetails* (which is a normalized view of *cwaudittrail* and *cwfieldaudittrail*). To create Finder, first import the table by selecting the [Import DB Schema](#) command from the **Database** menu to establish it as a Document in the metadata, then create a [Finder](#). The column description of *AuditDetails* is as follows:

Column	Description
<b>DOCID</b>	Document key.
<b>DOCMETADATATYPE</b>	Document metadata name.
<b>UPDATEDBY</b>	The user who made the change.
<b>LASTUPDATEDDATE</b>	The date the change was made.
<b>OPERATIONTYPE</b>	Can be: Add, Del(elete), Upd(ate).
<b>ATTRIBUTENAME</b>	Document Variable name.
<b>NEWVALUE</b>	New value.
<b>OLDVALUE</b>	Old value.

## Document Variables

The **Variables** tab is used to add Variables into the Document. The **Variables list** contains one entry for each variable added to the Document.

**Note:** If the Document extends another Document (a result of selecting the **Extend...** command from the **Documents** pop-up menu when creating a new Document), the Variables from the base Document will be shown in blue color. Newly added Variables will appear in black color.

The screenshot shows the 'Variables' tab selected in a tool window. The main area displays a table of variables with columns for Name, Type, Methods, Vis, Opt, and Edit. The 'Encrypt' variable is highlighted in blue, indicating it is a base document variable. A detailed view pane below shows the properties for 'Encrypt': Name (Encrypt), Data type (cwt\_pcdd.string10), and Element properties (<Inherited>). A table at the bottom lists element properties with columns for Name and Value.

Name	Type	Methods	Vis	Opt	Edit
firstName	com.conceptwave.system.String(String)	<input type="checkbox"/>			
Encrypt	cwt_pcdd.string10(String, 10)	<input checked="" type="checkbox"/>			
lastName	com.conceptwave.system.String(String)	<input type="checkbox"/>			

**General** **Methods**

**Name:** Encrypt **Constant** **Audit**

**Description:**

**Data type:** cwt\_pcdd.string10

**Type error code:**  **Mandatory error code:**  **Encrypt**

**Element properties:** <Inherited>

Name	Value

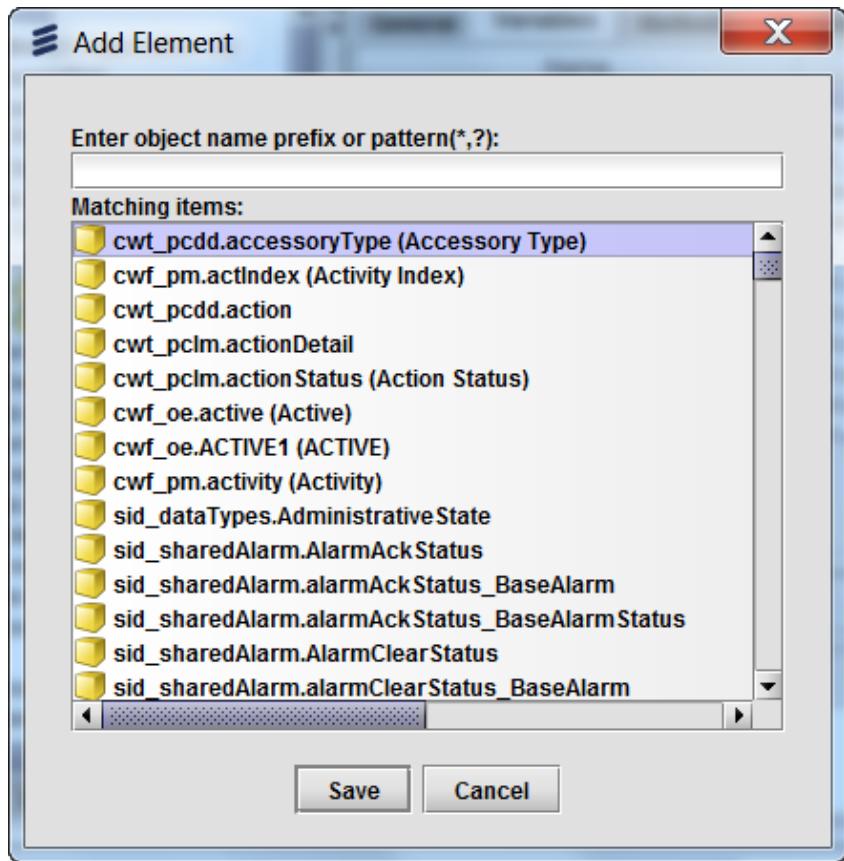
The Document Variable details appear in the detail pane when a Variable entry is highlighted in **Variables list**. The Document Variable properties can be found in the [section that follows](#). Some of the properties appear in the columns of the **Variables list** table.

- The **Type** column is populated with the primitive data type of the Variable's selected Data Type (for example, String), its actual data type (for example, com.conceptwave.system.String), and its length, if it is specified.
- The **Methods** checkbox is checked whenever the Document Variable contains one or more Methods.
- The **Vis**, **Opt** and **Edit** fields denote the **Visible**, **Optional**, and **Editable** Element properties, respectively.

The rows in the **Variables list** table can be moved up or down the list using the up and down arrow buttons located beside the table. Highlight an item, then click either the up arrow or down arrow button to move it. They can also be sorted alphabetically by clicking the **Sort** button .

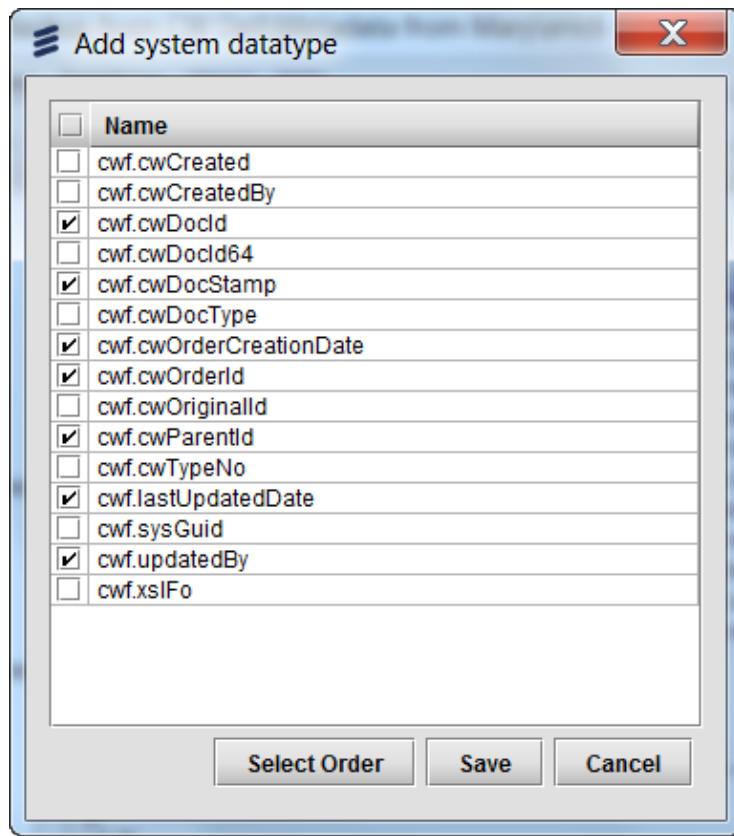
The **Add** button is used to add a Data Type as a Document Variable in the **Variables list**. When the user clicks the **Add** button, the **Add**

Element dialog will open.



The dialog contains a list of Data Types under all namespaces in the metadata. To add Variables, highlight the elements, and click the **Save** button. The dialog closes and the selected elements are added as Variables in the **Variables list**.

There are also several system-defined fields in the cwf namespace that can be added to the Document as Variables. Each system-defined field can exist in only one instance in the Document. The AVM automatically makes them read-only and assigns values to them. To add one or more system-defined fields to the Document, click the Add Sys button . The **Add system Variable** dialog will open.



Click the **Select Order** button to select the set of system-defined Variables required for a Document that is used in Orders. Alternatively, select the individual Variables from the list and click the **Save** button. The dialog will close and the Variables will be added to the **Variables list**. The list of system-defined fields are described below.

System Field	Name	Description
Document ID	cwf.cwDocId	System-generated unique Document key.
Document ID 64 bit	cwf.cwDocId64	System-generated unique Document key for 64 bit system.
Document order ID	cwf.cwOrderId	ID of the Order if the Document is part of an Order. Applies only to Documents with a system generated key.
Document parent ID	cwf.cwParentId	The immediate parent ID of the Document if the Document is part of an Order. Applies only to Documents with a system generated key.
Last update timestamp	cwf.lastUpdatedDate	The timestamp of the last update or create operation.
Last updated by	cwf.updatedBy	The ID of the user who performed the last update or create operation.
Created by	cwf.cwCreatedBy	The ID of the user who created the Document.
Creation timestamp	cwf.cwCreated	Document creation timestamp.
Document type	cwf.cwDocType	The fully qualified metadata name of the Document. Useful in the base Document of several derived Document types that are mapped to the same table because it allows the user to search for Documents of this particular type in this table.
Document type number	cwf.cwTypeNo	The system number of the metadata Document. Useful in the base Document of several derived Document types that are mapped to the same table because it allows the user to search for Documents of this particular type in this table.
Document stamp	cwf.cwDocStamp	The Document stamp used for conflict detection. Should exist if the Document is part of Order.
System GUID	cwf.sysGuid	Represents the globally unique identifier (GUID) for this object.
System order creation	cwf.cwOrderCreationDate	The system time on order creation.

<b>timestamp</b>		
<b>System timestamp</b>	cwf.cwTimestamp	The system field for a timestamp data type, which is based on com.conceptwave.system.Timestamp.
<b>XSL FO file</b>	cwf.xslFo	The XSL-FO file used for this Order in Order PDF Printing.

To remove Variables from the table, highlight one or more rows in the **Variables list** and click the Remove button . A Variable that is selected as a system-generated key cannot be removed.

These buttons beside the list follow the [Manage Array Elements](#) presentation convention in Velocity Studio. See this section for details about the aforementioned and additional buttons.

The CWPRODUCTPROPERTIES table contains the UB64 data value, which allows you to generate the database identifier as a base64 string. This data value can take any number that can ensure a unique AVM. You can use this value to generate a unique sequence ID for translation, a data object, and more.

### Document Variable Properties

The following describes each property of Document Variable, which is presented in two tabs: **General** tab and **Methods** tab.

#### *In General tab*

Column	Description
<b>Name</b>	This value is automatically set to the Data Type's original name. It may be overwritten when necessary.  <b>Note:</b> For this field, do not use <i>id</i> , as this name is reserved.
<b>Constant</b>	If checked, the property <b>Constant value</b> appears. It becomes a constant variable that has a fixed value specified in the Constant value.
<b>Audit</b>	If checked, the runtime system will log the variable update history in the database if the <b>Audit Trail</b> checkbox for the Document is checked in the <b>General</b> tab and the Configuration application has specified that audit is allowed. Any change made to the variable will be recorded into the system table CWFIELDAUDITTRAIL in the database.  <b>Note:</b> Translation fields are audited. The old and new values in the CWFIELDAUDITTRAIL table indicate the language of the change, and takes the format <language code> <translation text> (for example, en-xx HELLO).
<b>Data Type</b>	The data type of the Variable.
<b>Type Error Code</b>	Specifies an error message code that should be in the application resources. If set, the specified error message will be used instead of the default system message when the field contains an invalid value.
<b>Mandatory Error Code</b>	Specifies an error message code that should be in the application resources. If set, the specified error message will be used instead of the default system message when the field is mandatory and it has no value.
<b>Encrypt</b>	If checked, the leaf will be encrypted when saving the document and decrypted when loading the document. The encrypt option in the data type will be overridden. If not checked, the encrypt option in the data type will be overridden.
<b>Constant Value</b>	Appears only when <b>Constant</b> is checked. Specifies the value of the constant variable.
<b>Element</b>	The type of <a href="#">Form Element</a> that the Data Type shall be defaulted to present. Available choices depend on the base Data Type, and whether enumeration is defined.
<b>Element Properties</b>	List of properties that any Form Element of this Document Variable shall be defaulted to. These properties can be overridden at lower levels; see <a href="#">Permission</a> for details on how the properties are computed at runtime. Available properties to be inherited depend on the selected Element Type; they are: <ul style="list-style-type: none"> <li>o <b>Display Format:</b> Specifies, in regular expression, the format of how the field Element shall be displayed. The use of this property depends on the Element Type selected. See documentation of the selected Element Type for details.</li> <li>o <b>Optional:</b> Determines whether the field Element can have empty value (true) or a mandatory field (false). True or false, or None, or assign a Permission Method to determine the boolean result.</li> </ul> You cannot override the <b>Optional</b> property in the following cases: <ul style="list-style-type: none"> <li>▪ If the data structure leaf is <b>Mandatory</b> or the variable data type is not <b>Nullable</b>, at runtime, the <b>Optional</b> property is false.</li> <li>▪ If the data type is overridden and <b>Nullable</b> is not checked for the overridden type, at runtime, the <b>Optional</b></li> </ul>

property is false.

- If the **Optional** property is set to false in the **Element** property of the data type or at the document variable level, the element's **Optional** property remains set to false.

In other cases, the element's **Optional** property takes precedence over the **Optional** property defined at the document variable level.

- **Editable:** Determines whether the field Element is editable (true) in UI or read-only (false). True or false, or None, or assign a Permission Method to determine the boolean result.

**Note:** If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.

- **Visible:** Determines whether the field Element is visible in UI or not. True or false, or None, or assign a Global or Document Permission Method to determine the boolean result.
- **Label:** When this Variable is assigned to a Form Element, Label in the Form Element is defaulted to this value, to be displayed in runtime. This value is automatically set to the Data Type's **Label** value, and may be overwritten as necessary.

**Note:** A document that has a variable based on a data type, with element properties set to define a label, inherits the element properties from the data type. If it is not set, the document inherits the data type's element properties. If it is also not set, the document then inherits data type's label and the default control based on base data type (string, date integer).

*In Methods tab*

The screenshot shows the Velocity Studio interface for managing Document Variables and Methods.

**General Tab:**

Name	Type	Methods	Vis	Opt	Edit
firstName	com.conceptwave.system.String(String)	<input checked="" type="checkbox"/>			
lastName	com.conceptwave.system.String(String)	<input type="checkbox"/>			
active	com.conceptwave.system.Boolean(Boolean)	<input type="checkbox"/>			
duedate	com.conceptwave.system.Date(Date)	<input checked="" type="checkbox"/>			

**Methods Tab:**

**Method List pane:** Shows a tree view of methods under the 'customer' document variable. The selected method is 'cwLeafInitAction\$duedate'.

**Details pane:** Displays the details for the selected method:

- Name:** cwLeafInitAction\$duedate
- Leaf:** sampleNS.customer.duedate
- Description:** (empty)
- Script:**

```
function cwLeafInitAction(document) { // Leaf initialization rule. Used to generate
    return Calendar.today();
}
```
- Return:** sampleNS.customer.duedate

The Methods tab is a [script management screen](#) in Velocity Studio, which is composed of the **Method List pane** at the left which lists all scripts of the Document Variable, and **Details pane** which displays the details of a script when selected at the Method List pane. To create a new Method, or to override an existing Method of its base Document, right-click the Document in the Method List pane. A pop-up menu appears.



**Note:** This creates Method at the Document Variable level, impacting only to the Document Variable. To create Methods at the Document level, go to [Methods tab in Document](#).

The table below lists the type of Methods that can be added or overridden at the Document Variable level.

#### New Methods

Method Type	Description
Validation	Document Variable validation script that determines the validity of runtime data values of the Document. It is triggered with Document method ".validate", and by default, the generated Document Form's "save" menu button triggers such validation. See <a href="#">Document Validation Method</a> and the <a href="#">phasedValidation configuration variable</a> for details.

[System-defined Methods that can be defined](#)

Method Type	Script Name	Parameters	Return Type	Description
Initialization	<code>cwLeafInitAction\$&lt;Variable Name&gt;</code>	None.	Document Variable	<p>Document Variable Initialization script is to set initial values for the Document Variable.</p> <p>It is triggered when the Document is created. This Method is not executed when the Document is read from the database or an external system interface.</p>
Calculate	<code>cwLeafCalculateAction\$&lt;Variable Name&gt;</code>	None.	Document Variable	<p>Document Variable Calculate script is to compute a value for the calculated Document Variable. A calculated Document Variable is a Variable that is not stored in the database or sourced from external sources, but their values computed based on values of other Document Variables. Variables with calculate Method are read-only fields in the UI at runtime.</p> <p><b>Note:</b> Because the value of the calculated field is computed every time the field is accessed, it may be accessed many times for different purposes by AVM at runtime. Unwise and excessive use of calculated fields may introduce serious performance problems. It is recommended to limit the Method to simple calculations based on Document Variable values, and not invoke database access operations or external interface calls.</p>
Trigger	<code>cwLeafTriggerAction\$&lt;Variable Name&gt;</code>	None.	None.	<p>Document Variable Trigger script is executed when the content of a Document Form field is changed (and loses focus) at runtime. Trigger execution may come automatically from the UI or can come from a user script at runtime. When a trigger execution request is received, all triggers for Variables that has been changed are executed.</p> <p><b>Note:</b> As a Trigger Method is triggered at UI, a server round trip is required to send updates to the client; this is done in AJAX, with only changed variables updated to browser with no page refresh. Regardless, Trigger Methods should be used sparingly to avoid incurring this overhead.</p>

For Method Types that can be created, you can create multiple Methods of the same Method type (for example, Validation 1, Validation 2...), each identifiable by unique Method name. However, for Methods with system-defined names, or to be overridden from base Document, you can only define one Method per base Method.

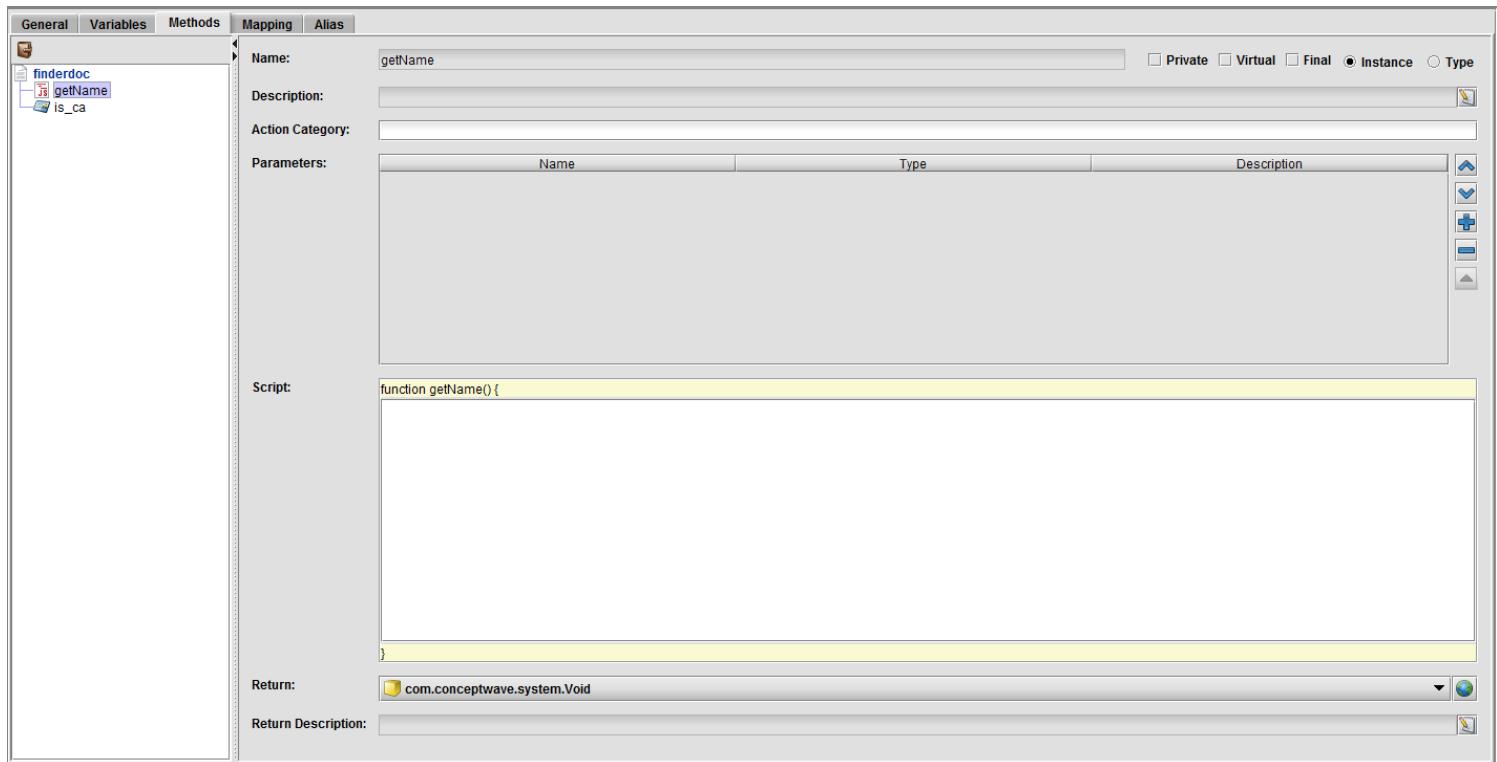
For scripts that are already overridden, they appear in the Method List pane and may no longer be available from the right-click context pop-up menu.

The Method created in Method List pane can be right-clicked, with various commands available; see [script management UI](#) for details.

## Document Methods

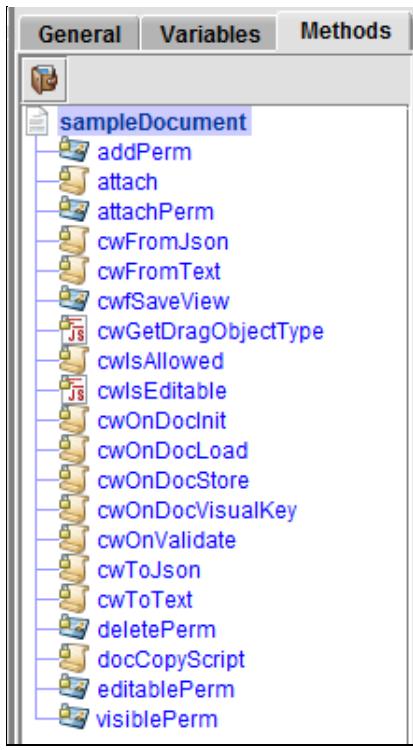
The **Methods** tab is used to add Document-level scripts. There are different types of Methods that can be defined for the Document; they are all consolidated into this tab.

**Note:** For Document Variable-level scripts, they can be managed under **Variables** tab, by selecting the Variable in the Variables List in that tab, and then select the **Methods** tab in the Variable detail pane to edit the Variable's Methods. These Variable-level Methods also appear and are editable on Document Method tab; these Methods can be differentiated from other Methods by their Method names, which are appended by \$<Variable Name>.

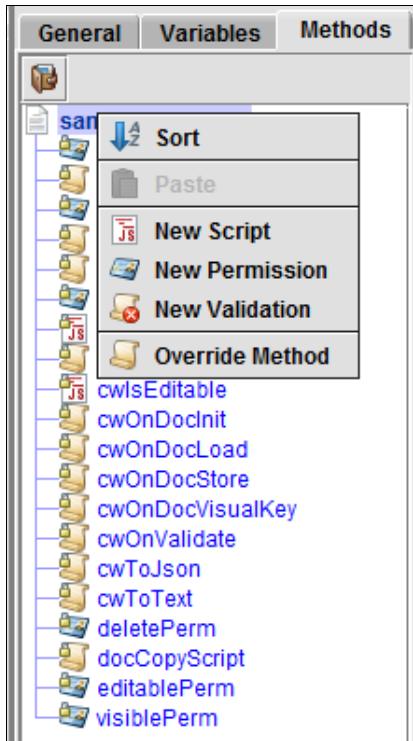


The Methods tab is a [script management screen](#) in Velocity Studio, which is composed of the **Method List pane** at the left which lists all scripts of the Document, and **Details pane** which displays the details of a script when selected at the Method List pane.

To show all base methods, click the **Show base methods** ( ) button. Click the button again to hide all base methods. Base methods appear in blue font, as shown in the following figure. Any method highlighted in blue indicates that you can override it. Methods in black font cannot be overridden.



To create a new Method, or to override an existing Method of its base Document, right-click the Document in the Method List pane. A pop-up menu appears.



When you select the **Type** radio button for a method, it means that the method can be invoked directly through the script and does not require an instance object.

If you have a long list of methods in the left pane, you can press the first letter of the method you are looking for to jump to the list of methods beginning with that letter. In this example, pressing the I key highlights the first instance of a method that begins with the letter I. The method displays in the right pane.

General	Variables	Methods	Mapping	Alias
	Name: <input type="text" value="is_ca"/>	<input type="checkbox"/> Private <input type="checkbox"/> Virtual <input type="checkbox"/> Final <input checked="" type="radio"/> Instance <input type="radio"/> Type		
	Description: <input type="text"/>			
	<input type="checkbox"/> Reference			

The table below lists the wide-ranging type of Methods that can be added at the Document level.

#### New Methods

Method Type	Description
	Script Generic user-defined JavaScript function that can be defined in a Document and explicitly invoked by other scripts; the function is not triggered by other built-in means in the Document.
	Permission Method that can be associated to determine permission for a certain property of the Document. (for example, Element properties). See the <a href="#">Permissions</a> section for details.
	Document validation script that determines the validity of runtime data values of the Document. It is triggered with Document method ".validate", and by default, the generated Document Form's "save" menu button triggers such validation. See the <a href="#">phasedValidation configuration variable</a> for details.

#### Some of the system-defined Methods

Method Type	Script Name	Parameters	Return Type	Description
<b>Initialization</b>	<i>cwOnDocInit</i>	None.	None.	Document Initialization script to set initial values for any Variables in the Document. It is triggered when the Document is created, after the Variables scripts have executed. This Method is not executed when the Document is read from the database or an external system interface.
<b>Load</b>	<i>cwOnDocLoad</i>	None.	None.	Document Load script that is executed once after the Document is loaded from the database.
<b>Store</b>	<i>cwOnDocStore</i>	None.	None.	Document Store script that is executed each time the Document is saved to the database, immediately before the database operation is performed.
<b>Visual Key</b>	<i>cwOnDocVisualKey</i>	None.	<i>String</i>	Script that shall return a string representing the visual key. The length of the string shouldn't be longer than 64 characters. If longer, the value is truncated. If this Method is empty, the Document Label is used as visual key.
<b>Document Copy</b>	<i>docCopyScript</i>	None.	None.	Script that is executed whenever the Document is copied.
<b>View Permission</b>	<i>visiblePerm</i>	None.	<i>Boolean</i>	View Permission that specifies whether the participant is allowed to view the Document.
<b>Add Permission</b>	<i>addPerm</i>	None.	<i>Boolean</i>	Add Permission that specifies whether the participant is allowed to create an instance of this Document (for example, Document instances in an Order Collection).
<b>Delete Permission</b>	<i>deletePerm</i>	None.	<i>Boolean</i>	Delete Permission that specifies whether the participant is allowed to delete an instance of this Document (for example, Document instances in an Order Collection).
<b>Update Permission</b>	<i>editablePerm</i>	None.	<i>Boolean</i>	Update Permission that specifies whether the participant is allowed to update the Document (for example, change Variable of the Document).
<b>Attach Permission</b>	<i>attachPerm</i>	None.	<i>Boolean</i>	Attach Permission that specifies whether attachments are allowed to be attached to the Document.
<b>Attach</b>	<i>attach</i>	None.	None.	
<b>Get Drag Object Type</b>	<i>cwGetDragObjectType</i>	None.	<i>String</i>	Returns the metadata object's type. The purpose of this method is to allow metadata to provide a different type for the object than what is used by the product.
<b>cwOnChangedFromTable</b>	<i>cwOnChangedFromTable</i>	State	None.	This method takes a single integer parameter. The following is

the explanation of values:

- When set to 0, this document instance is changed within a dynamic table.
- When set to 1, this document instance is added to a dynamic table.
- When set to 2, this document instance is deleted from a dynamic table.

This method is triggered when you edit a row in a dynamic table, or when you delete a row in a dynamic table using the default per-row Delete button, or when you add a new row in a dynamic table using the default Add button in the dynamic table.

**Note:** This method is not invoked when you remove or add new rows to the dynamic table through APIs rather than using the default UI buttons.

You can override the default empty implementation to add logic to handle the data changes in a dynamic document through editing the dynamic table, or when the instance is removed or added to the dynamic table through the default UI buttons. This implementation covers the scenario where you have no way to get the notifications when dynamic document is modified through the dynamic table.

For Method Types that can be created, you can create multiple Methods of the same Method type (for example, Validation 1, Validation 2...), each identifiable by unique Method name. However, for Methods that are to be overridden from base Document, you can only define one Method per base Method; this is applicable to both system-predefined scripts (that is, from *com.conceptwave.system.Document*) and any additional Methods that are defined in non-system-based base Document. All of the system-predefined Methods *com.conceptwave.system.Document* have empty scripts except Permission Methods, which means the default Method is to do nothing. All Permission Methods of *com.conceptwave.system.Document* return *true*, which allows all operations on the Document by any participant.

For scripts that are already overridden, they appear in the Method List pane and no longer be available at the right-click context pop-up menu.

If this Document extends from another (non-system based) Document, and the base Document has additional Methods defined in it, those Methods are not displayed in the Methods tab of this Document.

The system-defined Document-level Permission Set above (*permView*, *permAdd*, *permDelete*, *permUpdate*, *permAttach*), in essence, govern the Document's Permission. That is, whenever the Document is acted upon (create, read, update, delete...) by a participant, the corresponding Permission Method is invoked by AVM to determine whether the operation is allowed or not. Conversely, the "new" Permission Methods that can be added are not invoked unless explicitly associated with a Permission-related Property (for example, Element Properties in Document Variables).

The Method created in Method List pane can be right-clicked, with various commands available; see [script management UI](#) for details.

## Validation Method

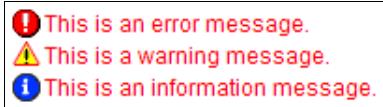
A Document Validation Method is a user-defined script that determines the validity of runtime data values of the Document. It can be triggered with Document method ".*validate()*". By default, the generated Document Form's "save" menu button triggers such validation.

**Note:** Document Variable-level Validation Methods can be defined under **Variables** tab, by selecting the Variable in the Variables List in that tab, and then select the **Methods** tab in the Variable detail pane to edit the Variable's Methods. These Variable-level Validation Methods also appear and are editable on Document Method tab.

When a Validation Method is triggered and evaluated to true, the user is presented with a message of an assigned severity.

There are three levels of severity:

- Information
- Warning
- Error



At runtime, all Validation Methods defined in the Document (and its base Document) are evaluated. The maximum severity of all Validation Methods that are "fired" are to collectively set the Document status as shown in table below.

Document Status	Collective Method Severities
Valid	None.
Information	Information or warning only.
Invalid	Any error.

See the [phasedValidation configuration variable](#) for validating all variables and document rule scripts even when there are errors for mandatory fields or at the data type level.

Add, edit, and remove Validation Methods in the Document's **Methods** tab.

 newDoc  PM  permView  validateCustomer	<p><b>General</b></p> <p><b>Name:</b> validateCustomer</p> <p><b>Description:</b></p> <p><b>Script:</b></p> <pre>function null() { // Metadata type method. Can be called by scripts.   var cust = this.custRef._refData;   if (this.newCompanyName == cust.compName)     return true;    return false; }</pre> <p><b>Return:</b> com.conceptwave.system.Boolean</p> <p><b>Severity:</b> warning</p> <p><b>Message:</b></p> <p><b>Message Code:</b> AW0034</p>
---	--

Field	Description
<b>Name</b>	Name of the Method. Must be unique within the Document.
<b>Description</b>	Descriptive text for documentation purposes.
<b>Script</b>	Validation JavaScript script. The script can be one of the following: <ul style="list-style-type: none"> <li>An expression that evaluates to true to indicate that the text in <b>Message</b> field or the message corresponding to the specified <b>Message Code</b> is to be displayed. Otherwise it must evaluate to false.</li> <li>A script that returns true to indicate that the text in <b>Message</b> field or the message corresponding to the specified <b>Message Code</b> specified is to be displayed. Otherwise it must return false.</li> <li>A script that returns a resource identifier string or list of identifiers separated by semicolons to indicate the resource message(s) to be displayed. For example: AW0155 or AW0155;AE0156. We recommend that the first character must be A. The second character indicates the message type: I - information, W - warning, and E - error.</li> <li>A JavaScript script returning text (that is not a resource identifier) that will be displayed as a rule message.</li> </ul>
<b>Return type</b>	The Object type to be returned by the script.
<b>Severity</b>	Indicates the severity of the Method: information, warning or error.
<b>Message</b>	Message to display at runtime when this Method is evaluated and the condition is met. Plain text. May include the placeholder {1} for the Document visual key. If this is a Document Variable Validation Method, it may include the place holder {0} for the field label. It is mandatory that one of either the <b>Message</b> or <b>Message Code</b> fields are specified. If both the <b>Message</b> and <b>Message Code</b> fields are specified, the <b>Message Code</b> will be ignored.
<b>Message Code</b>	Defines the message code. It is mandatory that one of either the <b>Message</b> or <b>Message Code</b> fields are specified. If both the <b>Message</b> and <b>Message Code</b> fields are specified, the <b>Message Code</b> will be ignored.

Document validation methods permit setting breakpoints within Velocity Studio for [debugging](#). Velocity Studio translates a breakpoint set on a validation rule to a breakpoint on the generated validation script. Whether a document has one or more validation methods, setting or unsetting the

breakpoint on any of them also sets or unsets a breakpoint on the combined script, which is named <*document name*>.cwOnGenDocValidation. The individual scripts are visible within that combined script.

## Mapping Tab: Database Mapping

The **DB Map** sub-tab in the **Mapping** tab is used to connect Document Variables to database table columns. Documents are only mapped to database tables when it is necessary for their data to be stored in the database. A Document Variable will correspond to a database table column, but it is not mandatory that all Variables will be mapped to that database.

A Document may be mapped to more than one table if the Document has one key and all the database tables that it is being mapped to have the same key column (with exactly the same name and attribute in each database table). A Document cannot be mapped to more than four tables. Moreover, if a document is mapped to multiple tables and contains system field *cwordercreationdate*, all tables must contain the column CWORDERCREATIONDATE.

It is important to note that the mapping to database will not create the database table(s) if they do not already exist. The SQL statements to create the tables can be generated by the **Upgrade System** command. If the **Skip DB schema generation** check box is selected on the **DB Map** sub-tab, then the corresponding DDL statements will be excluded from the SQL script.

The **DB Map** sub-tab will be in one of two states the first time you open it.

- If the Document is derived from a base Document that has DB mapping, the Document will inherit the database schema and table mapping from the base Document. The DB mapping of the base Document will be shown in blue and neither the schema nor mapping can be changed or removed from the derived Document.
- Otherwise, the **Disconnected** icon will be shown and all fields and tables will be blank.

**Note:** In the case of connecting with non-Oracle databases, it is your responsibility to synchronize case-sensitivity.

The screenshot shows the **DB Map** sub-tab with the following interface elements:

- Logical connection:** ORDER
- Status:** Connected (checked)
- Checkboxes:** Skip DB schema generation (unchecked), Map to separate table (unchecked)
- Tables List:** A tree view showing the **customers** table with its columns: CWDOCID, ID, NAME, DIVISION, PHONE, and ACTIVE.
- Mapping Table:** A grid showing the mapping between document variables and database columns. The columns are Leaf, Type, and Don't Map.

Leaf	Type	Don't Map
cwDocId	String, 16	<input type="checkbox"/>
id	String, 10	<input type="checkbox"/>
name	String, 30	<input type="checkbox"/>
division	String	<input type="checkbox"/>
phone	String	<input type="checkbox"/>
active	Boolean	<input type="checkbox"/>

- Buttons:** Rename, Remove, Map All Unmapped, Unmap, <- Map ->

### Logical Connection

The **Logical Connection** field specifies the logical connection at which the Document's tables may be found.

If the project has not yet connected to database, the **Connect** button appears beside the **Logical Connection** field. Click it to connect to

database via **Database Login** dialog, which is the same method as if invoking via **Database / Connect** menu command. If the connection is successful, the connection will appear in the **Logical Connection** field, and the connection state icon will change from **Disconnected** to **Connected**.

Once connected to database, choose from the list of logical connections available, which is the same list of [logical connections in Configuration application](#).

If a database mapping already exists, you may remove the connection and database mapping by clicking the **Clear** button. The **Confirm Database Clear** dialog will open. If the **Yes** button is clicked, you will be disconnected and the database mapping will be removed.



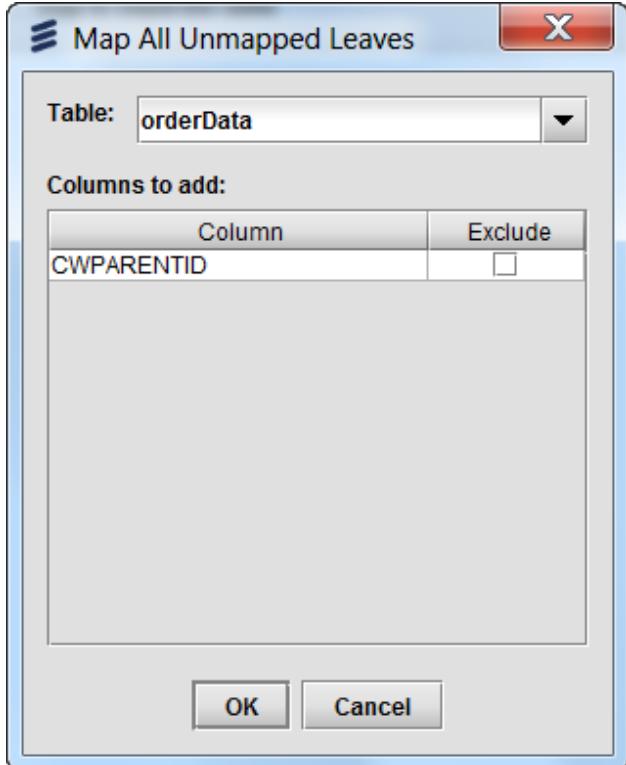
## Database Mapping

A new DB mapping can be created either by selecting the **Map to DB** item from the Document right-click pop-up menu or by clicking the **Map All Unmapped** button on the **DB Map** sub-tab.

When the **Map to DB** item is selected from the right-click pop-up menu, the default DB mapping will be created with automatically generated table and column names. The **Database schema** must be specified in the Document before this action, whether it is a new or existing one. In the case of a derived Document, any additional Variables will be mapped to the base Document table if the **Map to separate table** check box is unselected. If the **Map to separate table** check box is selected, the additional Variables will be mapped to a separate table. If the Document has more than one base Document with database mappings, the additional Variable will be mapped to a separate table regardless of the **Map to separate table** check box state.

When using the **Map All Unmapped** button to create a DB mapping, neither a database connection nor a new **Database Schema** is required. However, it is recommended that a database schema is either created first or connected too.

When the **Map All Unmapped** button is clicked, the **Map All Unmapped Leaves** dialog will open. The dialog generates a default table name based on the Document name and default column names based on the Variable names.



All generated names in the dialog can be updated. The table name can be modified in the **Tables** combo box. When connected to a database, the **Tables** combo box will also contain database table names for selection. The column names in the **Columns to add** table can be edited in

place. The check box in the **Exclude** column should be selected to not generate a column for the Variable. This allows you to map excluded Variables later to another database table. When connected to a database, the **Match existing DB columns only** check box will be visible and, if selected, only Variables that match existing database column names will be mapped. This allows you to manually map the rest of the Variables if the database table already contains the required columns but with different names. Click the **OK** button to add the generated table and columns to the **Tables** tree.

The screenshot shows the DB Map interface with the following components:

- Top Bar:** DB Map, Text Map, XML Map.
- Logical connection:** ORDER (dropdown menu) and Clear button.
- Status:** Connected (checked), Skip DB schema generation (unchecked), Map to separate table (unchecked).
- Tables Tree:** Shows a tree structure under the 'Tables' node, with 'customers' expanded. Columns include: CWDOCID (highlighted in red), ID, NAME, DIVISION, PHONE, and ACTIVE.
- Variables Table:** A grid showing variables and their properties:
 

Leaf	Type	Don't Map
cwDocId	String, 16	<input type="checkbox"/>
<b>id</b>	String, 10	<input type="checkbox"/>
<b>name</b>	String, 30	<input type="checkbox"/>
<b>division</b>	String	<input type="checkbox"/>
<b>phone</b>	String	<input type="checkbox"/>
<b>active</b>	Boolean	<input type="checkbox"/>
- Buttons:** <- Map -> (centered between the tree and the variables table), Rename, Remove, Map All Unmapped, and Unmap.

All database tables used in the mapping are shown on the left side of the tab in the **Tables** tree with their columns. The database tables and their columns may or may not physically exist in the database, therefore, icons are used in the tree to reflect the items' state.

Icon	Description
	A table that physically exists in the database.
	A table that does not exist in the database.
	A column that physically exists in the database.
	A column that does not exist in the database.

When **Disconnected**, all tables and columns are shown as not existing in the database. When a connection is established, the tables and columns from the **Tables** tree will be matched against the existing database structure and their icons will be updated accordingly. The columns in the database will also show a tooltip with their data type information. The **Rename** and **Remove** buttons are used to rename or remove tables and columns from the tree. If the **Remove** button is clicked, any existing mapping to these table(s) or column(s) will be removed as well, after a confirmation message.

All Document Variables are shown on the right side of the tab in the **Variables** table. The **Leaf** column shows the Variable label and icon for regular Variables and keys. The **Type** column shows the Variable data type information. The **Don'tMap** column contains check boxes that can be used to exclude a Variable from DB mapping.

**Note:** Fields that are mapped to data structure variables appear in bold and have an asterisk (\*) at the end. If you find a data structure variable that is displayed with double asterisks (\*\*), remove the extra \* from the label in the metadata.

Color conventions are used for the items in both the **Tables** tree and **Variables** table to reflect if the item is mapped or unmapped and if the item belongs to a base or derived Document.

Color	Meaning
Blue	Mapped and belongs to base Document.
Brown	Mapped and belongs to this Document.
Grey	Not mapped and belongs to base Document.
Black	Not mapped and belongs to this Document.

The **Map** button is used to map an unmapped Variable in the **Variables** table to an unmapped column in the **Tables** tree. When a mapped item is selected in either tree or table its corresponding item will be selected on the other side. To unmap a Variable, select the Variable in the **Variables** table and click the **Unmap** button.

## Mapping Tab: Document to Text Mapping

The **Text Map** sub-tab in the **Mapping** tab allows the user to specify how to import/export Documents as text. Table below contains descriptions of the fields in the **TextMap** tab.

Field	Format	Length
basketID		
customerReference		
cwOrderId		

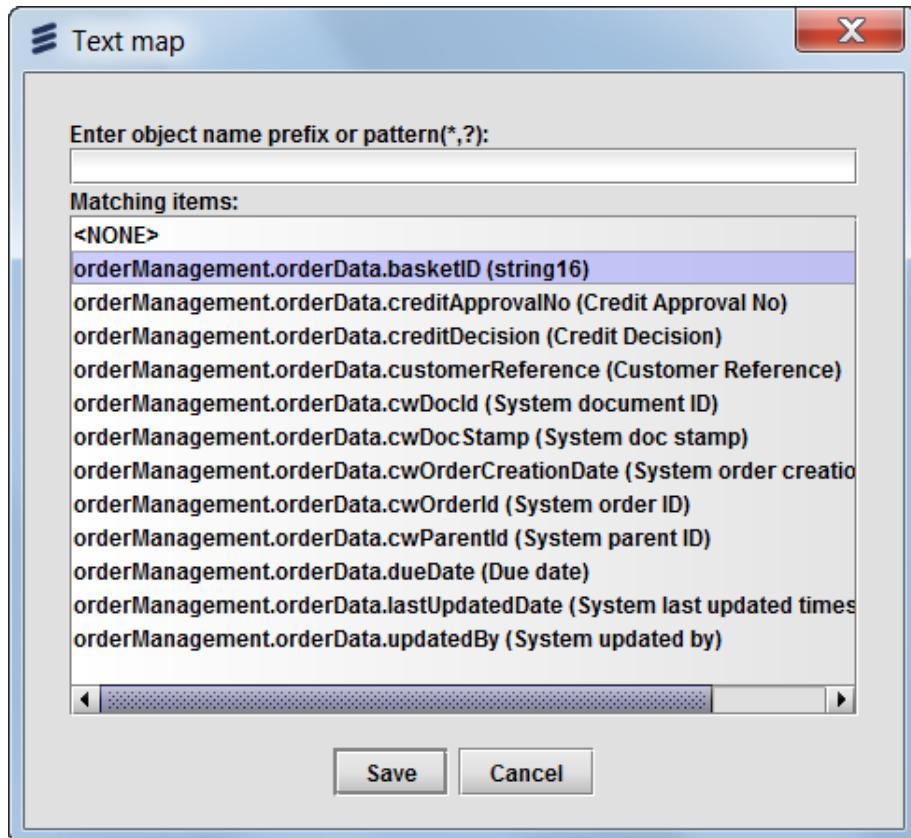
Field	Description
<b>Record type</b>	Specifies <b>Fixed</b> or <b>Variable</b> length records. If <b>Fixed</b> is selected, then every data field will have a data length defined. If <b>Variable</b> is selected, then the record fields will be separated by the separator string specified in the <b>Separator</b> field and the records will be separated by character specified in <b>Terminator</b> field.
<b>Separator</b>	String used to separate the record fields for <b>Variable</b> record type format (for example, comma ,).
<b>Quotation</b>	Character denoting quoted text (for example, single quote ' or double quote ). For <b>Variable</b> record format only
<b>Terminator</b>	String appended to the end of each record during export that denotes the end of a record. (for example, new line \n). For <b>Variable</b> record format only.
<b>List of fields</b>	The list of Document fields that will be exported/imported to/from the text file. May not include all Document Variables.

### List of Fields

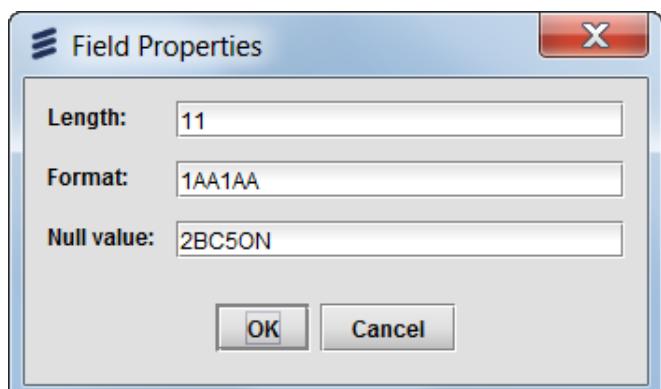
The list of fields may be a subset of the Document variables In addition, a **Field** labeled none can be added to the list to allow fixed text output.

Click the **Add** button to add a field from the Document Variables. The **Text map search dialog** appears; highlight the item to be added and click

"Save" to add the field.



Click the **Properties** button, the **Field Properties** dialog will open where the user can enter the field length, format and null value (only the **Length** field will be visible for none field items). If the length and/or format is defined, it will override the values defined in the **Data Types** properties. The **Length** is mandatory for **Fixed** record types and for none field items.

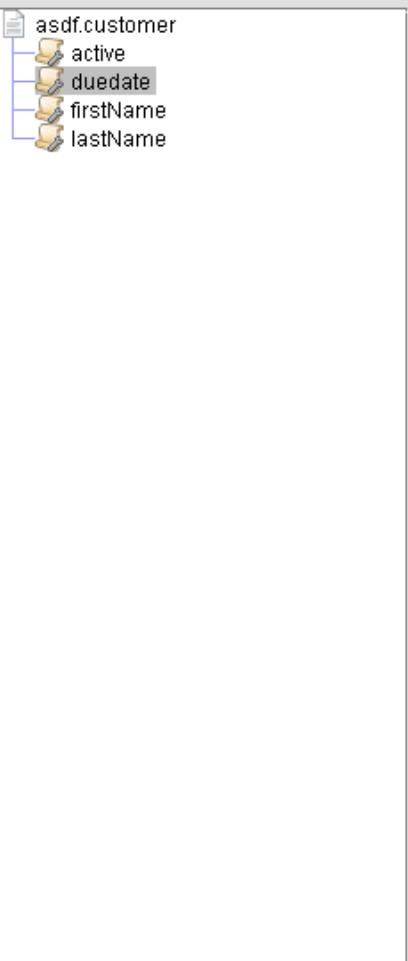


The rows in the list of fields can be reorganized using the up and down arrow buttons located beside the table. Highlight an item, then click either the up or down button to move it. The order in the list will determine the order that the fields are imported/exported to/from the text file.

## Mapping Tab: Document to XML Mapping

The **XML Map** sub-tab in the **Mapping** tab specifies how to export/import Documents to/from XML format. Table below contains a description of the fields in this tab.

**DB Map**
**Text Map**
**XML Map**



**XML Settings**

**XML name:**

**Length:**

**Format:**  [...]

**Exclude**    **Attribute**    **CDATA**

**Update** **Remove**

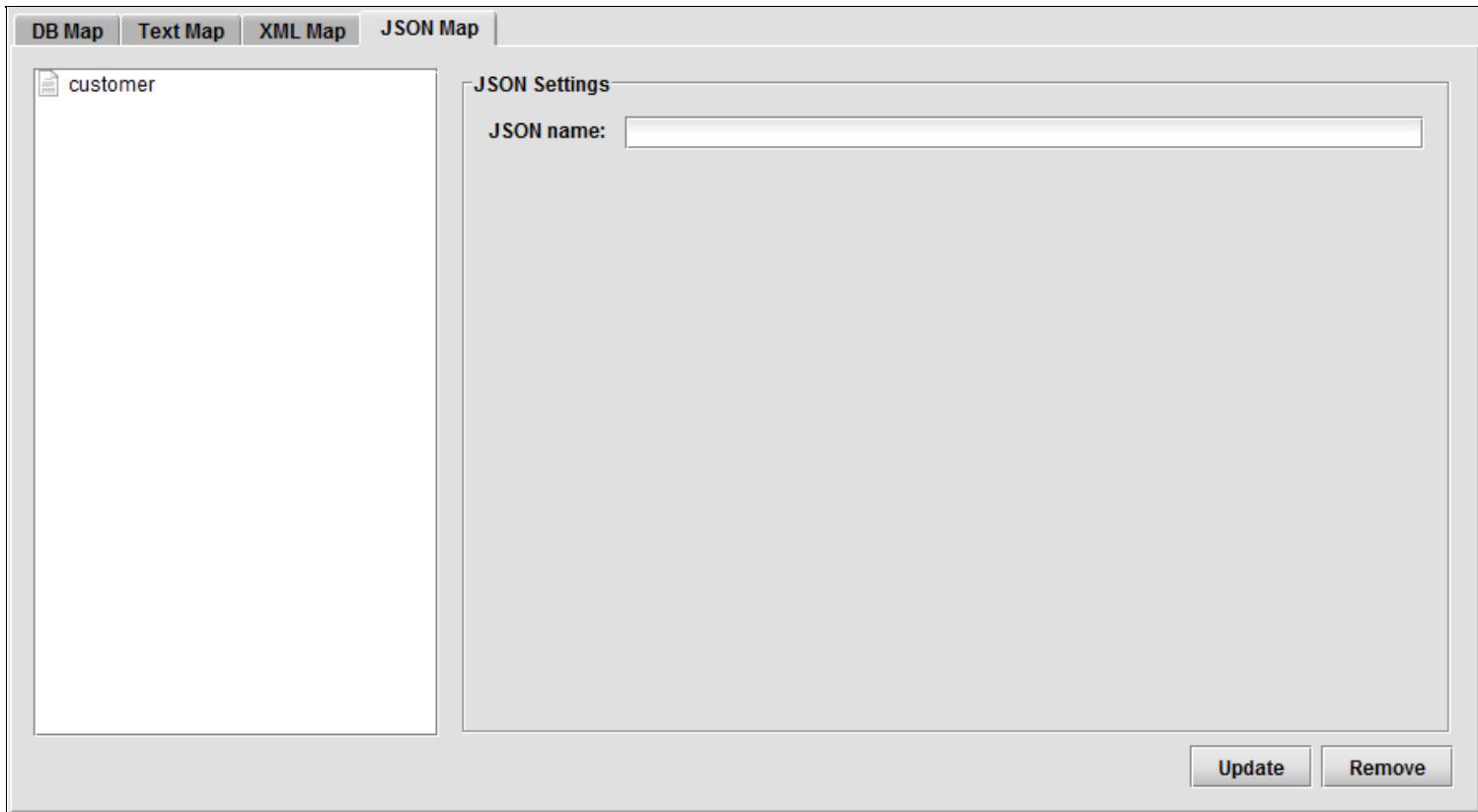
Field	Description
<b>XML Name</b>	XML tag name for this element. Default value is the element name.
<b>Length</b>	Visible for Variable elements only. Data length.
<b>Format</b>	Visible for Variable elements only. Data format.
<b>Exclude</b>	When checked for Variable elements, this Variable will be excluded when generating the XML. When checked for branch elements, when generating the XML this element's XML tag will not be generated, but the XML tags for the children elements will be.
<b>Exclude children</b>	Visible for branch elements only. When checked, this element, and all of its children, will be excluded when generating the XML.
<b>Include namespace</b>	Visible for a Document root only. When checked, the <code>xmlns</code> attribute will be generated with the namespace value of the namespace this Document belongs to.
<b>Attribute</b>	Visible for Variable elements only. When checked, the data will be generated as an attribute of the Document tag when generating the XML.
<b>CDATA</b>	When checked, CDATA will be generated when generating the XML.
<b>Update</b>	Must be clicked for the data entered on the screen to be stored in the metadata as a XML mapping for the selected Variable.

**Remove**

Cleans the screen and removes the XML mapping for the selected Variable.

## Mapping Tab: Document to JSON Mapping

The **JSON Map** sub-tab in the **Mapping** tab specifies how to export/import Documents to/from JSON format.



The following table describes the field in the tab.

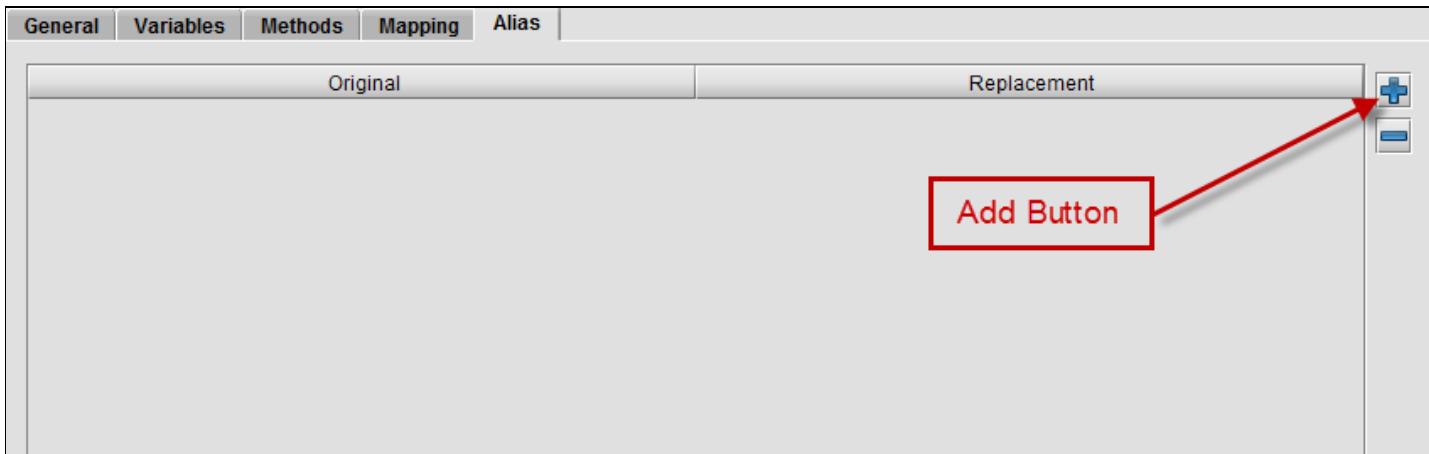
Field	Description
JSON Name	JSON tag name for this element. Default value is the element name.

## Document Alias

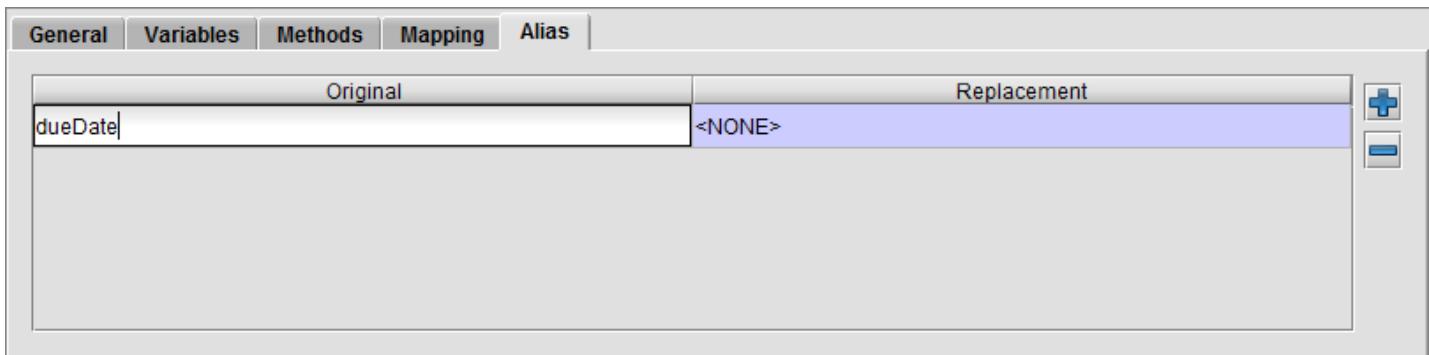
When variables under documents are renamed in template metadata, dependent metadata with references to the original variables results in unresolved reference errors. The Alias tab allows you to provide an alias for a renamed variable so that dependent metadata that references the original variable will reference the new variable instead.

To add an alias, complete these steps:

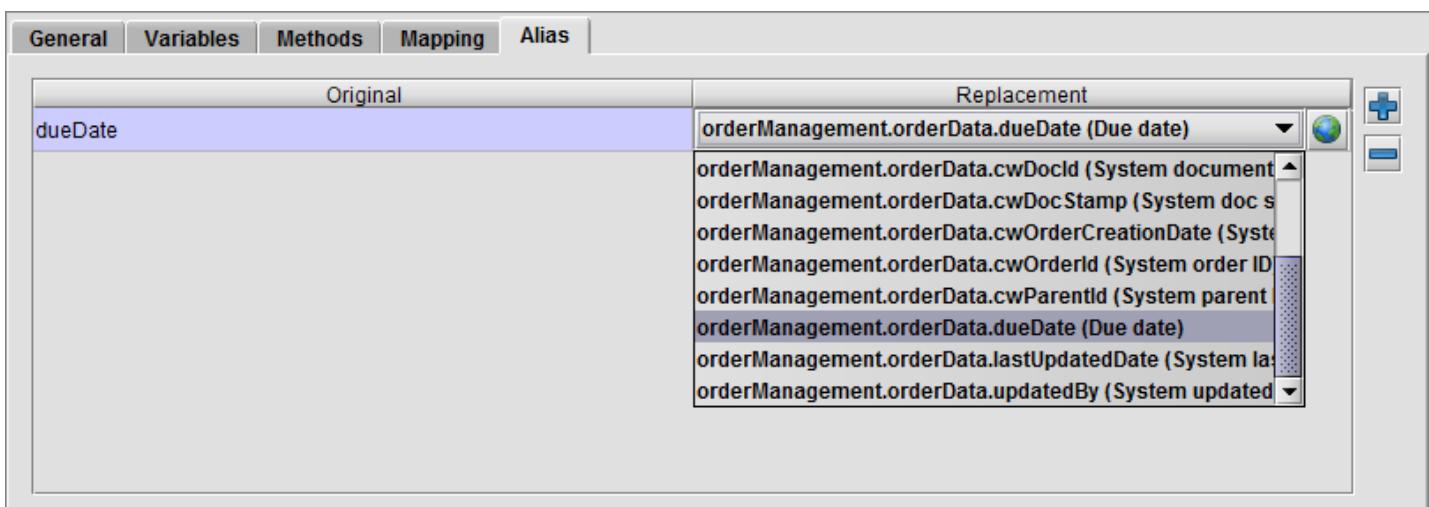
1. From the Alias tab, click the **Add** button.



2. Double-click the **Original** field and specify the original variable. Press the **Enter** key to continue.



3. Click the **Replacement** field of the new row that you have added and select the new variable that you want from the list. Alternatively, you can click the **Select Element** button (with a globe icon) and enter the object name that you want in the field provided.



4. When you save and reload your metadata, all original elements in your metadata are replaced with the replacement elements using this aliases map.

**Notes:**

- Both the **Original** and **Replacement** fields are mandatory. If the **Original** field does not contain a value, the row entry is automatically removed. If the **Replacement** field does not contain a value, a validation error occurs.
- Even though the alias information appears under the document's Alias tab, the actual information is stored under the metadata header. Therefore, whenever you make changes to a document's alias, the metadata header becomes dirty and not the document itself.

**Delete an Alias**

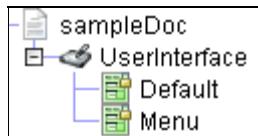
To delete an alias from the Alias tab, highlight the alias that you want and then click the **Delete** button.

General	Variables	Methods	Mapping	Alias	
				Original	Replacement
				fName	CustomerNS.customerInfo.fName
					 

A red box highlights the "Delete" button icon (a small red square with a white minus sign) located in the bottom right corner of the Alias table. A red arrow points from this highlighted box towards the center of the table, indicating the target for deletion.

## Document User Interface and Forms

Document Forms cohere to the User Interface generalization among metadata objects; that is, the Document contains a [User Interface](#), where [Forms](#) are contained within the User Interface. See [User Interface Modelling](#) on how you should use the User Interface object as *Controller* to connect the Document as *Model* to present Document Forms as *View* to users, adhering to the MVC architectural pattern.



A Document Form typically defines how the Document will be displayed at runtime in the browser, and is typically referenced in a [Form Frame](#) of another Form or Page to be displayed as part of a web-based application. You can define the layout of the Form, which Document Variable to display and/or edit, which Form Element to use to present each Document Variable, and also control over the Form Elements to tailor to your needs of the Document Form.

## Document User Interface

Identical to other User Interface metadata, Document User Interface has the [General](#), [Variables](#) and [Methods](#) tabs; please see the respective section for details.

Specific to Document User Interface, however, is existing User Interface Variables and Methods already defined from its base User Interface object `com.conceptwave.system.Document.UserInterface`.

### Pre-defined Variables of Document User Interface

Variable	Type	Description
<b>model</b>	<code>&lt;Type of this Document&gt;</code>	<p>The Document Variable that models the data of this Document. As per MVC architecture, this Variable is the means of the User Interface as (C)ontroller to expose the (M)oel to the (V)iew, thus allowing to present data of this Document metadata object type in the Forms.</p> <p>At runtime, this Variable may be initialized as a new Document object or as an existing Document, depending on how this User Interface is instantiated. If the <i>model</i> parameter of the Document User Interface constructor is not provided or null, a new Document object is created into this Variable. Otherwise, this Variable assumes the object passed in through the <i>model</i> parameter. In either case, you are not required to instantiate a new Document object for this Variable.</p>
<b>confirmObject</b>	<code>com.conceptwave.system.Object</code>	

You are strongly encouraged to read [Scripting](#) on how to use User Interface Variables.

### Pre-defined Methods of Document User Interface

Method	Return	Script
<b>onAutoSave</b>	<code>com.conceptwave.system.Void</code>	System method that is triggered on auto-save.
<b>onClick</b>	<code>com.conceptwave.system.Void</code>	The generic system method that is triggered when there is a mouse-click.
<b>onFullTextEdit</b>	<code>com.conceptwave.system.Void</code>	Parameters <i>leafName</i> , <i>isEditable</i> . A User Action Method that is invoked by default when the drill-down icon of <a href="#">Large Text</a> Form Element is clicked in any of the Forms. The default action is showing the Large Text pop-up Dialog Form.
<b>onInit</b>	<code>com.conceptwave.system.Void</code>	Document User Interface initialization script to set initial values for any Variables in the User Interface. It is triggered when the User Interface is created.

<b>onRefClick</b>	<i>com.conceptwave.system.Void</i>	Parameters <i>leafName, isEditable</i> . A User Action Method that is invoked by default when either the drill-down icon of <a href="#">Translation</a> or <a href="#">Reference</a> Form Element is clicked in any of the Forms. The default action is showing the Translation Finder or Reference Finder respectively as pop-up Dialog Form.
<b>onTimer</b>	<i>com.conceptwave.system.Void</i>	The Method that is invoked periodically for every <b>Time duration</b> defined in User Interface <b>General</b> tab since the creation of this User Interface, if it is defined. By default, this Method is empty.
<b>onValidation</b>	<i>com.conceptwave.system.Void</i>	The method <i>onValidate</i> is called on validate notification. The <i>validate</i> method of the metadata validates all fields and the model of the UserInterface (if exists) and returns ValidationErrors object.
<b>runTrigger</b>	<i>com.conceptwave.system.Void</i>	The Method that is invoked when the content of a Form Element is changed at runtime, which by default invokes all Trigger Methods of the Document.
<b>Save</b>	<i>com.conceptwave.system.Void</i>	The action of saving <i>model</i> data of the Document. By default, validation Methods are invoked before the save operation takes place. This is the Method that is invoked when the Save button in <b>Menu</b> Form is clicked.

When a Document extends from a base (non-system) Document, the Document has a User Interface that extends from the base Document's User Interface. In particular, the extended Document User Interface inherits all the Forms, Variables and Methods of the base User Interface, with the exception of the **model** Variable. This model Variable does not inherit, but rather created at the extended User Interface with Data Type of the extended Document. Thus, the extended User Interface "connects" to the extended Document rather than the base Document (which makes sense!).

## Document Forms

Multiple forms may be defined for one Document. This is useful for scenarios when you need to create different views of the Document for different users (for example, an administrative user versus a regular user) or for different occasions (for example, a "summary" view versus a "detailed" view). A Form can have permissions that specify in which situation it can be used. When different users open the Document, they may see different sets of fields in different formats, based on their individual roles and the content of the data.

There is tremendous flexibility that is provided to Form design. See [User Interface Modelling](#) to leverage full capabilities of UI design.

**Note:** You must create or generate a Form after defining a Document to present it in UI.

**Note:** If you change your Document's Variables after the Form is generated, you must update the created / generated Form corresponding to your Document Variables' changes.

The User Interface of a new Document contains two default Forms, which are inherited from base User Interface *com.conceptwave.system.Document.UserInterface*:

- **Default** Form: The default Form of the Document. By default, this Form is empty. Override this Form to present the Document with Form Elements and layout that suits your needs.
- **Menu** Form: contains object menu that pertains to using the Default Form, namely the Save menu button.

You can override this **Default** Form to create your own default Form for the Document, or use the [Generate Form command](#) to have Velocity Studio generate a Form based on the Document's current set of Variables.

Notice that the **Default** Form, itself, does not contain the **Menu** Form. The **Menu** Form appears below the **Default** Form (by default, but can be chosen) when the Document Form is to be displayed within an Application (see *com.conceptwave.system.Application* in [Top-Level User Interface Objects](#)) via [User Action Method](#). This User Action Method may be invoked by any existing Form Element in the Application; for example, it can be setup to be invoked by a Menu Item click.

[Document Form shown within an Application in runtime](#)

**Customer** ← Menu Item of Application

**customer**

First Name  lastName \*

active  True  False duedate

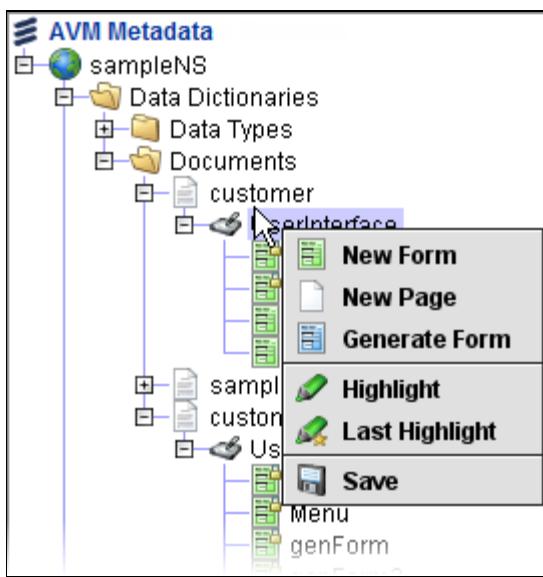
Save ← Menu Form of Document

A screenshot of a Document User Interface window titled "Customer". It contains a form with fields for "First Name" and "lastName \*". Below these are radio buttons for "active" (True or False) and a date input field for "duedate". At the bottom is a "Save" button. Red arrows point from the text labels "Menu Item of Application" and "Menu Form of Document" to the window title and the "Save" button respectively.

A Document User Interface can also contain a [Page](#) in addition to Forms. Similar to other User Interface objects, only one page is allowed in a Document User Interface, with fixed name *Page*.

## Context Menu in Document User Interface

To create a Form or a Page in a Document User Interface, right-click the User Interface node in the **Metadata tab of Navigation** pane. Select **New Form** or **New Page** respectively. Enter Form/Page name in the subsequent wizard pop-up, and click **Finish**.



The right-click pop-up menu also has a [Generate Form](#) command which creates a Form based on the Document's Variable list. For the rest of the pop-up commands, please see pop-up menu description in [Navigation Pane](#).

## Generating a Document Form

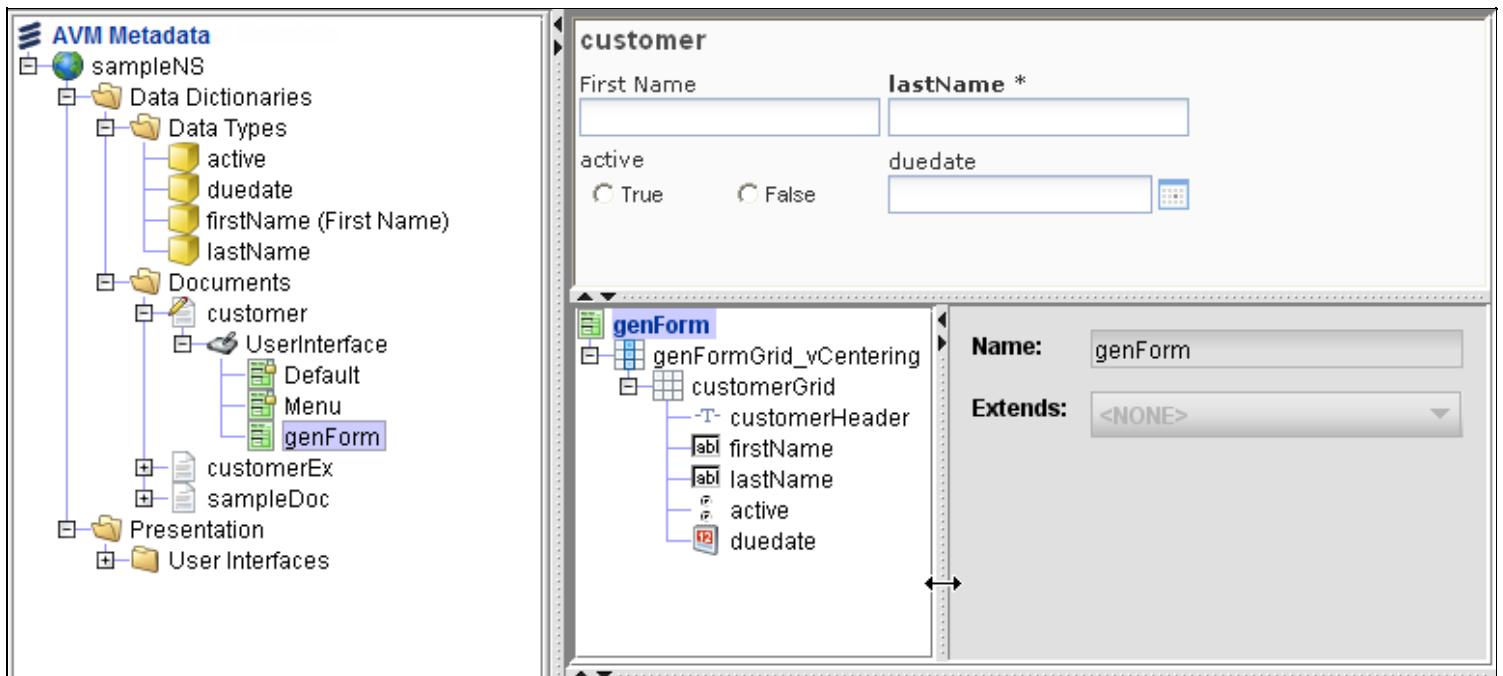
Based on the model of the Document, A default Form can be generated. This Form can be edited, which can be served as a library from which you refine the Form as desired.

This generated Form includes all Variables of the Document to be presented in a Grid Layout with a configurable number of columns. A Header Element showing the Document label is first shown in the Form. The Form Elements of the Form are generated in the same order as the order of Document Variables list.

To generate a Document Form

- In the **Metadata tab of Navigation** pane, right-click the User Interface of a Document. Select **Generate Form**.
- **New Generated Form** wizard appears as pop-up. Type in the name of the Form (must conform to JavaScript naming conventions). Specify the **Number of Columns** that the Document fields shall be spanned across. Then click **Finish**.

After the Form is created, the node is added under the User Interface node.



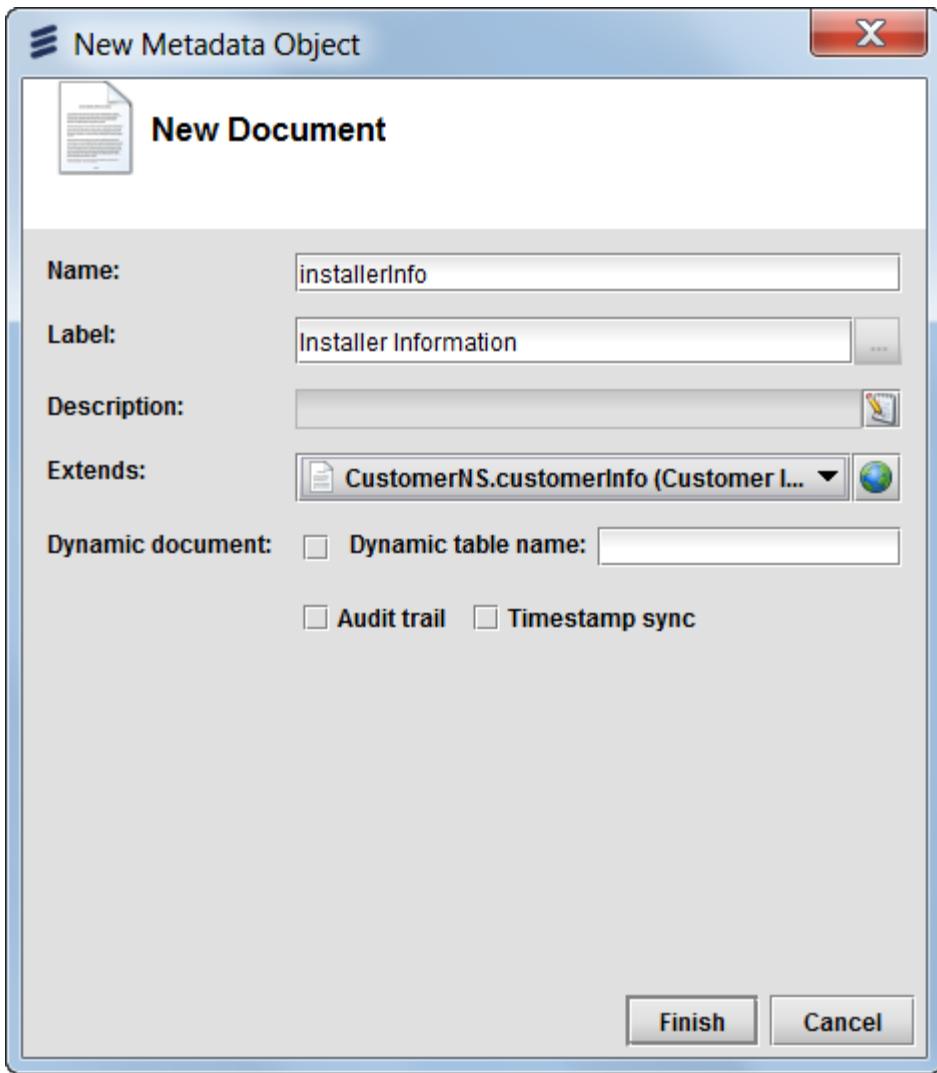
## Extending Documents

Existing Documents from system metadata or libraries can be extended in the user metadata to change them or add new functionality.

A Document that extends an existing Document (base Document) is called a derived Document. These rules apply to derived Documents:

- A Document can be derived from only one base Document. The base Document Variables are placed at the beginning of the derived Document's **Variables** list on the **Variables** tab and cannot be changed.
- The derived Document can add new Variables to the base Document, but cannot have another base Document inside it.
- The derived Document inherits the forms of the base Document. An inherited form can be overridden in the derived Document.
- The base Document Variable initialization scripts are always processed before the derived Document Variable initialization scripts defined in the **Methods** tab.
- For a given Document, the Variable initialization scripts are performed before the Document initialization script defined on the **Methods** tab of the Document.
- For all Document-level scripts, the base Document's scripts are run at the derived Document, unless they are overridden by derived Document's scripts (that is, with override command in **Methods** tab at the derived Document), where the derived Document's scripts are run, but not the base Document's.
- For all Document Variable-level scripts, the Variables from the base Document may have scripts that are run at the derived Document. These scripts can neither be overridden at the derived Document, nor can additional scripts be added to that Variable at the derived Document.
- Derived Documents cannot change or override the DB mapping of the base Documents.
- When extending or overriding a Document, you can add methods..

To extend an existing Document, follow the same instructions as [creating a new Document](#). However, in the **Extend** property in the Document creation wizard, choose the desired base Document to extend (instead of **com.conceptwave.system.Document**, which creates a new Document).

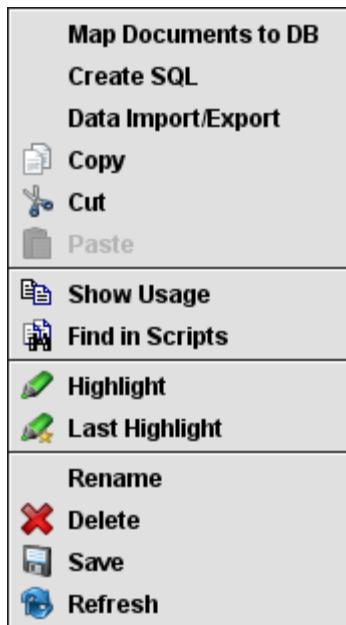


You can extend from a Document in your application Metadata (that is, Document found in **Metadata** tab in Navigation Pane) or extend from a Document in the library Metadata (that is, Document found in **Library** tab in Navigation Pane).

Once the derived Document is created, the Document cannot change its base Document (that is, **Extends** property in **General** tab of the derived Document is read-only).

## Document Operations

The right-click pop-up menu for an individual Document has several items in addition to the [common right-click menu items](#).



[Description of additional pop-up menu items for a Document.](#)

Field	Description
<b>Map Documents to DB</b>	Maps the Document to the database, which is similar to <a href="#">Map Documents to DB</a> in <b>Database</b> main menu of Velocity Studio, but only limited to the current Document. This item will be disabled if the <b>Skip DB schema generation</b> check box is selected on <b>DB Map</b> sub-tab of the <b>Mapping</b> tab.
<b>Create SQL</b>	Generates a SQL file containing the DDL (Data Definition Language) statements needed to create the tables that the Document is mapped to. This is similar to <a href="#">Upgrade System</a> in <b>Database</b> main menu of Velocity Studio, but only limited to the current Document. This item will be disabled if the <b>Skip DB schema generation</b> check box is selected on <b>DB Map</b> sub-tab of the <b>Mapping</b> tab.
<b>Data Import/Export</b>	Opens a dialog (Figure 4-76) which allows the user to import Document data from a text or XML file, modify it and export it back into a file or to create new data dynamically and export it into a file.

### Create SQL

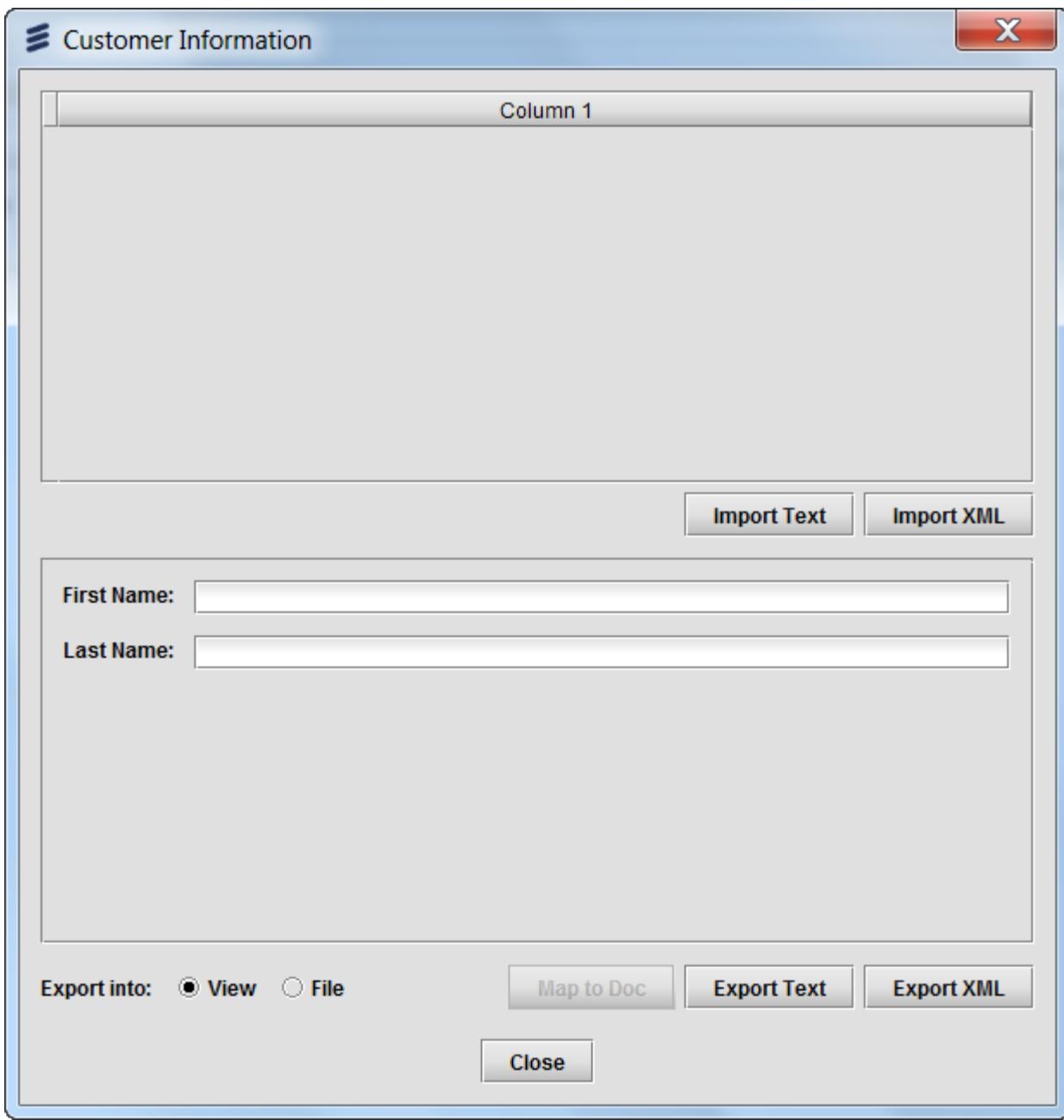
The **Create SQL** menu item is used once the Document Variables are mapped to the database table columns. This can be done by either selecting the **Map Documents to DB** menu item or on the **DB Map** sub-tab of the **Mapping** tab.

When the **Create SQL** menu item is selected, the **Generate SQL** dialog will open where the user can specify the file name and location. Once a directory is selected and a file name is entered, click the **Save** button. A SQL script will be created that contains the DDL (Data Definition Language) statements needed to create the database tables that the Document is mapped to. It is important to note that the SQL script contains DROP and CREATE statements. Therefore, it should not be executed against an existing table that contains data that must be kept. Connect to the appropriate database (use SQLPlus) and execute the SQL.

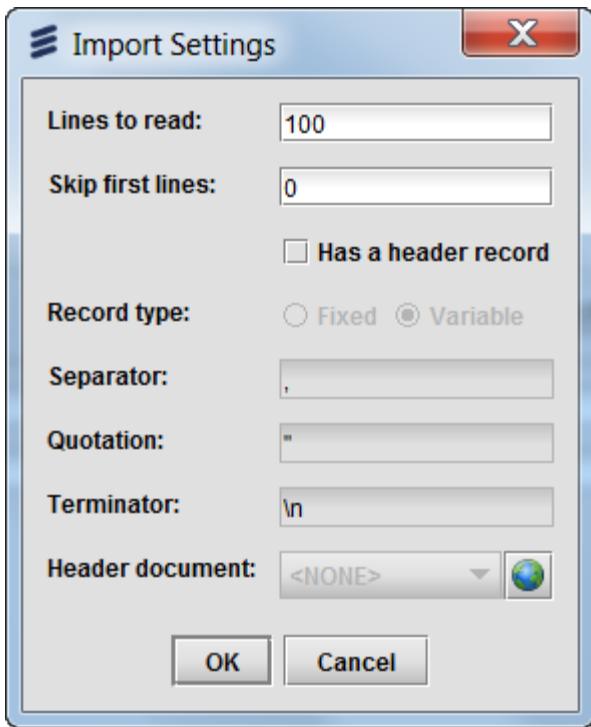
**Important note:** The generated DDL is intended as a starting point for DBA work, as it does not contain indexes, constraints, triggers, or any advanced DDL clauses.

### Data Import/Export

The **Data Import/Export** menu item is used to import data from a text or XML file, modify it and export it back into a file or to create new data dynamically and export it into a file. When selected, a data import/export dialog will open.



To import data from a text file, click the **Import Text** button. The **Import Document from Text** dialog will open where the file is selected. Click the **Open** button and the **Import Settings** dialog will open where the text file settings can be defined for the fields that are enabled. These settings will determine how the file is processed.

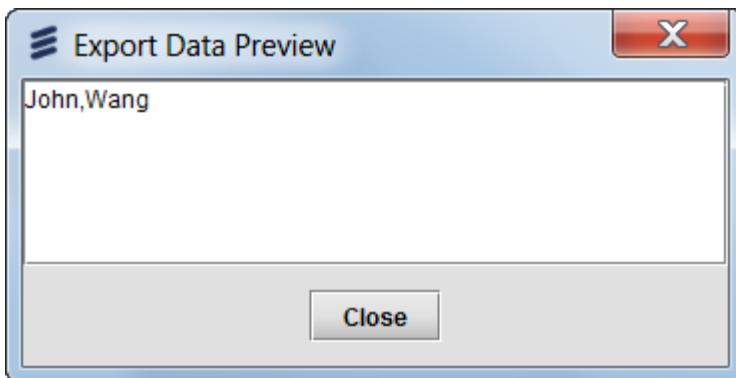


To import data from a XML file click the **Import XML** button. The **Import Document from XML** dialog will open where the file is selected.

Once imported, the data will be displayed in the top pane of the dialog and the values for the first record will be populated in the corresponding fields in the bottom pane.

The bottom pane of the dialog contains one field for each Document Variable defined in the **Variables list** of the **Variables** tab. The field name will correspond to the label specified in the **Label** column of the **Variables list**. Those Variables that have the check box selected in the **Mandatory** column of the **Variables list** will appear in red, denoting a required field. The data entered in the field will be validated against the Variable data type as specified in the **Type** column of the **Variables list**.

Data can either be entered manually in the fields in the bottom pane or a record can be selected from the top pane to edit. To preview the data and file structure before exporting, select the **View** radio button and click either the **Export Text** or **Export XML** button depending on the export format. The **Export Data Preview** dialog will open.



Once the user is satisfied with the file structure, select the **File** radio button and click either the **Export Text** or **Export XML** button to export the Document data into a text or XML file.

Note: Only one record can be exported at a time, each to a separate file.

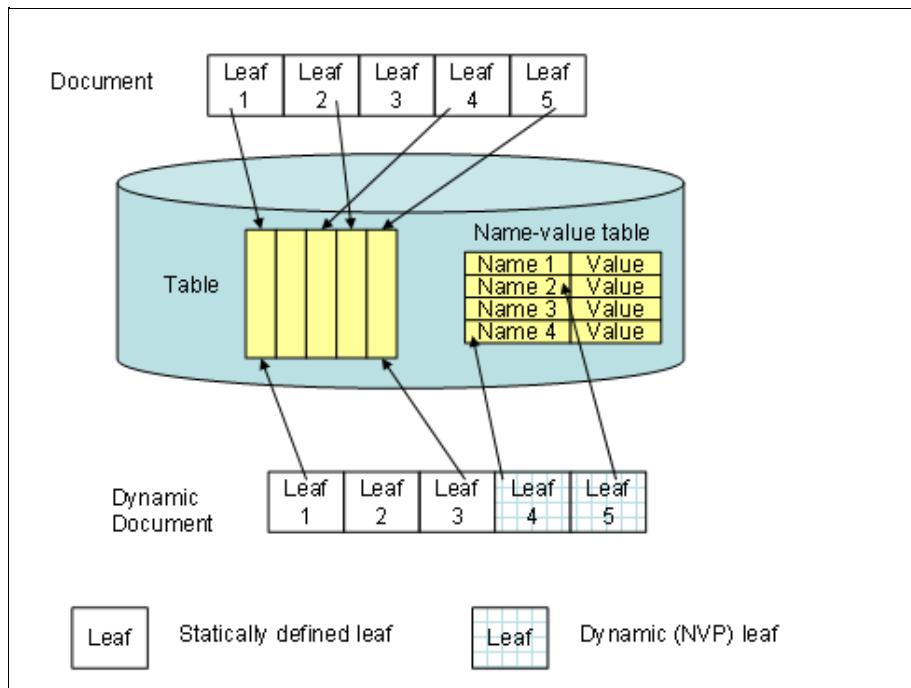
The **Map to Doc** button is currently not supported.

## Dynamic Document

Dynamic Documents are Document metadata objects that have Variables with name, data type, and database mapping that are not predefined in metadata, but programmatically defined at runtime. These Variables are called *dynamic Variables*.

Dynamic Documents are an extension of Document metadata, and thus have static Variables as well -- Variables that have name, data type, and mapping information defined in metadata. Therefore, Dynamic Documents may have both static Variables and dynamic Variables.

Dynamic Variables are mapped to rows in *name-value pair* (NVP) table, which is in a predefined format, to store their values at runtime. NVP table is also referred to as *dynamic table*.



The previous diagram shows the differences between (static) Documents and dynamic Documents with respect to their Variables (*Leaf*):

- Documents have only static Variables defined in the metadata and mapped to columns of database tables.
- Dynamic Documents have static Variables as well as dynamic Variables. At runtime, the dynamic Variables are mapped to rows in the NVP table.

The dynamic Document can predefine some of the dynamic Variables, but there can be those that will be known only at runtime.

**Note:** Velocity Studio features require a separate license key that is activated in Velocity Studio.

### Create New Dynamic Document

To create a new dynamic Document, first follow [instructions](#) on creating a new Document. Then, check the **Dynamic document** checkbox in **General** tab of the Document, and specify the NVP table name in **Dynamic table name**.

To be explained below, a dynamic Document must have a Document Key to create the NVP table. Thus, specify the keys of the dynamic Document in the **Keys** field.

General   Variables   Methods   Mapping   Alias

Name:	thing	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated
Label:	thing	...
Description:	<input type="button" value="Edit"/>	
Extends:	com.conceptwave.system.Document	<input style="vertical-align: middle;" type="button" value="..."/>
Overrides:	<NONE>	
<b>Dynamic document:</b> <input checked="" type="checkbox"/> <b>Dynamic table name:</b> thingNvPtable		
<input type="checkbox"/> Audit trail <input type="checkbox"/> Attachments <input checked="" type="checkbox"/> Generated key <input type="checkbox"/> Strict valid <input type="checkbox"/> Timestamp sync		
Keys:	cwDocId	

### NVP Table

The NVP table has the following predefined columns. The first four columns are the unique keys of the table.

Column Name	Type	Description
DOC_ID	VARCHAR2(36)	Contains the GUID or ID of the document.
DOCUMENT_TYPE	VARCHAR2(38)	GUID of the document type.
LEAF_NAME	VARCHAR2(64)	The dynamic Document variable name.
ARRAY_INDEX	NUMBER(4)	The array size. 0 for single values. 1 to 9999 for arrays.
DATA_TYPE	NUMBER(2)	Data type of the dynamic Document variable. <ul style="list-style-type: none"> <li>• 0 - Decimal</li> <li>• 1 - Integer</li> <li>• 3 - String</li> <li>• 4 - Boolean</li> <li>• 5 - Date_Time</li> <li>• 6 - Date</li> <li>• 9 - Code Table</li> </ul>
VALUE	NVARCHAR2(256)	The value stored as string. Details are defined in below section.

Notice that the NVP table contains *Document ID* and *Document Type* as part of the composite key of the table. Thus at runtime, each dynamic Document object instance may have its own NVP table, separate from other dynamic Document object instances. Indeed, an application may have many NVP tables, but in practice, their number must be limited.

## Data Type Support to Dynamic Variables (and limitations)

Dynamic Variables are limited to the following base types:

- Boolean
- Number (Integer or Decimal)
- String
- Enumeration (Code table)
- Date

Notably, [Reference](#) and [Translation](#) Data Types are not allowed in Dynamic Variables.

There is considerable difference in how values are stored in database for dynamic Variables versus static Variables. For dynamic Variables, they are stored always as characters in the **VALUE** column as *NVARCHAR2(256)* with the following restrictions and conventions:

1. Boolean values - stored as string 0 to denote false and 1 to denote true
2. Number values - stored as decimal string (with or without decimal point)
3. String values - limited to 256 characters
4. Date values - stored as we keep them in memory (as in XSD standard)

As a comparison, the static Variables are stored in database columns of the corresponding type as specified in column *Oracle* in [Type Mapping](#).

Because of this implementation in NVP table, numeric manipulation in database with dynamic Variables of Number type is extremely awkward and is discouraged. Thus, *only dynamic Variables of String and Boolean types should be used in creating dynamic Finders*.

## Scripting and Runtime

At runtime, the dynamic Document is not different from the (standard) Document. The difference is the ability to create the dynamic part on the fly.

In terms of scripting, there exists the [Dynamic Document](#) object class which is extended from *Document* object class. This is consistent with the notion that Dynamic Documents are an extension of Document metadata. Dynamic Documents are created by the constructor of this Dynamic Document object class.

Creating dynamic Variables can be done in the [constructor of dynamic Document object](#), or by the method [createLeaf](#). The dynamic Variable names must be unique in the dynamic document (including the static Variable names).

**Note:** Catalog does not support the basketitem.createLeaf operation. As an example, instead of creating a leaf in Catalog, you would create an item attribute relation for the item to the corresponding attribute.

Access to dynamic Variables does not differ from the access to static Variables. The same dot notation is used to get and set values. For example, if dynamic Document *myDoc* has defined dynamic Variable *myVar* of type string, then:

```
myDoc.myVar
```

returns the value of of the dynamic Variable, and

```
myDoc.myVar = "string example";
```

sets the value of the dynamic Variable to the value "string example".

It is recommended that the Variables must be defined before the first operation which sets or gets Variable value, regardless whether the Variable is static or dynamic. The dynamic Variable behaves just as the static Variable. However, you may continue to add new dynamic Variables at any moment of the Document life.

Note that there is no way to remove dynamic Variables from a dynamic document.

## Array Types in Dynamic Variables

Unlike static Variable where it must be single value per Variable, dynamic Variables allow arrays of values per Variable. The type of the array can be one of the types allowed for the dynamic Variables, and the database format of the values remains the same as explained above. The array is ordered, with size up to 9999, but should not be more than, say, 20 elements in practice. For the sake of database operations, the dynamic Variable array values are stored in the same NVP table as the single dynamic Variable values.

In scripting, the getter of dynamic Variable of Array type returns an array, even if it consists of one element only.

The setter, however, accepts array of objects or a single object as parameter. In the latter case, it is converted to a string array of one element. Assigning value to an array Variable will replace the current array value of the Variable with the specified array. It will not add the values to the

existing array.

## Create a Dynamic Document with Dynamic and Static Layouts

There are two methods of creating a Dynamic Document:

- [Create a Dynamic Document with a completely dynamic layout](#)
- [Create a Dynamic Document with a predefined, static layout](#)

## Dynamic Document Definition and Reference Table

A **dynamic document definition** is used as the metadata object for a dynamic document. It contains a list of dynamic variables. Dynamic document definitions were originally both instance and metadata objects. Each dynamic Document can contain a unique set of dynamic variables.

A dynamic document definition is an effort to provide a common definition for dynamic documents where you specify the set of dynamic variables in the dynamic document definition. Instances of dynamic documents that were created based on the dynamic document definition then share the same set of dynamic variables.

The scriptable type that deals with the dynamic document definition is the `DynamicDocumentDefinition` type. The constructor takes two parameters:

- The first parameter is the static metadata document's full name.
- The second parameter is the dynamic document definition's unique name. The system manages every dynamic document definition created and this definition must have a unique name. Once created, you can retrieve the definition by name using the `DynamicDocumentDefinition.findByName` static script method .

You can use the `save()` API to save your dynamic document definition. The default system implementation writes the dynamic document definition to the `CWDYNAMICDOCUMENTDEFINTION` system-managed table. To convert the dynamic document definition to and from XML, use the `toXML()` and `getItemsFromXML()` methods, respectively.

A **reference table definition** includes a data object list that contains a list of dynamic documents. A reference table is a way to manage dynamic documents created by the same dynamic document definition. Currently, there is a one-to-one relationship between a reference table and dynamic document definitions.

A reference table constructor takes a single parameter, which is either the dynamic document definition instance, or the dynamic document definition's name. In either case, the dynamic document definition's name also becomes the reference table's name. The reference table's name becomes a unique key with which you can find existing in the system.

Reference tables are saved in the database under the `CWREFERENCETABLE` system-managed table. While the system currently does not explicitly prevent creating multiple instances of reference tables based on the same dynamic document definition, attempts to save these instances conflict in the database as they end up being saved under the same row and overwrite one other. Future implementations will remove this one-to-one restriction and multiple reference tables will be defined from a single dynamic document definition. In such a case, the reference table's name will still be unique, but will no longer come from the dynamic document definition.

Dynamic documents are not created based on a dynamic document definition. Rather, they are created using the reference table that is based on a dynamic document definition. You would usually create an instance of the dynamic document definition, create an instance of the reference table, and then use the reference table to create and remove instances of dynamic documents. You can use the `ReferenceTable.createNewRecord()` API to create new dynamic documents. This API returns an instance of the dynamic document based on the dynamic document definition that the reference table is based on. The API offers the option to automatically add the created instance to its managed list of dynamic document instances.

You can retrieve the dynamic document instances managed by the `ReferenceTable` by using the `findById()` and `findByValue()` methods. You can also convert the reference table's entire contents to XML using the `toXML()` method. To create a list of reference tables from XML, use the `getItemsFromXML()` static method.

## Attachments in Documents and Orders

In product version 5.x, the following metadata objects were used to add, load, and display attachments to document and order items:

- cwf\_oe.AttachmentFinder
- cwf\_oe.CWDOCATTACHMENT

The following table describes the metadata object pertaining to attachments in the product:

Metadata Object	Description
cwf.attachmentDoc	This object replaces the cwf_oe.CWDOCATTACHMENT document. It has the same structure (variable names, variable order, and mapping to database) as the old document to support backward compatibility. The document is located in system metadata and has no child UI.
ui_common.attachmentFinder	This object denotes the new attachment finder implementation. It has a child UI based on the ui_common.baseFinder user interface.
ui_common.attachmentSearch	This object is a search document that replaces cwf_oe.AttachmentSearch. The search document has no default child UI.
ui_common.attachmentUI	This object represents the top-level UI for the attachment document detail view.
ui_common.attachmentResult	This object denotes the top-level UI for the attachment finder's result table.

## Events

The following table describes events pertaining to attachments. Parameters that are bolded signify that they are mandatory:

Event	Parameters	Description
SYSTEM_ATTACHMENT_ADD	<ul style="list-style-type: none"> <li>• owner (order item)</li> <li>• description</li> <li>• <b>fileName</b></li> <li>• mimeType</li> <li>• note</li> <li>• <b>attachmentData</b></li> <li>• attachmentDoc</li> <li>• <b>attachmentId</b></li> </ul>	The upload servlet calls this event to store the uploaded file into the attachment document. There are three scenarios to consider: <ul style="list-style-type: none"> <li>• Only the attachmentId is provided</li> <li>• The attachmentDoc is provided</li> <li>• The attachmentDoc is NOT provided</li> </ul>
SYSTEM_ATTACHMENT_FILE_GET	<b>attachmentId</b>	This event loads the attachment file by the given attachmentId and returns array [attachmentFile, fileName, mimeType].
SYSTEM_ATTACHMENT_QUERY	<ul style="list-style-type: none"> <li>• <b>ownerId</b></li> <li>• topOrderId</li> </ul>	This event returns a list of cwf.attachmentDocs by the given parameters.
SYSTEM_ATTACHMENT_INFO_QUERY	<b>ownerId</b>	This event returns attachInfoArr. The return information is an array in the following format: [attachmentId_1, attachmentName_1, attachmentMimeType_1, ..., attachmentId_N, attachmentName_N, attachmentMimeType_N ]
SYSTEM_ATTACHMENT_REMOVE	<ul style="list-style-type: none"> <li>• <b>docId</b></li> <li>• topOrderId</li> </ul>	This event deletes all attachments for the given document or order. This event is called when deleting the object from the database.
UI_GET_ATTACHMENT_FINDER	<ul style="list-style-type: none"> <li>• <b>ownerItem</b> (order item)</li> <li>• parentUI (for example, order UI)</li> </ul>	This event creates, initializes, and returns the finder attachment UI for the given ownerItem.

## Get and Display the Attachment Finder UI

To get an instance of the attachment finder, the UI\_GET\_ATTACHMENT\_FINDER event should be fired with the following parameters:

```
[ownerObject, parentUI]
```

The following example retrieves an instance of the finder for the document, which can be an action in the document UI:

```
var finderUI = publishEvent("UI_GET_ATTACHMENT_FINDER", FIRST_ONE, null, [this.model,this]);
return finderUI;
```

To see an example of this action, see com.conceptwave.system.OrderUserInterface > attachAction from Velocity Studio's Library tab.

## Customize the Default Behaviour

The UI Common template provides the default handler for the event to return the ui\_common.attachmentFinder implementation.

In your metadata, you can provide your own handler to return the old attachment implementation (cwf\_oe.AttachmentFinder) or any own extended finder.

## Document Notes

Document notes are stored in a separate CWDOCNOTE table. To have notes in a document, you must add two variables added to the document metadata:

- A variable of type cwf\_oe.sysNoteField to enter new notes  
This variable is used to store the value into the CWDOCNOTE table. Once the value is saved (that is, a new row is inserted into CWDOCNOTE), the variable is cleared.
- A variable of type cwf\_oe.sysNoteList to view early entered notes, which is read-only  
On accessing the variable, it used to load all notes from the table and combined them into one string separated with '\n'.

Because storing occurs using the cwf\_oe.sysNoteField data type and loading takes place using the cwf\_oe.sysNoteList data types, metadata that uses document notes should still work without any changes.

Two system events support these data types:

- The cwLoadValue method is called when you first access a document variable of the data type. The method loads and returns the value.
- The cwStoreValue method is called when you store a document to the database, to store the variable value.

### cwLoadValue and cwStoreValue methods

These methods add more flexibility to the loading and storing processes of document variables. A data type can have only one of those methods or both at the same time.

The cwLoadValue script loads the initial value for the variable:

```
function cwLoadValue(parent, leaf) {
    // The script is called when accessing the variable for the first time
    // The 'parent' parameter is the parent object
    // The 'leaf' parameter is the name of the variable
    // The script returns the loaded value
    return value;
}
```

The cwStoreValue script is called on saving the document into the database to store the variable value:

```
function cwStoreValue(parent, leaf, value) {
    // The script is called on storing the object
    // The 'parent' parameter is the target object
    // The 'leaf' parameter is the name of the variable
    // The 'value' parameter is the data to store
    ...
}
```

The cwLoadValue method is called when accessing the document variable of that data type for the first time. Document notes are loaded only when accessing a variable (for example, displaying it in a user interface or retrieving a variable (leaf) value in scripts). The value is stored in memory into the document data array. Next time, when accessing the variable, it retrieves the already loaded value.

The cwStoreValue scripts is called on saving the document, right after all database mapped values are restored to the database.

#### Notes:

- The store script for the variable is called only if it has a *changed* state.
- If none of the document variables are changed (that is, neither the database mapped variables, nor script variables), calling save() skips storing, does not update the document in the database, and does not call any scripts.
- If a variable with the store script was changed, either through the user interface or scripts, it update the document in the database and calls the script even if none of database mapped variables have changed.

After the document is saved, it also clears all loaded states for variables with the cwLoadValue scripts defined, so that they are reloaded next time when accessing the value. This clearing is performed due to possible value cross-changes in the cwStoreValue script.

## Data Structures

---

Data structures are hierarchical structures consisting of Data Types, Element Types, Documents, Containers, and Arrays. Leaf nodes are always Documents, Data Types or Element Types. In the **Navigation** pane, Data Structures are found in the **Data Dictionaries** folder in each namespace.

To create a new data structure, complete these steps:

In the **Metadata** tab of **Navigation** pane, either:

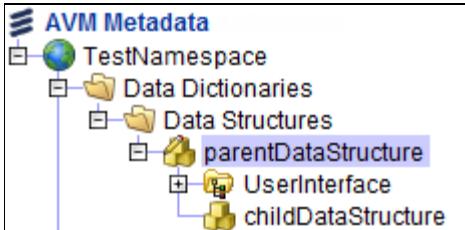
1. Right-click a Namespace. Select **New ...**
2. **New Metadata Object** wizard appears as pop-up. Expand **Data Dictionaries**, and select **Data Structure**, and then click the **Next** button.
3. Step two of wizard appears. Enter the **Name** of the data structure (must conform to JavaScript naming conventions), select the **Create UI** option if you want to create a user interface, and then click the **Finish** button.
4. To add a child data structure node, right-click the root data structure folder that you created and select **New Data Structure** from the menu.
5. Enter the **Name** of the child data structure, and then click the Finish button.

OR

1. Right-click the **Data Structures** folder or the **Data Dictionaries** folder in a Namespace, if it is present. Select **New Data Structure**.
2. Follow steps 3 through 5 in the previous section.

**Note:** You can adjust the order of child nodes by using the **Elements** tab of the root or parent node.

After the Data Structure is created, the icon will be added under the **Data Structures** folder.



**Note:** To add or remove a user interface, right-click your data structure and select **Create UI** or **Delete UI**, respectively.

Each node in the Data Structure hierarchy has properties that are displayed when the node is selected in the **Navigation** pane. New child nodes may be created by right-clicking the parent container node and selecting the **New Data Structure** command. Once a child node is created, its type is defined by checking one of the **Array**, **Container**, or **Data Type** check boxes on the **General** tab.

Properties for each of these types of nodes are explained in their respective sections.

## Data Structures - General Tab

The general Data Structure properties are defined on the **General** tab. The following image shows the **General** tab of the Data Structure.

General   Elements   Methods   Alias

Name: address  Private  Restricted  Deprecated  Virtual  Final

Label: address

Description:

Extends:  com.conceptwave.system.Data Structure  

Overrides: <NONE>  

Abstract  Generated key

XML settings:

XML name:

Include  Exclude

Include namespace  Include nulls and empty  Include nulls if set

Include xsi:type  Element  Group

JSON name:

Keys:

The data structure data type variables get initialized with the default value that is defined in the data type.

The following table describes the fields for the **General** tab of the Data Structure:

Field	Description
Name	Name of the metadata object (used in scripts; must conform to JavaScript naming conventions). Read-only after creation. To change it, use the <b>Rename</b> command in pop-up menu by right-clicking the metadata object.
Private	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
Restricted	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
Deprecated	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
Virtual	If checked, it provides the ability to create methods in template metadata that does not have implementation, but can be used in template script and be bound to UI elements. The implementation is done in user metadata (see <a href="#">Virtual Field</a> for more information).
Final	Metadata elements can be marked as final, which includes any Boolean final property that is available on any metadata

	element. Additionally, any metadata element that has the <b>Final</b> property set to true can neither be extended, nor overridden.
<b>Label</b>	Metadata object display label.
<b>Description</b>	Description of the metadata object for documentation.
<b>Extends</b>	Base Data Type to derive from. Unchangeable after Data Type creation.
<b>Overrides</b>	Metadata object to override. Used to customize system or library metadata objects.
<b>Abstract</b>	An Abstract data structure may not have child nodes and will usually be extended by other data structures. Some data structures can refer to the Abstract structure, however at runtime the real data may correspond to one of its derived types.
<b>Generated key</b>	<p>If checked, this property indicates that the system automatically generates the key value for this data structure. When checked, the Confirm Generated Key dialog appears.</p> <p>If you click the <b>Yes</b> button, Velocity Studio tries to locate, within existing variables, a variable with the <i>cwf.cwDocId</i> data type to use as a key. If such a variable is not found, the key is automatically created and added to the Variable list of the data structure. This variable is added to the key list of the data structure and all other keys are removed.</p>
<b>XML Settings</b>	XML settings control the conversion of the Data Structure to and from XML. The properties vary depending on the node type (see <a href="#">XML Settings</a> for more details).
<b>JSON name</b>	The JSON tag name for this data structure. The <b>JSON name</b> field is used to specify the JSON tag name for this node. This field defaults to <b>Name</b> .
<b>Keys</b>	<p>If the <b>Generated key</b> checkbox is unchecked, this field defines the list of data structure variables used to uniquely identify a data structure instance. To define keys, click the <b>Add</b> button and select variables from the list in the Add Element dialog that opens. To remove an existing key, select the key and click the <b>Remove</b> button. When the <b>Generated key</b> checkbox is selected, this field contains the system-generated key and is disabled so that it can neither be changed, nor removed. A key must be defined for the following types of data structures:</p> <ul style="list-style-type: none"> <li>• A data structure that is mapped to a database.</li> <li>• A <a href="#">database map</a> having a key column used during DDL generation where all SQL statements use a primary key.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• You can add any simple type variable in this keys list and you can reorder it. However, changing the key order may affect the key-generating logic. Existing records that have been previously saved may not be accessible when you change the key order. In this case, a warning appears.</li> <li>• When you define a key variable in the data structure, the database map uses this data structure key variable and shows it as key columns on the Mapping tab's source tree.</li> <li>• When a key variable is not mapped to a database column, an error message appears. If the data structure defines multiple key columns, all key variables must be mapped to database columns.</li> </ul>

## Root and Container Nodes

Root and container nodes are nodes that contain a set of child nodes that can be array nodes, container nodes or data type nodes. Root and container nodes have an **Elements** tab (refer to section on Elements Tab) in addition to the **General** tab.

A Data Structure is initially created with one root node that is the root container of the Data Structure. The root node is a container node with slightly different properties. The following figure shows the **General** tab of a root node.

The screenshot shows the 'General' tab of a root node configuration window. The 'Elements' tab is also visible at the top. The 'Name' field contains 'testdatastr'. The 'Label' field contains 'testDataStructureContainer'. The 'Description' field is empty. The 'Extends' field shows 'com.conceptwave.system.DataStructure' with a dropdown arrow and a globe icon. Below it, a checkbox for 'Container' is checked, while 'Array' and 'Mandatory' are unchecked. The 'XML settings' section includes an 'XML name:' field and four radio buttons: 'Include' (selected), 'Exclude', 'Exclude children', and 'Include Always'. The 'JSON name:' section includes a 'JSON name:' field. The entire window has a light gray background and a white content area.

The **Extends** field is included on the **General** tab of the root and container nodes. It allows the user to extend another top level Data Structure. The children from the extended Data Structure will appear as children of the current Data Structure in the data mapping UI. Children do not have the **Extends** field.

An **Abstract** data structure may not have child nodes and will usually be extended by other data structures. Some data structures can refer to the Abstract structure, however at runtime the real data may correspond to one of its derived types.

All nodes have **XML settings** (refer to section on XML Settings). The root node also has an **Initialization** script, which is currently not implemented.

**Note:** Top-level nodes that do not have a namespace are unsupported. Instead, you will need to manually create the xmlns attribute in your metadata and populate the value in your script.

All nodes have **JSON settings**. The **JSON name** field is used to specify the JSON tag name for this node. This field defaults to **Name**.

The **General** tab of a container node is shown in the figure below. The **Container** checkbox must be selected in a new Data Structure node to make it a container. The checkbox will not appear in the properties of an already created container node. The **Mandatory** checkbox must be checked if the data objects contained in this node must be always initialized.

**Note:** The **Array** checkbox is described in section on Array Node.

General Elements

Name: customerOrder

Label: customerOrder

Description:

Container  Data type  Array  Mandatory

Instance: Minimum 0 Maximum

Element: <Any>

XML settings:

XML name:

Include  Exclude  Exclude children  Include Always

JSON name:

JSON name:

## XML Settings

XML settings control the conversion of the Data Structure to and from XML. The properties vary depending on the node type.

Name	Description	Usage
<b>XML name</b>	XML tag name for this node. Defaults to <b>Name</b> .	All.
<b>Length</b>	Data length.	Data type and array node.
<b>Format</b>	Data format.	Date type and array node.
<b>Exclude</b>	Data type node: excludes this node. Container node: excludes this node but includes the child nodes.	All.
<b>Exclude children</b>	Excludes this node and all of its children.	Container and array node.
<b>Include always</b>	Includes this node regardless of parent node settings. If this field is checked, the node (its tag) will be included in the XML even if its content is empty. (The default behavior is that the node (its tag) will not be included in the XML if its content is empty).	Container, data type and array node.
<b>Include namespace</b>	Generates xmlns attribute with namespace value.	Root node.
<b>Include nulls</b>	Generates tags for data elements that have no value (null). By default, only data elements with data will appear in the generated XML	Root node.
<b>Group</b>	When checked, the Data Structure will be treated as a XSD model group definition (refer to XML Schema Part 1: Structures Second Edition of W3C Recommendation) on XML import/export if used as reference in the Data Structure node. In this case, when the runtime system generates XML or parses XML data, the node itself will be replaced with child elements of the group Data Structure.	Root node.
<b>Attribute</b>	If checked, the node will be added as an attribute in the parent.	Data type node.
<b>CDATA</b>	Generates CDATA.	Data type node.
<b>Text</b>	Uses this data type node as a TEXT node of the parent (that is, the value of the node is added directly under the parent node tag). If multiple TEXT nodes are defined under the same parent, the values will be appended with each other.	Data type node.
<b>Include xsi:type</b>	Includes the XML xsi:type for this node.	Root node, Data type node.
<b>Element</b>	Check to specify if node is an element (that is, when doing WSDL import/export).	Root node, Data type node.

Elements of an XML-outgoing Data Structure with names that are invalid in XML must define the XML Name field. For example, an invalid XML name may have hats such as <^data^>, or start with a digit such as <1foobar>.

Conversely, these same elements in an XML-incoming Data Structure does not require XML Names. Instead, the adaptor transforms these names in the messages; for example, the hats character are parsed out (<^data^> become <data>) and underscore is prepended for the digit-starting tags (<1foobar> become <\_1foobar>).



## Data Structures - Elements Tab

Once one or more child nodes have been created under a root or container node, an **Elements** tab appears. This allows the ordering of the children to be changed. The user can randomly change the order using the arrow buttons located at the side of the list or the user can click the **Sort** button to order the list alphabetically. For Data Types, the data type name and length (if applicable) are shown in brackets.

The screenshot shows a software interface for managing data structures. At the top, there are two tabs: "General" and "Elements". The "Elements" tab is selected, indicated by a thicker border around its tab area. Below the tabs, the title "Children list:" is displayed. To the right of the list, there are two small square buttons with arrows: an upward-pointing arrow at the top and a downward-pointing arrow below it. A large rectangular list box contains the following items:

- CORSFX (string, 2)
- CUSTORD (string, 32)
- CUSTORDPASS (string, 3)
- CUSTORDTYPE (string, 8)
- DD (date)
- DDD\_FLAG (string, 1)
- DEFITEMACTN
- ENTITY (string, 8)
- Header Extension
- IMGFMT (string, 6)
- ITEM
- MSG\_SRC (string, 8)
- MSGDATE (date)
- MSGTIME (time)
- ORDACTN (string, 10)
- ORDIMG (string, 32000)
- ORDNUM (string, 32)
- ORDPASS (string, 3)
- ORDTYPE (string, 2)
- RESEND (string, 1)
- SDD (date)

At the bottom center of the list area is a small "Sort" button.

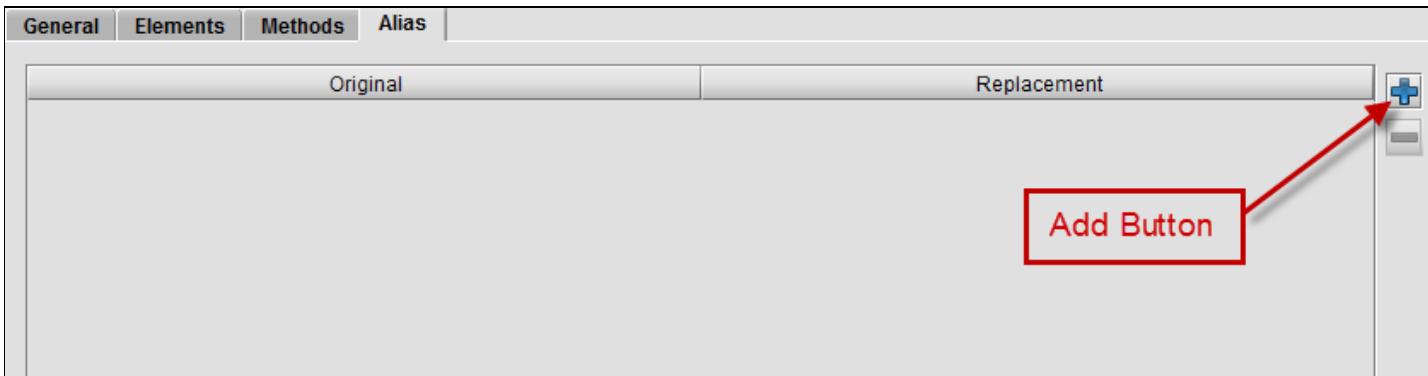
**Note:** Refer to [data type node](#) for more information.

## Data Structures - Alias Tab

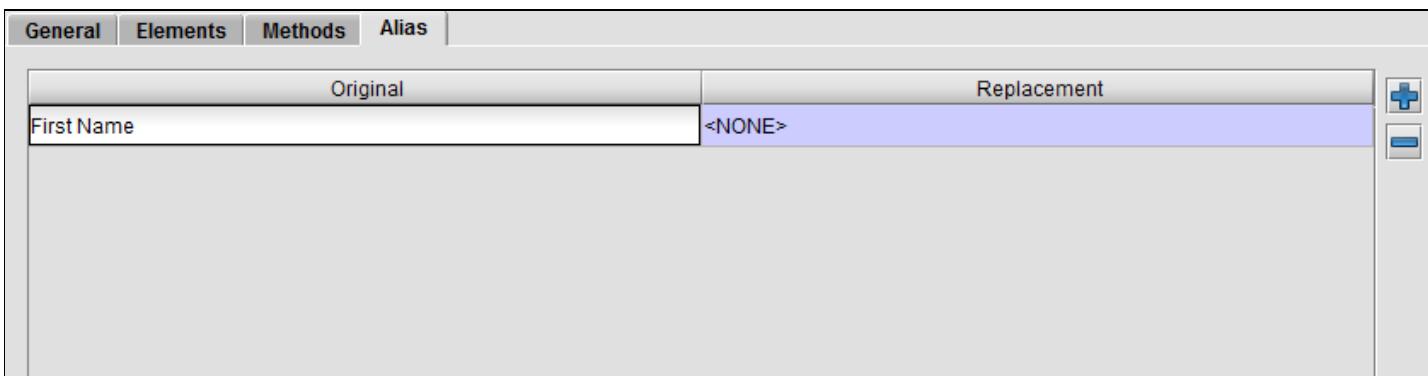
When variables under child data structures are renamed in template metadata, dependent metadata with references to the original variables results in unresolved reference errors. The Alias tab allows you to provide an alias for a renamed variable so that dependent metadata that references the original variable will reference the new variable instead.

To add an alias, complete these steps:

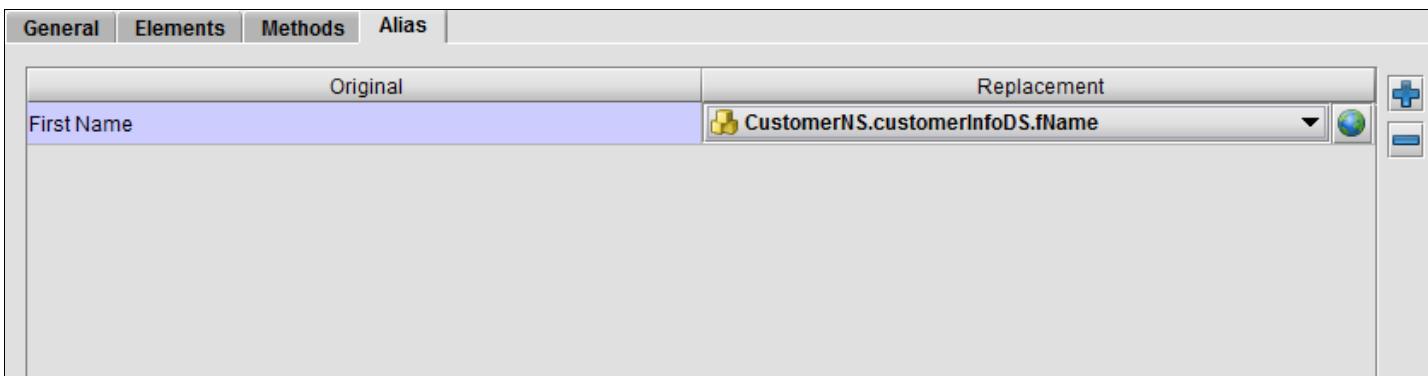
1. From the Alias tab, click the **Add** button.



2. Double-click the **Original** field and specify the original variable. Press the **Enter** key to continue.



3. Click the **Replacement** field of the new row that you have added and select the new variable that you want from the list. Alternatively, you can click the **Select Element** button (with a globe icon) and enter the object name that you want in the field provided.



4. When you save and reload your metadata, all original elements in your metadata are replaced with the replacement elements using this aliases map.

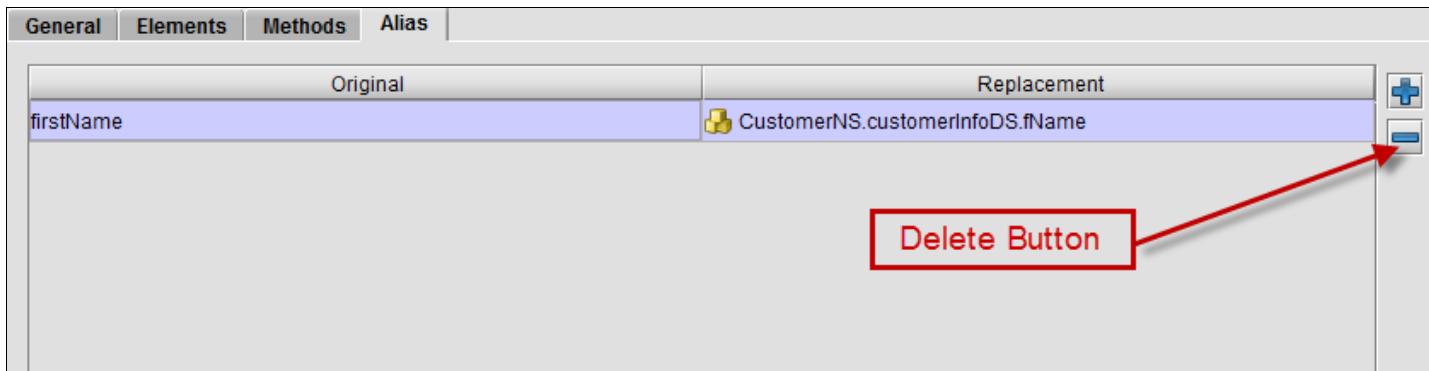
### Notes:

- Both the **Original** and **Replacement** fields are mandatory. If the **Original** field does not contain a value, the row entry is automatically removed. If the **Replacement** field does not contain a value, a validation error occurs.
- Even though the alias information appears under the data structure's Alias tab, the actual information is stored under the metadata header. Therefore,

whenever you make changes to a data structure's alias, the metadata header becomes dirty and not the data structure itself.

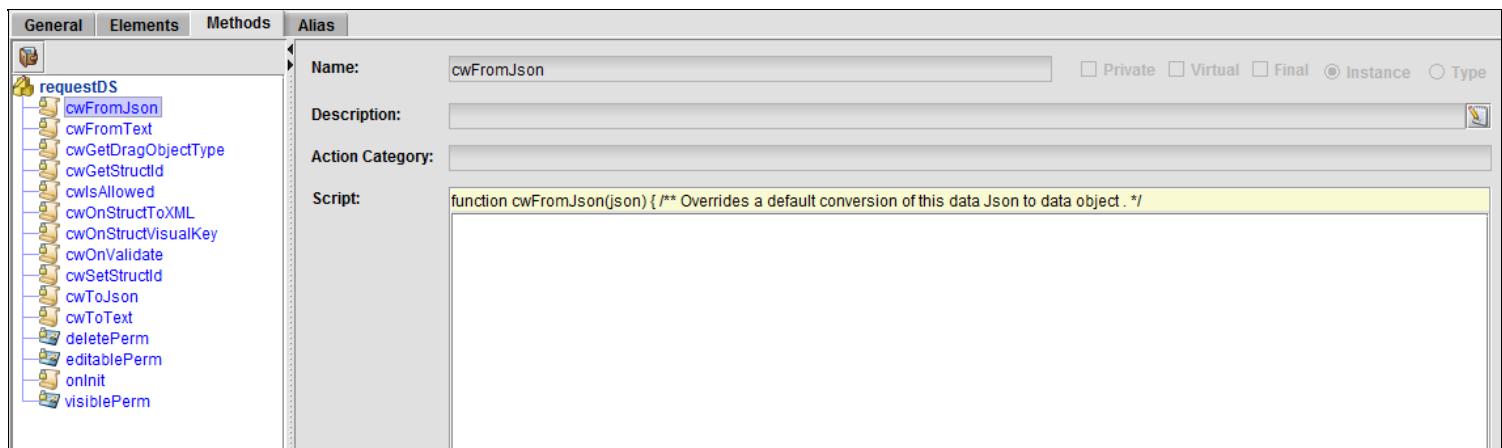
### Delete an Alias

To delete an alias from the Alias tab, highlight the alias that you want and then click the **Delete** button.

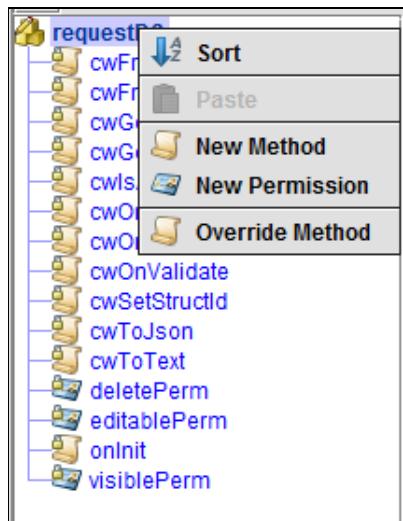


## Data Structures - Methods Tab

The **Methods** tab is used to add and define data structure-level methods and permissions. There are different types of methods that can be defined for the data structure; they are all consolidated into this tab.



The Methods tab is a [script management screen](#) in Velocity Studio, which is composed of the **Method List pane** on the left that lists all scripts of the Document, and **Details pane** that displays the details of a script when selected at the Method List pane. To create a new method, or to override an existing Method of its base Document, right-click the data structure in the Method list pane. A pop-up menu appears.



When you select the **Type** radio button, it means that the method can be invoked directly through the script and does not require an instance object.

To show all base methods, click the **Show base methods** ( ) button. Click the button again to hide all base methods. Base methods appear in blue. Any method highlighted in blue font indicates that you can override it. Methods in black font cannot be overridden.

If you have a long list of methods in the left pane, you can press the first letter of the method you are looking for to jump to the list of methods beginning with that letter. In this example, pressing the **D** key highlights the first instance of a method that begins with the letter D. The method displays in the right pane.

The screenshot shows the 'Methods' tab in the Conceptwave Studio interface. A tree view on the left lists methods under 'requestDS', including 'cwFromJson', 'cwFromText', 'cwGetDragObjectType', 'cwGetStructId', 'cwIsAllowed', 'cwOnStructToXML', 'cwOnStructVisualKey', 'cwOnValidate', 'cwSetStructId', 'cwToJson', 'cwToText', 'deletePerm', 'editablePerm', 'onInit', and 'visiblePerm'. The 'deletePerm' method is currently selected. The main panel displays its details: Name: 'deletePerm', Description: empty, Parameters: '\$psCondition' of type 'com.conceptwave.system.Boolean'. There are also checkboxes for 'Private', 'Virtual', 'Final', 'Instance', and 'Type'.

The following table lists the wide-ranging type of methods that can be added at the data structures level.

#### New Methods

Method Type	Description
Method	Generic user-defined JavaScript function that can be defined in a data structure and explicitly invoked by other scripts; the function is not triggered by other built-in means in the data structure.
Permission	Permission method that can be associated to determine permission for a certain property of the data structure (for example, Element properties). See the <a href="#">Permissions</a> section for details.

#### System-defined Methods

Method Type	Script Name	Parameters	Return Type	Description
Conversion to Data Object	<code>cwFromJson</code>	None.	None.	Document Initialization script that overrides a default conversion of JSON data to a data object.
Conversion to Data Object	<code>cwFromText</code>	None.	None.	Data structure script that overrides a default conversion of text to a data object.
Get Drag Object Type	<code>cwGetDragObjectType</code>	None.	<code>String</code>	Returns the metadata object's type. The purpose of this method is to allow metadata to provide a different type for the object than what is used by the product.
Get Data Structure ID	<code>cwGetStructId</code>	None.	<code>String</code>	Data structure script that returns the data structure ID.
Conversion to XML	<code>cwOnStructToXML</code>	None.	None.	Data structure script that overrides a default conversion of this data structure to XML
Visual Key	<code>cwOnStructVisualKey</code>	None.	<code>String</code>	Data structure script that returns a string representing the visual key. The length of the string should not be longer than 64 characters. If longer, the value is truncated. If this method is empty, the data structure Label is used as visual key.  <b>Note:</b> Since a data structure can consist of other data structures and documents, making changes in child objects devaluates (that is, marks to recalculate) the parent data structure's visual key.
Set Data Structure ID	<code>cwSetStructId</code>	None.	None.	Data structure script that sets the ID for the data structure using the structId parameter.
Conversion to JSON	<code>cwToJson</code>	None.	<code>None</code>	Data structure script that overrides a default conversion of this data object to JSON.
Conversion to Text	<code>cwToText</code>	None.	<code>None</code>	Data structure script that overrides a default conversion of this data object to text.
Delete Permission	<code>deletePerm</code>	None.	<code>Boolean</code>	Delete Permission that specifies whether the participant is allowed to delete an instance of this Document (for example, Document instances in an Order Collection).
Update Permission	<code>editablePerm</code>	None.	<code>Boolean</code>	Update Permission that specifies whether the participant is allowed to update the Document (for example, change Variable of the Document).

<b>Initialization</b>	<i>onInit</i>	None	None	The system method that is triggered on data structure's initialization.
<b>View Permission</b>	<i>visiblePerm</i>	None.	<i>Boolean</i>	View Permission that specifies whether the participant is allowed to view the Document.

For method types that can be created, you can create multiple methods of the same method type (for example, Validation 1, Validation 2...), each identifiable by unique method name. However, for methods that are to be overridden from base data structure, you can only define one method per base method. This rule is applicable to both system-predefined scripts and any additional methods that are defined in non-system-based data structures. All system-predefined methods have empty scripts, except permission methods, which means the default method is to do nothing. All permission methods return *true*, which allow all operations on the data structure by any participant.

For scripts that are already overridden, they appear in the Method list pane and no longer be available at the right-click context pop-up menu.

If this data structure extends from another (non-system based) data structure, and the base data structure has additional methods defined in it, those methods are not displayed in the Methods tab of this data structure.

The method created in Method list pane can be right-clicked, with various commands available; see [script management UI](#) for details.

## Data Type Node

A data type node is a leaf node of the Data Structure hierarchy. It has a mandatory **Element** property that defines the Data Type, Element Type, Document or Data Structure reference for this node.

Note: The XML settings are described in section on XML Settings and the **Array** checkbox is described in section on Array Node.

**General**

<b>Name:</b>	FName
<b>Label:</b>	FName
<b>Description:</b>	<input type="text"/>
<input type="checkbox"/> <b>Array</b> <input checked="" type="checkbox"/> <b>Data type (string)</b> <input type="checkbox"/> <b>Mandatory</b>	
<b>Element:</b>	<input type="button" value="string (XML Schema)"/> <input type="button" value="..."/> 
<b>XML settings:</b>	
<b>XML name:</b>	<input type="text"/>
<b>Length:</b>	<input type="text"/>
<b>Format:</b>	<input type="text"/> <input type="button" value="..."/>
<input type="checkbox"/> <b>Exclude</b> <input type="checkbox"/> <b>Attribute</b> <input type="checkbox"/> <b>Include always</b> <input type="checkbox"/> <b>CDATA</b> <input type="checkbox"/> <b>Text</b>	

## Array Node

An array element is a special type of node that, at runtime, can contain a variable number of instances. An array is specified by checking the **Array** checkbox. Each array instance is of the type specified in the **Element** property (an array of Data Types, Element Types or Documents). The number of instances is further restricted by optional values in the **Minimum** and **Maximum** properties.

### Notes:

- XML settings are described in section on XML Settings.
- To return a validation error list when validating a data structure of type array, use the method `validateEx` in `CwObject` class. For more information on validating data objects refer to the [JavaScript Documentation](#).

General Elements

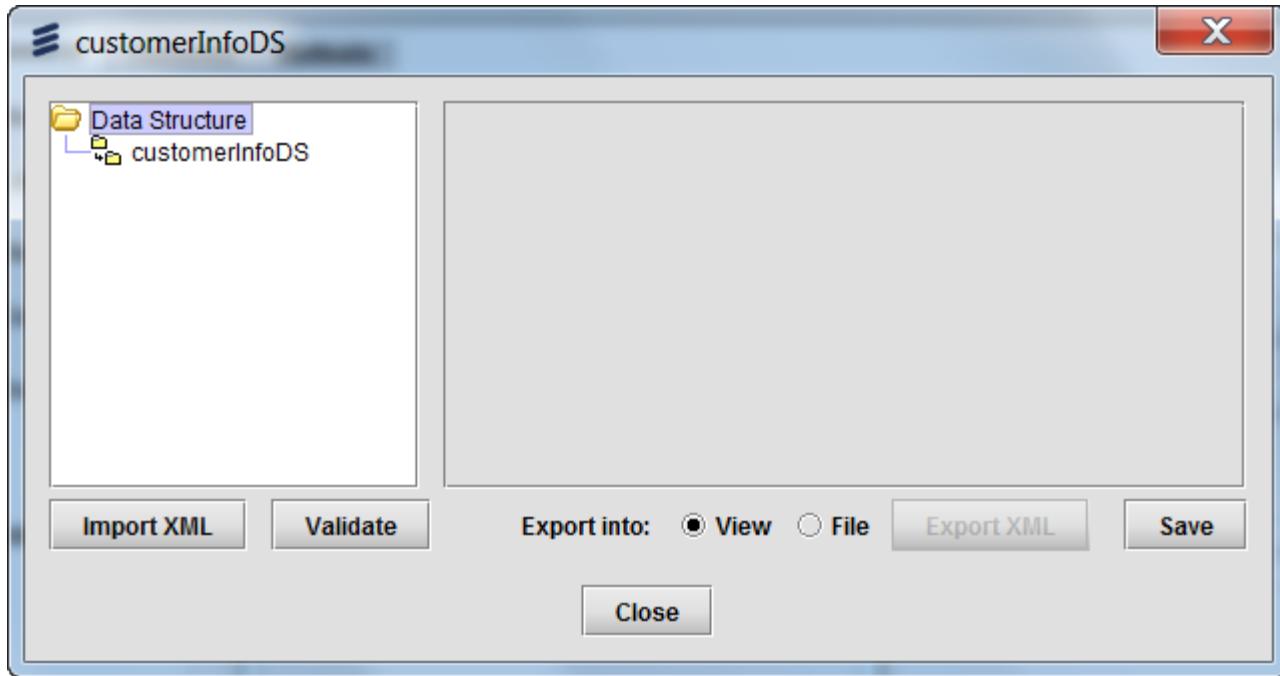
Name:	customerOrder
Label:	customerOrder
Description:	<input type="text"/>
	<input type="checkbox"/> Container <input type="checkbox"/> Data type <input checked="" type="checkbox"/> Array <input type="checkbox"/> Mandatory
Instance:	Minimum <input type="text" value="0"/> Maximum <input type="text"/>
Element:	<input type="text" value="&lt;Any&gt;"/> <input type="button" value="..."/> 
XML settings:	<p>XML name: <input type="text"/></p> <p><input checked="" type="radio"/> Include <input type="radio"/> Exclude <input type="radio"/> Exclude children <input type="radio"/> Include Always</p>
JSON name:	<p>JSON name: <input type="text"/></p>

## Data Structure Operations

The Data Structure right-click pop-up menu has the commands **Add Data Structure**, **Data Import/Export** and **Export to Document** in addition to the common right-click menu items. As mentioned above, clicking the **Add Data Structure** command allows the user to add a new child node.

### Data Import and Export

Clicking the **Data Import/Export** command opens an import/export dialog (see figure below) which allows the user to import data from an XML file, modify it and export the data back into a file. The user can also create new data dynamically.

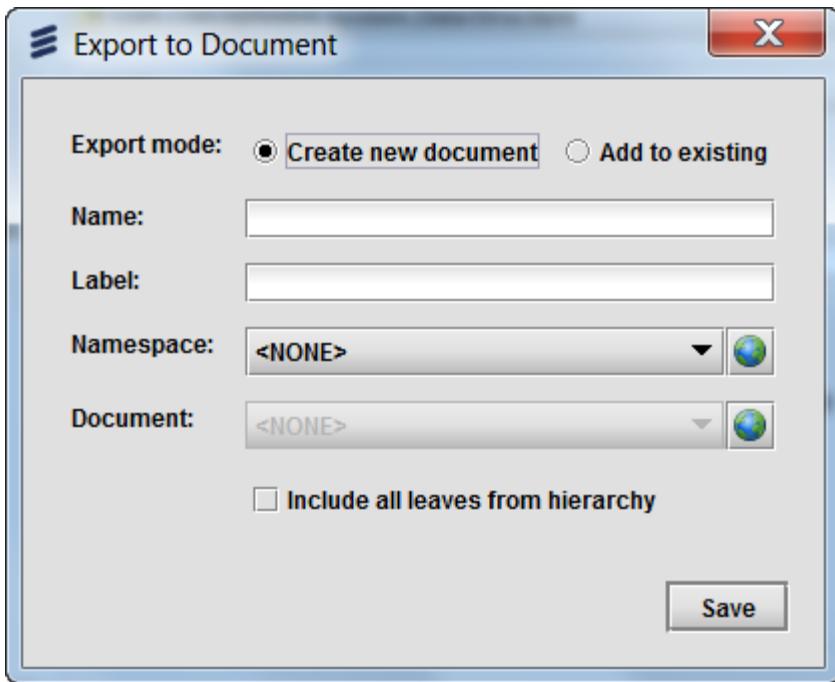


The tree on the left side shows a list of Data Structures, when they are imported from an XML file, or a single Data Structure. The hierarchy of the Data Structures can be expanded. When a tree node is selected, the right side panel allows the user to enter the data type values for containers or the leaf values for Documents. Arrays will appear as folders in the tree. Right-clicking the folder allows the user to add a new instance to the array. If one of the data types in the container is an array, it will appear as a special panel on the right side, which can be used to modify the array (see figure below).

The **Import XML** button imports data from an XML file. The **Validate** button checks the correspondence of the data to the metadata data type defined in the Data Structure and if there is data entered for the leaf nodes that have the **Mandatory** attribute. The **Export XML** button previews the data when the **View** radio button is selected, or saves the data into an XML file when the **File** radio button is selected. The **Save** button saves all changes made in the right side panel before changing selection in the tree.

### Export to Document

Clicking the **Export to Document** command opens the **Export Data Structure to Document** dialog which allows the user to create a new Document or extend existing Documents using the Data Structure leaf nodes. The operation is available on the top level nodes and containers.

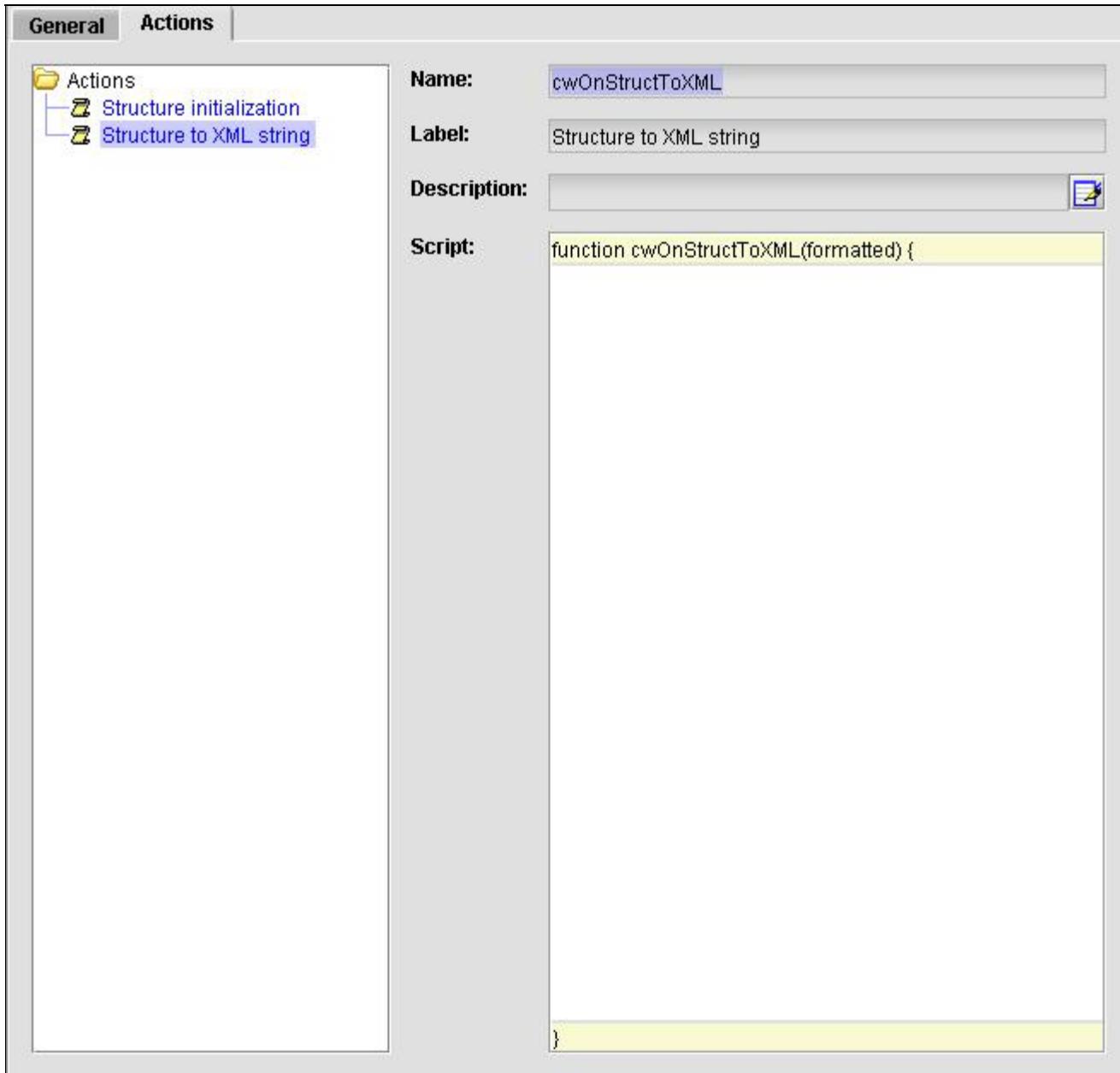


You may choose to create a new Document or add to an existing one.

- If **Include all leaves from hierarchy** is checked, Data Types inside containers will also be exported.
- If unchecked, only the immediate leaves with Data Type will be export to document.
- If there are no immediate leaves which are Data Types, a validation error will be displayed.

## Data Structure Actions

On the Actions tab, by default is the initialization script. Other actions can be added here.



An XML string that the cwOnStructToXML returns will also be returned by toXML() and transform(). This way a user can override the default conversion of data structure to XML.

## Data Structures in Tables

---

Velocity Studio allows you to display and edit data structures in a table element. You can also add a table component to any form (that is, not only under the finder UI) using DataObjectList as a data source variable (that is, any instance of DataObjectList, and not necessarily the finder result).

### Supported Data Structure Column Types

The table element supports all types of columns, such as image and calculated columns. The Reference type element can be bound to either a Select or text column.

The following sections contain examples how to complete the following tasks:

- Create a table with data structures
- Implement table methods
- Initialize the table data source variable at runtime
- Edit the data structure within the table

### Create a Table with Data Structures

To create a table with data structures, complete the following steps:

1. Add a table element to any form, which can be a top-level User Interface.
2. Create a new UI variable. Then, set the variable type to the data structure and select the **Array** checkbox.
3. Set the **Variable** property of the table element to the new UI variable.
4. Add columns under the table and attach them to the first-level data-type leaves of the data structure:
  - Label
  - Text field
  - Checkbox
  - Date
  - Image
  - Calculated field

### Implement Table Methods

When creating a calculated column, create a method with one parameter (the row data structure) and String as the return value. Creating a calculated column is similar to creating calculated columns in finders.

To track a changing selection in a table, create a new **OnSelChanged** method under your UI (provide any name you want for this method). Then, select the method as **Selection Changed** method.

If you require the **Show detail** image column, add the **showDetailAction** method to the UI. Proceed to implement the method, get the selected data structure, and create a UI. An example is as follows:

```
if (this.tableVar.selected && this.tableVar.selected.length > 0) {  
    return new this.tableVar.selected[0].metadata.UserInterface(  
        this.tableVar.selected[0], null);  
}
```

**Note:** The top-level UI does not have any finder functionality, such as the **Back** button. This functionality needs to either be implemented or details can be displayed in a popup window.

The table also supports the double-click method. You can create a new method under the UI with any name and then set it with the **Double Click Method** property for the table.

### Initialize the Table Data Source Variable at Runtime

Although the table variable type in Velocity Studio is an array of data structures, at runtime, the table variable requires a DataObjectList instance. The following is an example of initializing the variable and adding data structures to it:

```
this.tableVar = new DataObjectList(); // tableVar is the table data source
var ds = new DataStructure("ds:TableDS");
ds.id = "101";
ds.string1 = "Str value 1";
ds.number2 = 1;
ds.date3 = new Date();
ds.checkbox4 = true;
this.tableVar.addLast(ds); // add new data structure to the list
```

Every time you change the data object list (for example, by adding, updating, or removing items), call the **updateList** property to refresh the table in your browser:

```
this.tableVar.addLast(ds);
//or
this.tableVar.remove(ds);
this.tableVar.updateList;
```

## Edit the Data Structure within the Table

To create an editable table with data structures, set the **Editable** property to **TRUE** or any Permission method returning true when editable. Columns can then be edited at runtime and edited values are then stored into their corresponding data structure.

**Note:** Adding a new row within the table for editing is not supported. This action is only supported in finders.

## Data Structures in Finders and Reference Fields

Velocity Studio provides you with the following functions pertaining to data structures in finders and reference fields:

- Return a list of data structures from a finder
- Use data structures as reference field values, including displaying a data structure finder on a reference button and in drop-down elements
- Use reference data types and fields in data structures

### Data Structures with Get and Set ID actions, and Visual Key

To use data structures in finders and as a reference, the following data object methods must be overridden and implemented:

- `getDataObjectId()`
- `setDataObjectId()`
- `getVisualKey()`

The following system data structures support these methods:

```
function cwGetStructId(){ // Returns data structure ID as String.  
}  
  
function cwSetStructId(structId){ // Sets the ID for the data structure. The structId is a String  
parameter with an ID.  
}  
  
function cwOnStructVisualKey(){ // Computes the data structure visual key when requested.  
}
```

These data structure actions are available by clicking the **Methods** tab of the data structure property panel. Right-click your data structure and select the action you want.

#### Notes:

- All actions operate with String values. Both ID and visual key are assumed to be strings.
- The data structure ID's length does not have a limit.
- **Get ID** is required for the Output data structure of any script finder.
- The **visual key** action is required for the Output data structure of the finder when it is used as a reference type to display the value in either the **Reference** field or the drop-down menu.
- **Set ID** is required only for the Input data structure of the script finder's Get action when the finder is used as a reference type.

### Script Finders with Data Structures

Script finders can only have data structures as Input and Output parameters. Script finders can also be a combination of document and data structure types (for example, the search is a document and the output is a data structure, or vice versa).

A script finder has at least one property specified (Select output) and implements the Select operation. The Output data structure needs to implement the `cwGetStructId` method to provide a unique structure ID.

Since all finders share the same finder data object (either `CwfDataFinder` Java class or the Finder script type), the data object and all metadata finder objects accept `CwfDataObject` (`CwObject`) as search input and output.

**Note:** All properties, method names, and function parameter names remain the same and use the old "document" notation.

#### Caution Using the Finder Result Object (DataObjectList)

Caution is necessary when copying data structure objects from one finder result or `DataObjectList` to another.

Suppose a finder needs to return a subset of objects from an existing DataObjectList received by another action (for example, by invoking another finder). If the output is a document, when adding this document to the script finder result, a copy of this document is added. If the document has a parent document, which can be a finder or an order, the document is replicated to avoid damaging or changing the original list.

Additionally, when adding to the DataObjectList, all data objects are automatically linked. As a result, the new DataObjectList can be iterated by the object.next() method.

Data structures do not support replication, so they will add the original object to the new list, changing its parent and relinking the next() object. As a result, the original DataObjectList becomes broken. For this reason, the concatenate method is not supported for non-document lists.

## Reference Data Type with Data Structures

Finders used in reference types, in addition to a regular data structure finder, must do the following:

- Implement the Get operation
- Specify both the Get input and Get output types

Additionally, the Get input data structure must implement the cwSetStructId method, and the Get output data structure must implement both the cwGetStructId and cwOnStructVisualKey methods. Data structures can have a data type element with a reference data type that can use both document and data structure finders.

### Get Reference Field's Value

Internally, the data structure keeps a reference key (ID) as the value for the element. However, when the data structure retrieves the value in scripts or when displaying it in either the **reference** or drop-down field, the data structure returns the ReferenceValue object.

The ReferenceValue object supports the following properties:

- key
- visualKey
- reference

**Note:** The **reference** property returns the reference object, which is either a document or data structure.

### Set reference field's value

To change the value of the **reference** field, using either the key string (ID) value or the ReferenceValue object are acceptable.

### Retrieve the "Reference" object

When a finder needs to retrieve a referenced data structure by ID, the reference cache accesses the Interface Data Cache (CwfInterfaceDataCache) using the ID and the finder's full name as a reference type. If the object is not found in the cache, the finder invokes its Get operation.

### Reference Fields in the Data Structure UI

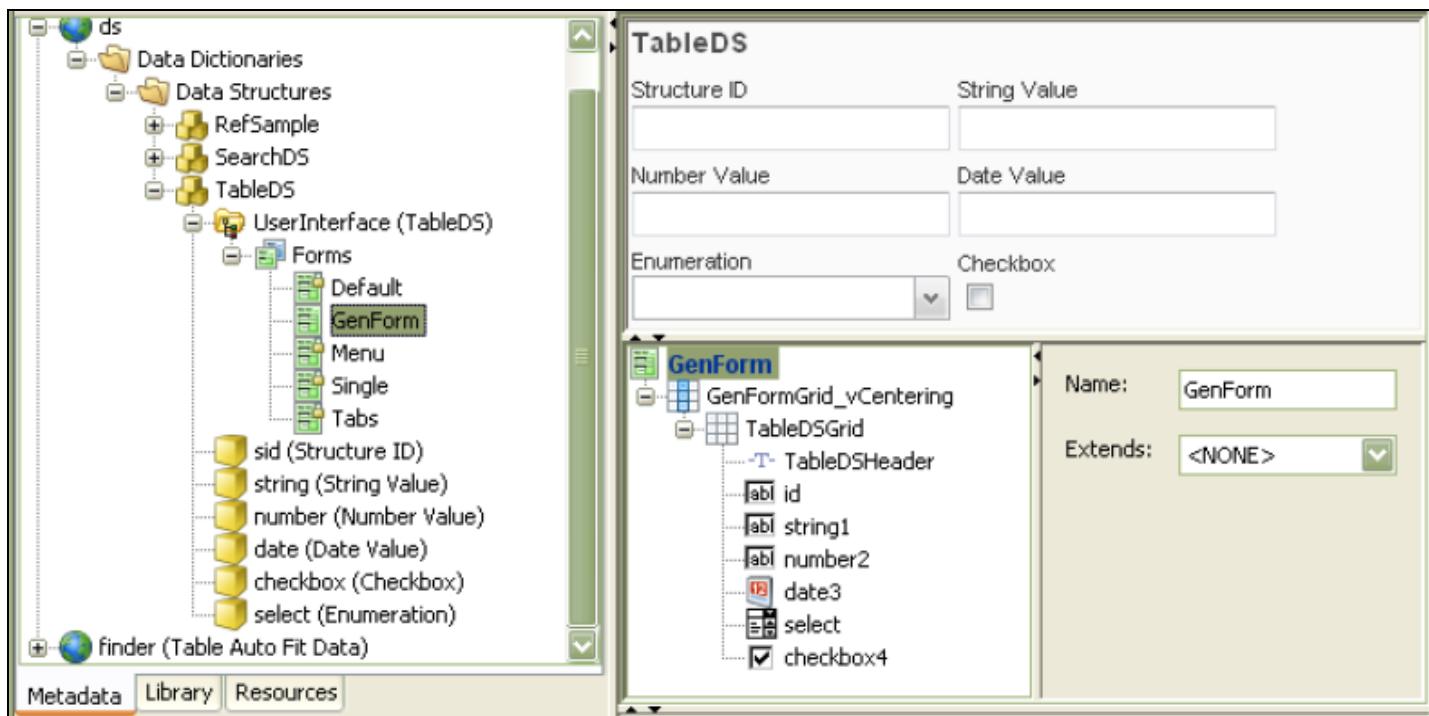
Technically, data structures can have reference types at any level. At this stage, displaying and editing the **reference** field is only implemented at the first level for data structures (direct leaves, which are document leaves).

## Sample: Script Finder with Data Structure Output

This sample shows how to create a script finder (finder.SimpleDsFinder) that returns a list of data structures as output and displays it in the result table.

1. Create a new namespace called **ds**.
2. Create a simple data structure, **ds.TableDS**, with different data type items:
  - o **sid**: cwf.String64 (this item will be a data structure ID)
  - o **string**: cwf.String64
  - o **number**: com.conceptwave.system.Decimal
  - o **date**: com.conceptwave.system.Date
  - o **checkbox**: com.conceptwave.system.Boolean
  - o **select**: cwf.severity
3. Right-click the data structure's **UserInterface** and select the **Generate Form** option. This form, GenForm, serves as a detail form in the finder.

**Note:** You can use the Default form that displays the data structure as a tree. Since the ds.TableDS data structure is a simple one, a tree is not necessary.



4. To use the data structure in a script finder, it requires implementing **cwGetStructId**, which provides a unique identifier for the data structure in the result table. Click the **Methods** tab of the data structure's property panel. Proceed by right-clicking the **TableDS** root and adding the following to the **cwGetStructId** method:

```
return this.sid;
```

**Note:** The data structure has a pair of methods, **cwGetStructId** and **cwSetStructId**, for the ID. For this finder, we only need the former method. The setter is required when a data structure is used in a reference finder's **Get Input** parameter for the GET operation. See the [Reference finder sample](#) for details.

5. Create a simple data structure called **ds.SearchDS**, which will be used as search input. Add data type elements for search criteria, including the following:
  - o **sid**: cwf.String64 (Search the data structure by ID)
6. Generate a form for **ds.SearchDS** and call it **SearchForm**.

### Create a Script Finder

The following sample shows how to create a script finder. Perform these steps:

1. Create a new namespace called **finder**.

2. Create a script finder called **finder.SimpleDsFinder** with the following parameters:

- **Select Input:** ds.SearchDS
- **Select Output:** ds.TableDS

**Note:** You can use a document for the search input. The reverse also holds true, where you can use the data structure as a search object and return documents as a result.

3. Click the **View** tab and change the SimpleDsFinderView default view:

- **Search Form:** ds.SearchDS.UserInterface.Forms.SearchForm
- **Show search form:** yes

4. Click the **Advanced** tab. For the **Detail Form**, select **ds.TableDS.UserInterface.Forms.GenForm**.

5. Implement the cwOnFinderSel method of the script finder to return a list of data structures matching the search criteria:

```
// searchData parameter is the search input:  
// ds.SearchDS data structure  
var searchById = searchData != null && searchData.sid != null && searchData.sid != "";  
var list = new DataObjectList();  
var ds;  
  
if (!searchById || searchData.sid == "101") {  
    ds = new DataStructure("ds:TableDS");  
    ds.sid = "101";  
    ds.string = "Str value 1";  
    ds.number = 1;  
    ds.date = new Date();  
    ds.checkbox = true;  
    ds.select = "1";  
    list.addLast(ds);  
}  
  
if (!searchById || searchData.sid == "102") {  
    ds = new DataStructure("ds:TableDS");  
    ds.sid = "102";  
    ds.string = "Str value 2";  
    ds.number = 2;  
    ds.date = new Date();  
    ds.checkbox = false;  
    ds.select = "2";  
    list.addLast(ds);  
}  
  
if (!searchById || searchData.sid == "103") {  
    ds = new DataStructure("ds:TableDS");  
    ds.sid = "103";  
    ds.string = "Str value 3";  
    ds.number = 3;  
    ds.select = "3";  
    list.addLast(ds);  
}  
  
return list;
```

## Create an Application User Interface for the Sample Finder

To create an application user interface for the finder in this sample, complete these steps:

1. Add a new UserInterface called **finder.app** that extends the **com.conceptwave.system.Application** interface.

2. Create a new User Action method called **SimpleDsFinder**, and set these properties:

- **Object:** finder.SimpleDsFinder
- **Form:** Default

3. Override the **Menu** form of the application UI.

4. Add a new Menu Item called **SimpleDsFinder** and set its **Click method** property to the **SimpleDsFinder** user action.

At runtime, the result table for **finder.SimpleDsFinder** is as follows:

SimpleDsFinder

SimpleDsFinderView

SearchDS

Search ID

Structure ID	String Value	Number Value	Date Value	Checkbox	Enumeration
► 101	Str value 1	1	02/28/2011	<input checked="" type="checkbox"/>	Information
► 102	Str value 2	2	02/28/2011	<input type="checkbox"/>	Warning
► 103	Str value 3	3		<input type="checkbox"/>	Error

Search

3 rows fetched.

Local intranet

100%

The detail form for finder.SimpleDsFinder looks like the following:

SimpleDsFinder

SimpleDsFinderView

TableDS

Structure ID	String Value
101	Str value 1
Number Value	Date Value
1	02/28/2011
Enumeration	Checkbox
Information	<input checked="" type="checkbox"/>

Done

Local intranet

100%

## Sample: Reference Finder with Data Structure Output

This sample shows how to create a reference finder (`finder.ReferenceDsFinder`) with a data structure finder. This example implements a reference data type with a data structure finder.

This reference type can be used both for document variables and data structure data type elements. The value of such a reference type is a special `ReferenceValue` object with its **key** equal to data structure ID and the display property **visualKey** is the visual key of the data structure. Thus, the output data structure in the reference finder must implement the `cwGetStructId` and `cwOnStructVisualKey` methods.

Another requirement for the reference finder is that it must support the GET operation to retrieve the data structure by ID. For the GET operation, both **Get Input** and **Get Output** must be specified. Additionally, the **Get Input** data structure it is required to implement the `cwSetStructId` method, to be able to pass the ID value to the GET script.

In scripts, the reference data structure can be retrieved (read-only) by the of the `ReferenceValue` object's **reference** property.

### Create a Reference Type

To create a reference type, perform these steps:

1. Using the `ds.SearchDS` and `ds.TableDS` data structures from the [sample script finder](#) example, create a new script finder called `finder.ReferenceDsFinder` with the following parameters:
  - o **Select Input (optional)**: <NONE> or `ds.SearchDS`
  - o **Select Output**: `ds.TableDS`
  - o **Get Input**: `ds.SearchDS`
  - o **Get Output**: `ds.TableDS`
2. Locate the `ds.SearchDS` data structure and implement the `cwSetStructId` method by adding the following:

```
// structId function parameter contains the ID to set this.sid to structId
this.sid = structId;
```
3. Implement the `cwOnStructVisualKey` method of the `ds.TableDS` data structure by doing the following
  - o Return "Ds: " + this.sid + " " + this.string.
  - o For `finder.ReferenceDsFinder`, implement the `cwOnFinderSel` method, and then copy the script from the `finder.SimpleDsFinder` script method.
4. For `finder.ReferenceDsFinder`, implement the `cwOnFinderGet` method as follows:

```
var ds = new DataStructure("ds:TableDS");
ds.sid = document.sid;
switch (ds.sid) {
    case "101":
        ds.string = "Str value 1";
        ds.number = 1;
        ds.date = new Date();
        ds.checkbox = true;
        ds.select = "1";
        break;

    case "102":
        ds.string = "Str value 2";
        ds.number = 2;
        ds.date = new Date();
        ds.checkbox = false;
        ds.select = "2";
        break;

    case "103":
        ds.string = "Str value 3";
        ds.number = 3;
        ds.select = "3";
        break;

    default:
        ds.string = "Unknown";
        break;
}

return ds;
```

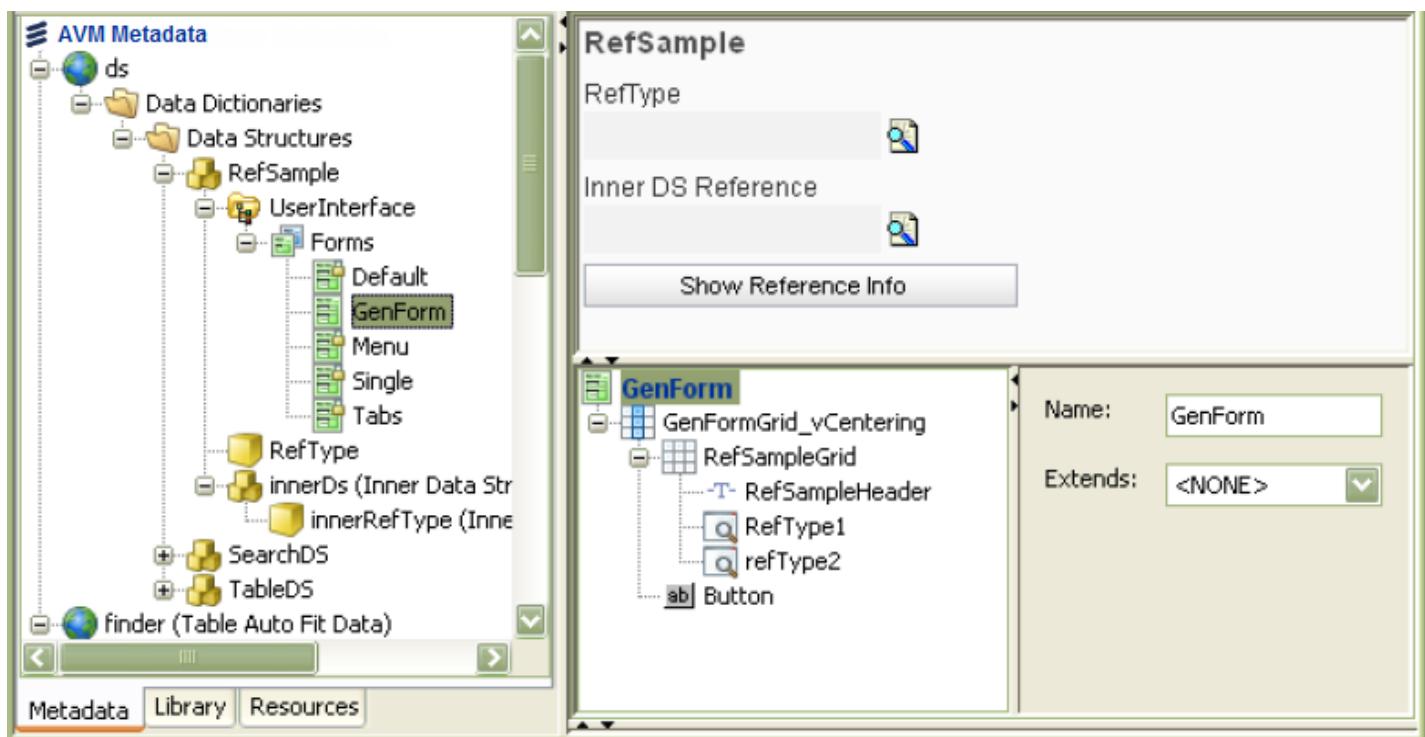
5. Create a new Data Type called **finder.DSReferenceType** that extends the **cwf.sReference** base type. Set its properties as follows:

- o **Finder:** finder.ReferenceDsFinder
- o **Length:** 64
- o **Element Properties:** Reference field

## Use the Reference Data Type

The Reference type can be used on any level of a data structure. In this example, a first-level data type element is created, followed by another data type element inside an inner data structure container.

1. Create a new data structure called **ds.RefSample**.
2. Create a child data type element called **RefType** under the data structure with the following properties:
  - o **Data Type:** yes
  - o **Element:** finder.DSReferenceType
3. Create a container child called **innerDs** with the following properties
  - o **Container:** yes
  - o **Extends:** <NONE>.
4. Create a new data type element in the **innerDs** container called **innerRefType** with the following properties:
  - o **Data Type:** yes
  - o **Element:** finder.DSReferenceType
5. Generate a form for the **RefSample** data structure called **GenForm**. By default, it creates fields only for first-level elements.



6. Change the GenForm form by adding a reference field to the **RefSampleGrid** element, and then bind it to the **model.innerDs.innerRefType** Variable.
7. Add a new **Show Reference Info** button to the form, create a new method called **showReferencelInfo** under **UserInterface**, and then set it as your button's **Click Method**:

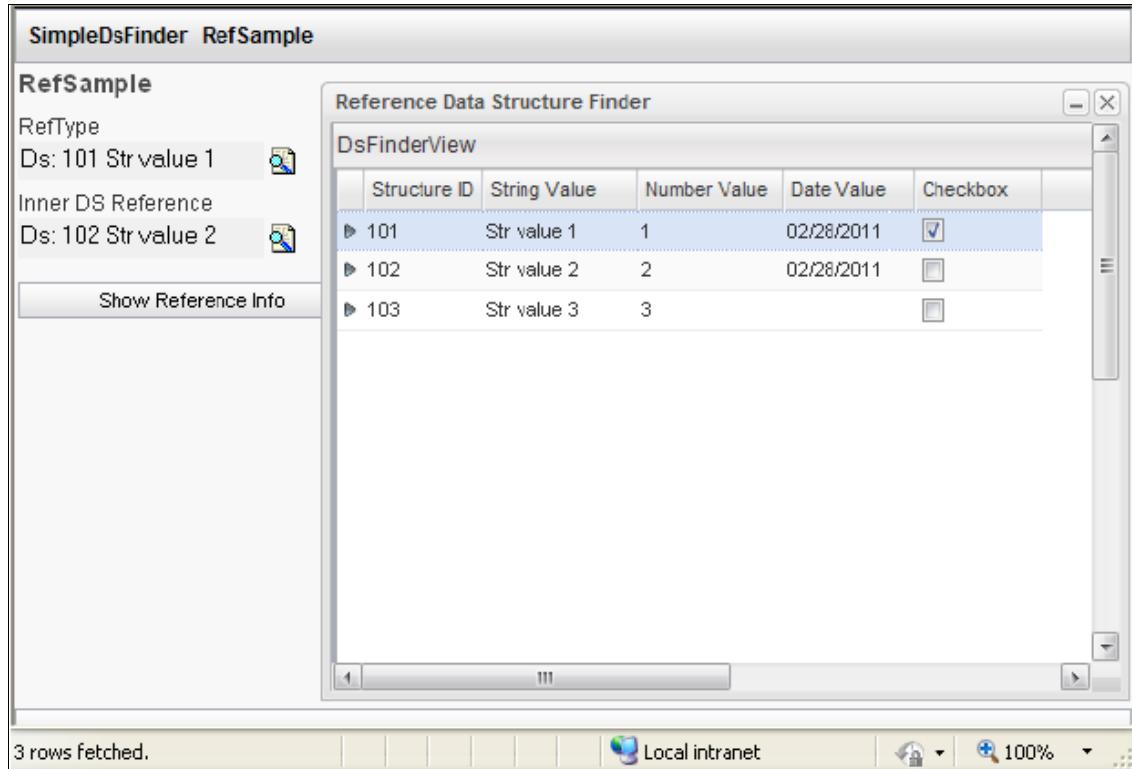
```
var message = "";

if (this.model.RefType == null) {
    message = "RefType value is null";
}
else if (this.model.RefType.reference == null) {
    message = "RefType.reference is null";
}
else {
    message = "RefType: id=" + this.model.RefType.reference.sid + ", string=" +
    this.model.RefType.reference.string;
}

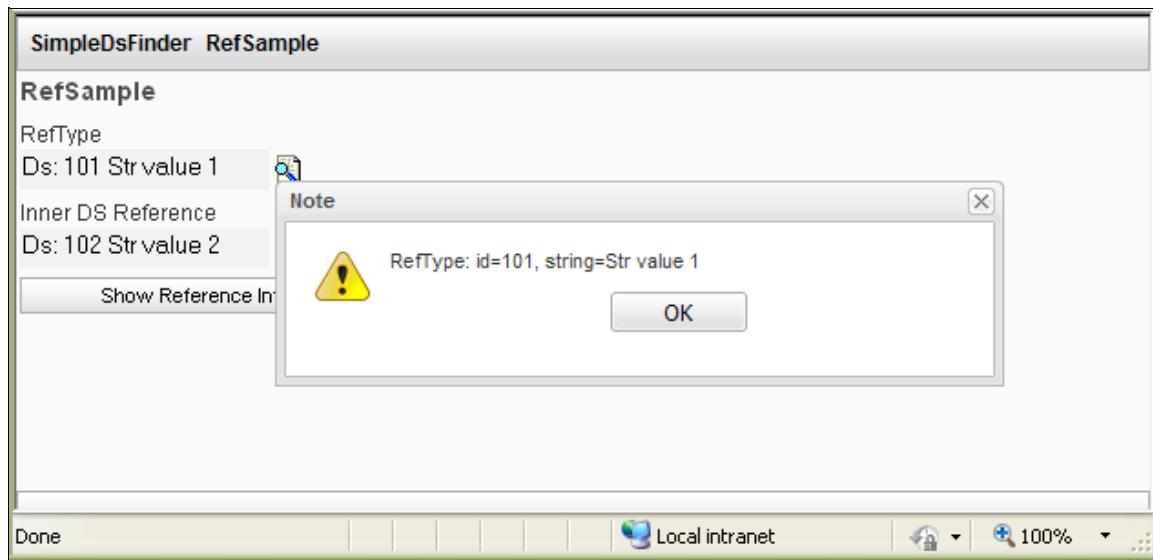
Global.showUserMessage(message);
```

8. Create a new User Action method in the finder.app application UI called **RefSample** with the following:
  - o **Object:** ds.RefSample
  - o **Form:** GenForm
9. Add a new Menu Item called **RefSample** on the application's **Menu** form, and then set its **Click Method** property to **RefSample**.

The reference finder appears in the popup dialog when the **Show Reference Info** button is clicked.



Clicking the **Show Reference Info** button retrieves the data structure object through the **Reference** property.



#### Notes:

- If a reference variable is bound to a Select field, for the drop-down choices, it runs the reference finder and automatically maps the source data object (document or data structure) to the search object (document or data structure), so that the finder can return the result depending on the source object values. The data structure selected in the Select Input field of the reference finder must be of same metadata type as the source object, or have a default conversion map that maps the source data structure to the search data structure.

- The mapping is done only for the Select fields. A reference field with the popup finder can be reached as this.parent in the cwOnFinderSel script, if the source object is not mapped to the search object, but it is set as the finder parent.

## Remove and Copy Data Structure Types

---

The following sections describe how to remove and copy data structure types.

### Remove a Data Structure

Suppose that you have the following structure:

```
<data><fieldA>valueA</fieldA></data>
```

When setting a data structure field to null (for example, `date.fieldA = null`), it removes the field from memory and the element is not created in the XML. When you set a data structure field to an empty string, it leaves the field in memory and creates an empty element in the XML.

### Copy a Data Structure

To copy a data structure, use the `mapToStructure` API in `DataStructure` to make a duplicate data structure and set the conversion map parameter to null.

# Database Maps

---

A database map allows you to map a data structure to a database and its tables. The database map can exist as children of other metadata elements. In Velocity Studio's navigation pane, you can find your database maps in the **Data Dictionaries** folder in each namespace.

This section contains the following information:

- **[Create a Database Map](#)**

This page describes how to create a database map and create a SQL script for a database map.

- **[Database Maps General Properties](#)**

This page allows you to view and define a database map's general properties.

- **[Database Maps Mapping Properties](#)**

This page provides a number of mapping options for database maps, including the following:

- Map top-level [variables](#) and container variables.
- Map a [data object variable](#) to a table column.
- Map [predefined system attributes](#) to a table column, which is used only in insert and update operations.
- Define [from and to scripts](#) for a table column that map data from and to this table column, respectively.
- Use [conversion map support for an object map that has a foreign key](#)

- **[Database Maps Driver Parameters Properties](#)**

This page allows you to define its schema, and driver read and write parameter properties.

- **[JavaScript and Query Support](#)**

This page describes both JavaScript and query support for database maps.

## Create a Database Map

---

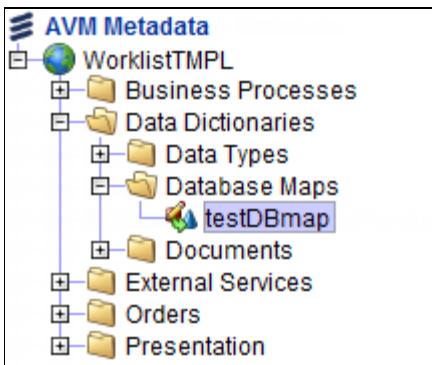
To create a new database map, complete these steps:

1. Right-click a namespace and select **New**.
2. The **New Metadata Object** wizard appears as a popup. Expand **Data Dictionaries**, select **Database Map**, and then click the **Next** button.
3. Step two of the wizard appears. Enter the **Name** of the database map.
4. You must enter a value in either the **Extends** or **Target** fields for your database map, as they both cannot be set to none. Then, click the **Finish** button.

OR

1. Right-click the **Data Dictionaries** folder in a namespace, if it is present. Select **New Database Map**.
2. Follow step 3 from the previous procedure.

After creating your database map, the icon representing your newly created database map appears under the **Data Structures** folder:



A database map has the following tabs in Velocity Studio:

- [General](#)
- [Mapping](#)
- [Driver Parameters](#)

## Create SQL Script for a Database Map

Support for DDL (Data Description Language) is available for each physical connection based on the defined mapping. You can generate an SQL script for a database map by right-clicking the database map that you want from the left menu, and then selecting **Create SQL** from the right-click menu. You are then prompted to connect to your database before specifying the name and where you want to save your SQL script.

## Database Maps General Properties

The database map's General tab allows you to define its general properties.

General    Mapping    Driver Parameters

Name: testDBmap     Private     Restricted     Deprecated

Label: testDBmap   

Description:

Extends: <NONE>

Target: worklist.data.action (Action)

Default     Conflict Resolution     Skip DB schema generation     Audit Trail

The following table describes the common fields for the database map's General tab:

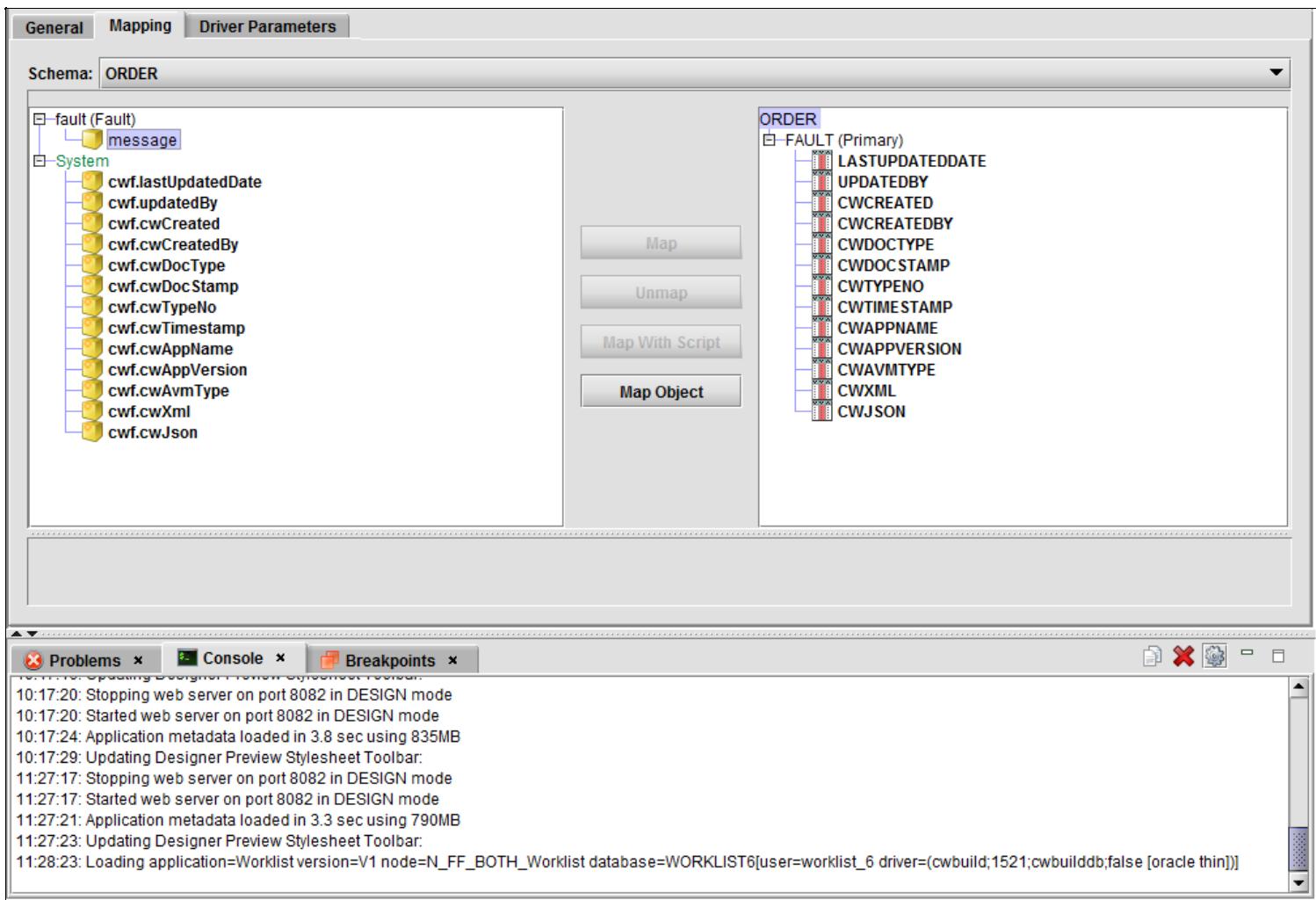
Field	Description										
Name	This field contains the name of the database map metadata object (used in scripts; must conform to JavaScript naming conventions). It is read-only after its creation. To change it, use the <b>Rename</b> command in the popup menu by right-clicking the metadata object.										
Private	If checked, the metadata object is hidden from display when the metadata is packaged as product metadata (that is, library).										
Restricted	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as product metadata (that is, library).										
Deprecated	If checked, the metadata object is designated for archival only. The object becomes hidden from all metadata object selection lists.										
Label	This field contains the database map object's display label.										
Description	This field represents a brief description about the database map for documentation.										
Extends	This field is of Metadata type and allows the database map to extend an existing map. Click the drop-down menu and select the map that you want to extend. This property is neither enabled, nor visible for non-top-level database maps.										
Target	This field specifies the data structure that is to be mapped to the column.										
Default	This field determines whether this database map is the default map to use for the given schema, for the given metadata. By default, this field is unchecked and can be changed. For a non-top-level database map, this field is automatically set to true, and is neither changeable, nor visible.										
Conflict Resolution	When selected, this property enables conflict resolution on the database map. When you create a variable with any of the following data types and they are mapped to a database column, the values are internally generated during the save or update operation:  <table border="1"><thead><tr><th>System Data Type</th><th>Description</th><th>Database Column Requirement</th></tr></thead><tbody><tr><td>CWDOCSTAMP</td><td>This system data type denotes the batch's base 64 sequence number.</td><td>VARCHAR with a length 9 character.</td></tr><tr><td>CWTIMESTAMP</td><td>This system data type represents the timestamp.</td><td>Either DATE or TIMESTAMP.</td></tr></tbody></table>		System Data Type	Description	Database Column Requirement	CWDOCSTAMP	This system data type denotes the batch's base 64 sequence number.	VARCHAR with a length 9 character.	CWTIMESTAMP	This system data type represents the timestamp.	Either DATE or TIMESTAMP.
System Data Type	Description	Database Column Requirement									
CWDOCSTAMP	This system data type denotes the batch's base 64 sequence number.	VARCHAR with a length 9 character.									
CWTIMESTAMP	This system data type represents the timestamp.	Either DATE or TIMESTAMP.									
	<b>Note:</b> These system data types must be a part of the data structure leaf to use them in the conflict check. If you have enabled this property, but the system type mapping does not exist, a warning message appears.										
Skip DB schema	There are cases in which database maps used on external database services does not require creating the table structure for these maps. When selected, this property skips generating the DDL for the specified database map and its corresponding API methods to										

<b>generation</b>	set or get the property value.
<b>Audit Trail</b>	<p>This field determines whether you want to track changes to mapped data structures. By default, this field is unchecked, which indicates that changes are not tracked. To track changes to mapped data structures, select this checkbox.</p> <p><b>Note:</b> To successfully track changes to mapped data structures, other audit settings must be set.</p>

## Database Maps Mapping Properties

The database map's Mapping tab provides a number of mapping options, including the following:

- Map top-level [variables](#) and container variables.
- Map a [data object variable](#) to a table column.
- Map to multiple tables, which is automatically enabled or disabled, based on driver support.
- [Map predefined system attributes](#) to a table column, which is used only in insert and update operations. This mapping works similarly to system attributes in a Document object, except that those attributes are not part of source data object. This mapping can be used to insert or update columns with values such as userId and current date.
- Define [from and to scripts](#) for a table column that map data from and to this table column, respectively. This feature can be used to map data from inner containers or to create composite values.
- Use [conversion map support for an object map that has a foreign key](#)



The following table describes the common fields for the database map's Mapping tab:

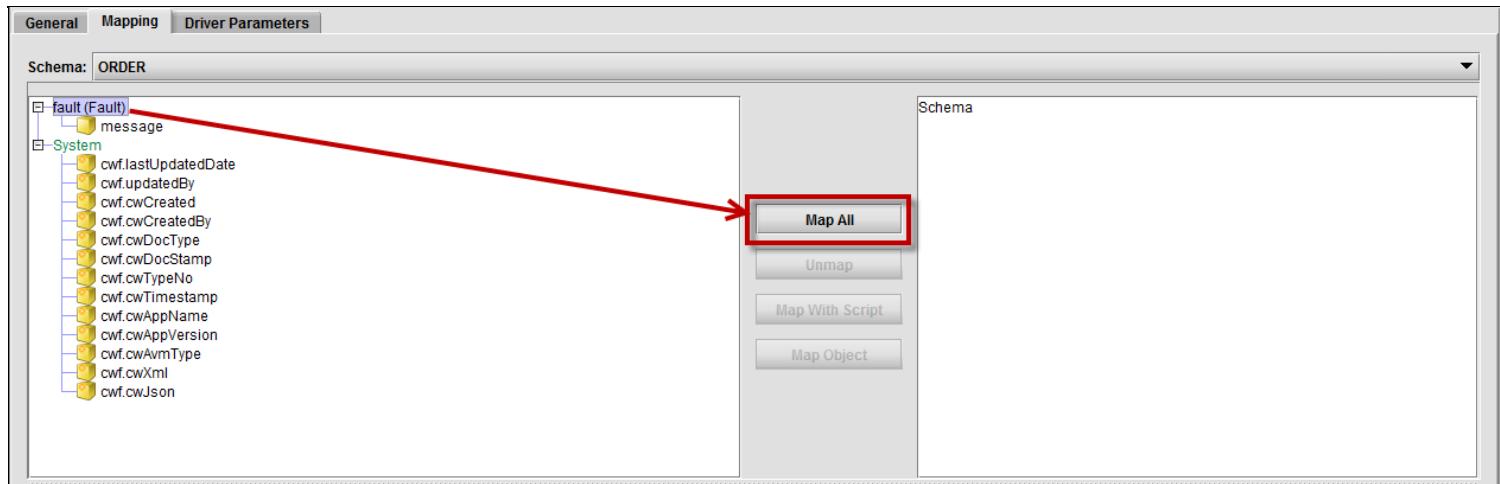
Field	Description
<b>Schema</b>	This field is of String type and denotes the database schema's name. Click the drop-down menu and select a schema from the list, and then click the <b>Connect</b> button to connect to the database. This field can only be changed when Velocity Studio has a database connection.
<b>Source</b>	This field represents the source object variable that you want to map to your database table. This field appears when you select a database map item that has already been mapped.
<b>Audit</b>	This field appears when you select a database map item that has already been mapped. You can <a href="#">track changes to mapped data structures</a> by selecting this checkbox. By default, auditing is disabled.

This page describes the following:

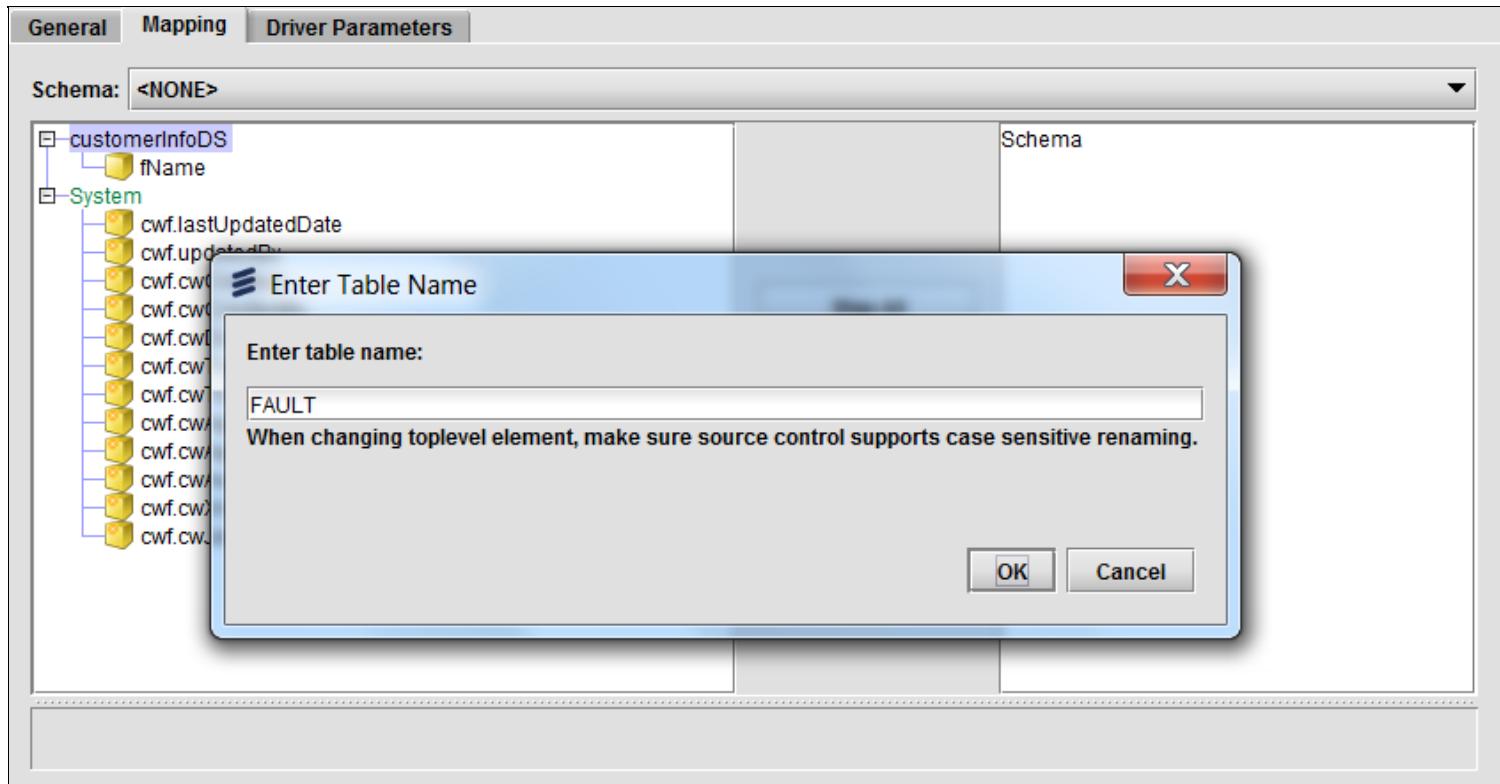
- The Mapping tab's [user interface](#) elements
- How to [add a database column](#)
- How to [rename a database column](#)
- How to [re-arrange database column nodes](#)

## User Interface

The section on the left side of the Mapping page, under the **Schema** field, represents the target data structure, showing all fields that can be mapped to another database or tables. Selecting the data structure on the left side of the page allows you to click the **Map All** button, which maps all identical fields simultaneously. The identical fields are based on the variable's data type.



After clicking the **Map All** button, the Enter Table Name dialog appears, which prompts you to enter the table or database name. Click the **OK** button to continue.



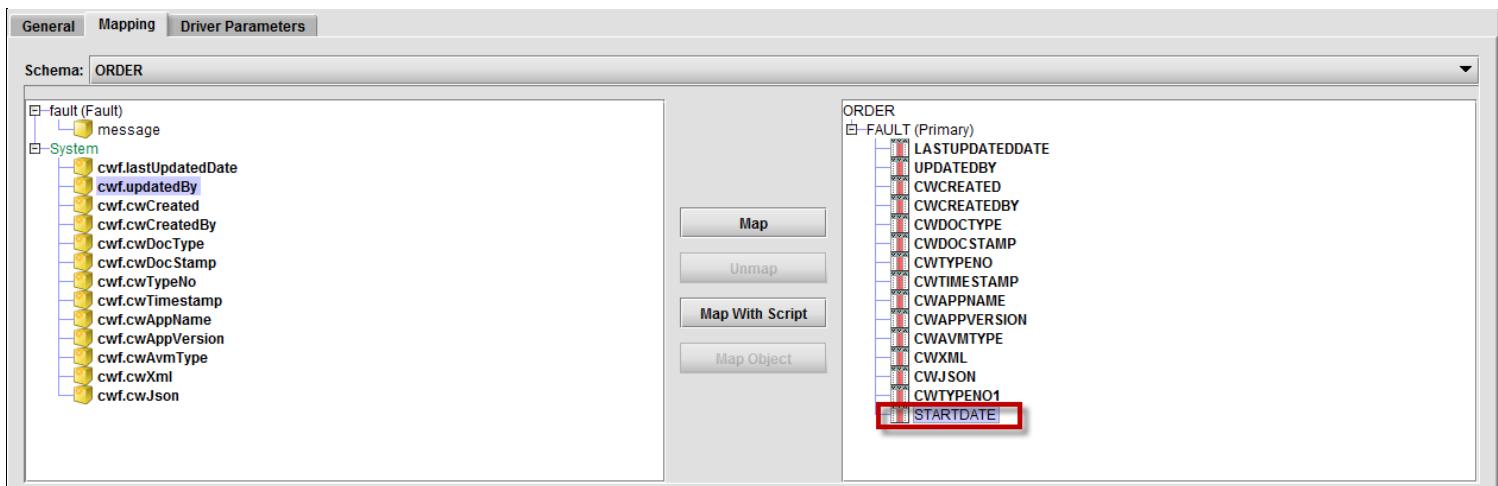
On the right side of the page, all fields are added under the schema name in the same order as items in the target data structure.

## Add a Database Column

You can add a database column to your database table by right-clicking your database table on the right side of the page, and then selecting **Add Column**.

The Enter Column Name dialog appears. Enter the database column's name, and then click the **OK** button.

The newly created database column appears on the right side of the page.



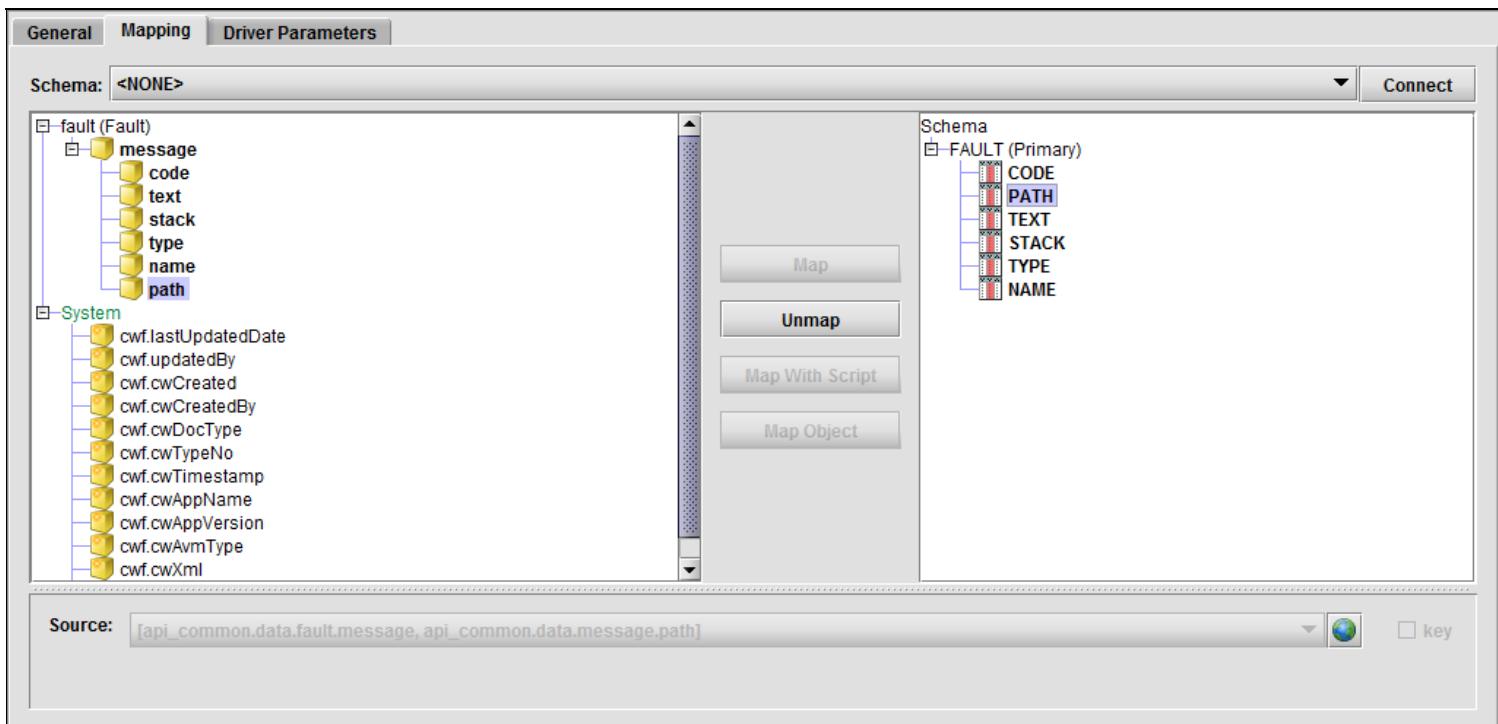
## Rename a Database Column

You can rename a database column by right-clicking your database column on the right side of the page, and then selecting **Rename Column**.

The Enter Column Name dialog appears. Enter the new name of the database column's name, and then click the **OK** button.

## Re-arrange Database Column Nodes

You can re-arrange database column nodes under the same table from the Mapping tab by selecting the node that you want, and then dragging it either up and down. In this example, the path node that was originally ordered last in the source has been moved to be after the CODE node.



### Notes:

- Database column nodes cannot be moved ahead of column nodes that are inherited from the base mapping.
- The order of database column nodes from the base cannot be changed.

## Map a Simple Type Variable

---

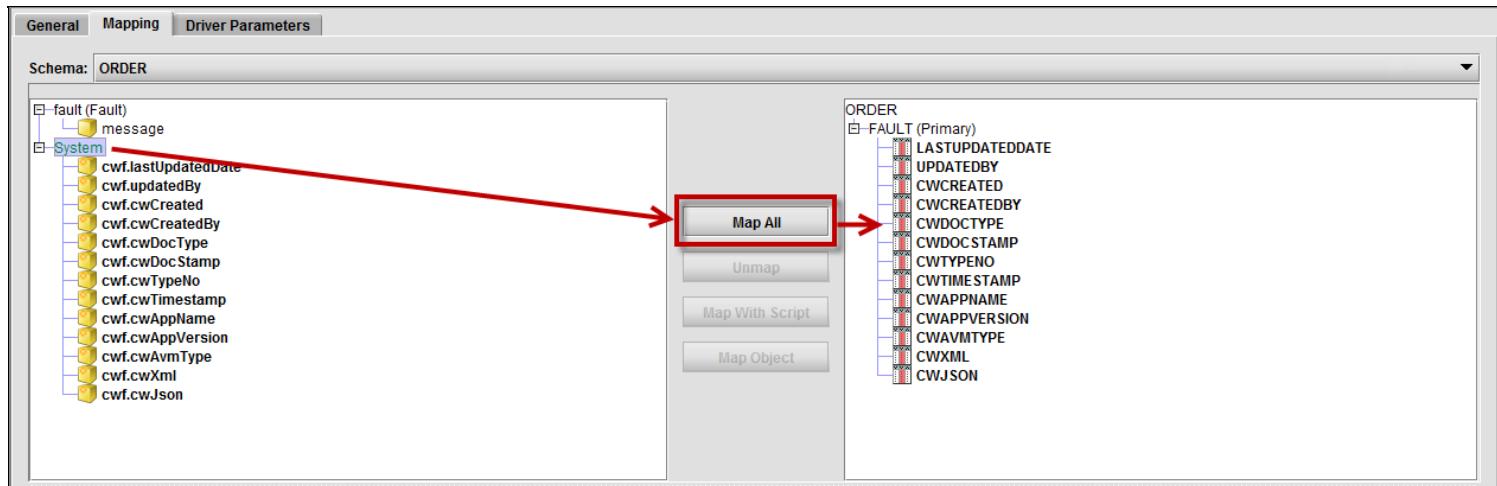
You can map a simple type variable to columns. You can map these simple type variables with database columns:

- Boolean
- Date
- DateTime
- Decimal
- Integer
- String
- Timestamp

This mapping allows you to specify a key column by selecting the database column, and then selecting the **Key** checkbox field. It is mandatory to specify the key column. Otherwise, the database operation does not work.

## Map a System Type

You can map all fields from a system type by selecting the system type on the left side of the page, and then clicking the **Map All** button. Your mapped database columns appear on the right side of the page.

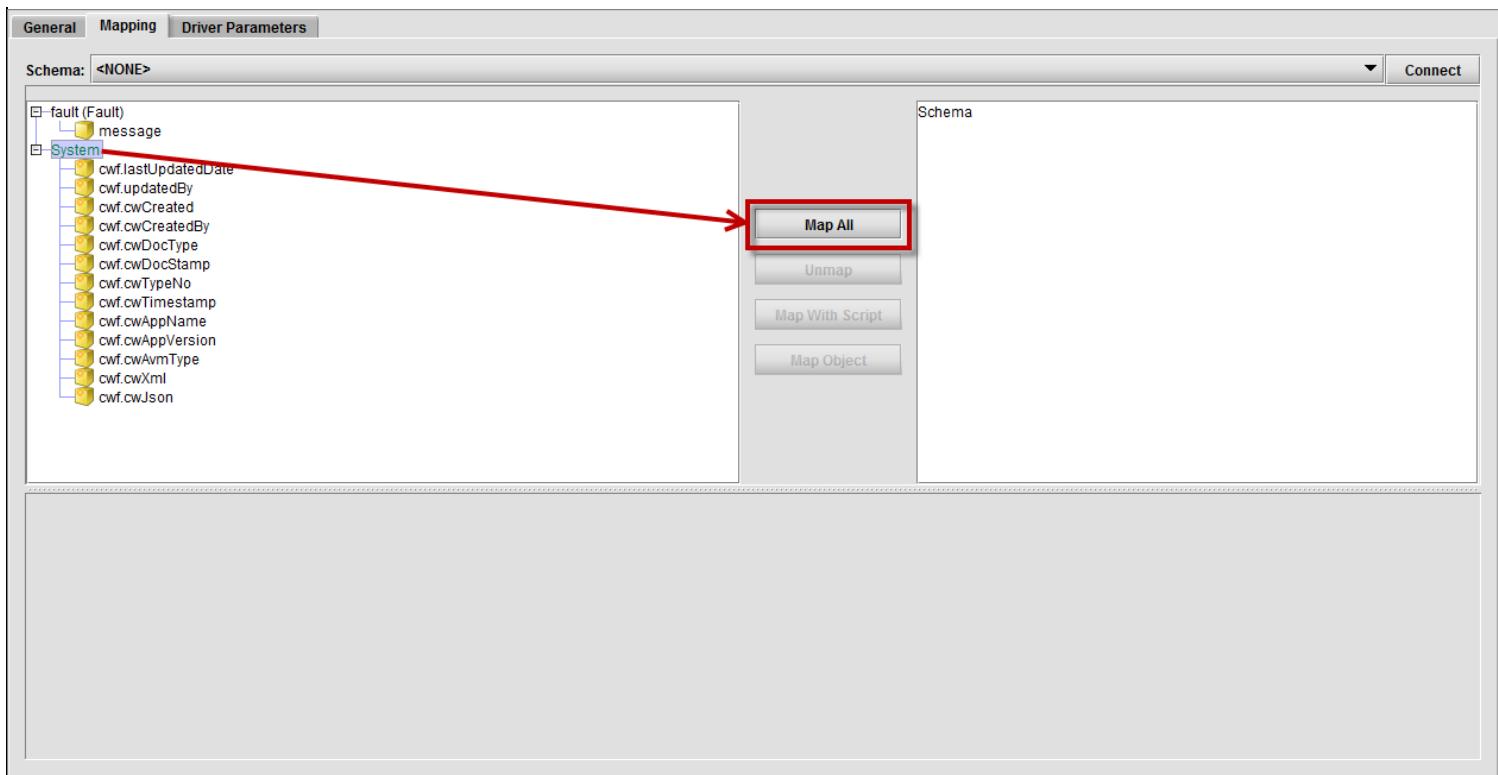


The following table shows system types that you can map with a database column and a description of each system type:

System Type	Description
AVM Type	This system type denotes the object's XSD name (for example, <a href="http://ericsson.com/cpm">http://ericsson.com/cpm</a> ,customer)
Application Name	This system type is the application's internal name.
Application Version	This system type represents the application's internal version.
Created By	This system type indicates the user ID of the person who created the record.
Created Timestamp	This system type denotes the timestamp when record was created.
JSON	This system type represents the data object stored as a JSON string in a column. It stores string data and supports an NCLOB, CLOB, or VARCHAR column with a sufficient size.
Last Updated Timestamp	This system type indicates the timestamp when record was created.
Type Name	This system type denotes the metadata document type name (that is, a qualified name, such as mynamespace.customer).
Type Number	This system type represents the metadata document type number.
Updated By	This system type indicates the user ID of the person who updated the record.
XML	This system type denotes the data object stored as an XML string in a column. It stores string data and supports an NCLOB, CLOB, or VARCHAR column with a sufficient size.

## Map a Single Database Column

You can map individual database columns. In the left pane, select the column that you want, and then click the **Map** button.

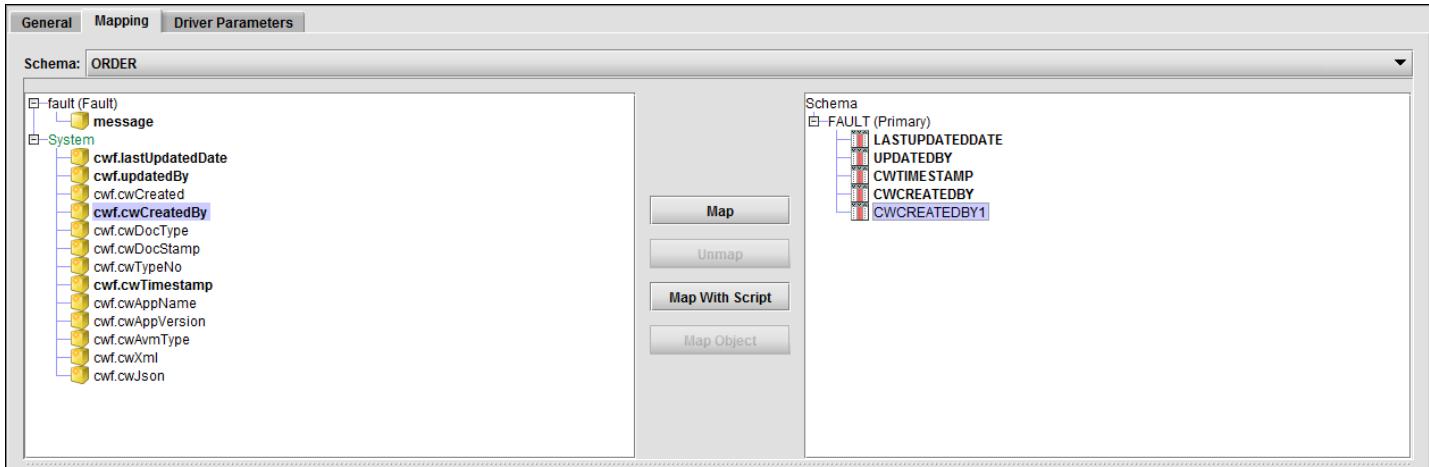


The right side of the page contains the mapped database column.

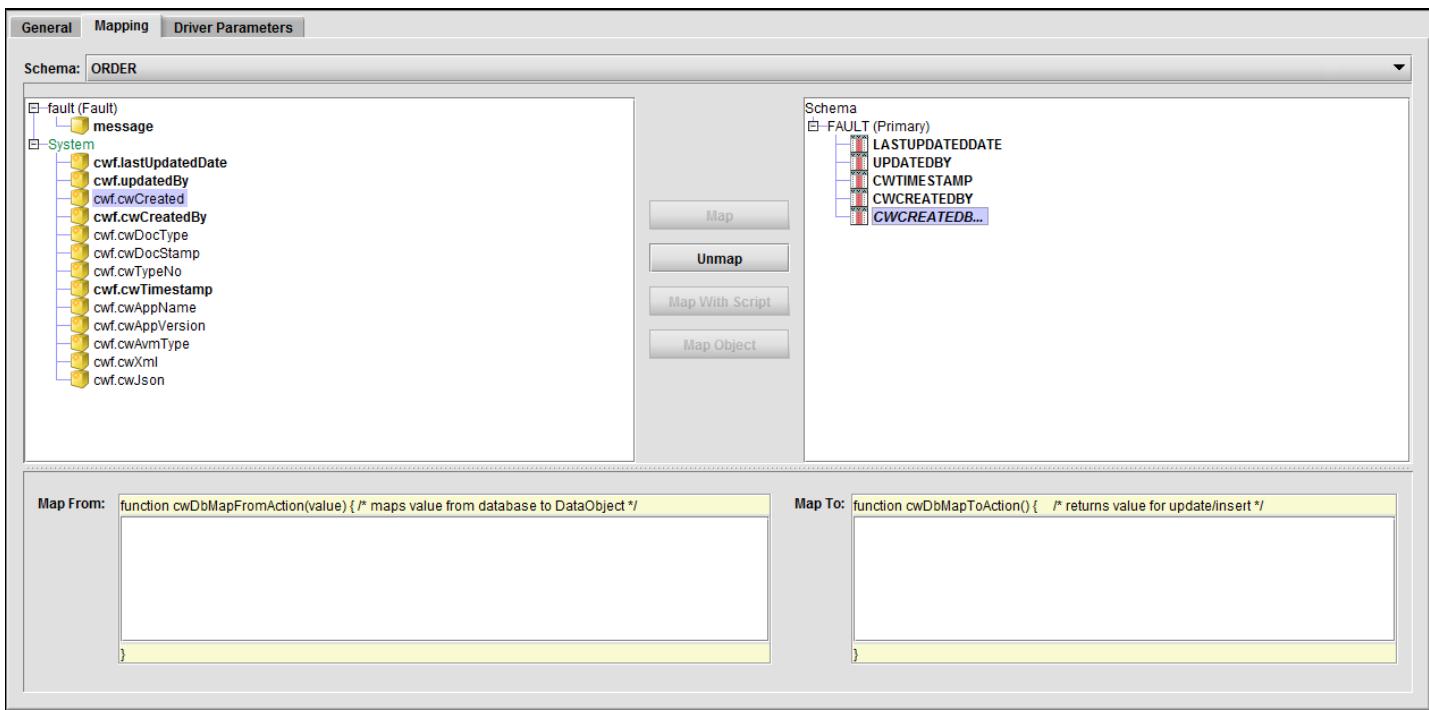
## Map with a Script

You can map using a script by completing these steps:

1. Select the column that you want to map, and then click the **Map With Script** button.



2. To denote whether the column is a key, your data structure must have a key variable defined within it. Refer to the [Data Structures General properties - Keys](#) field for more information.



3. Two separate script functions appear in the area under the **Map with Script** button:

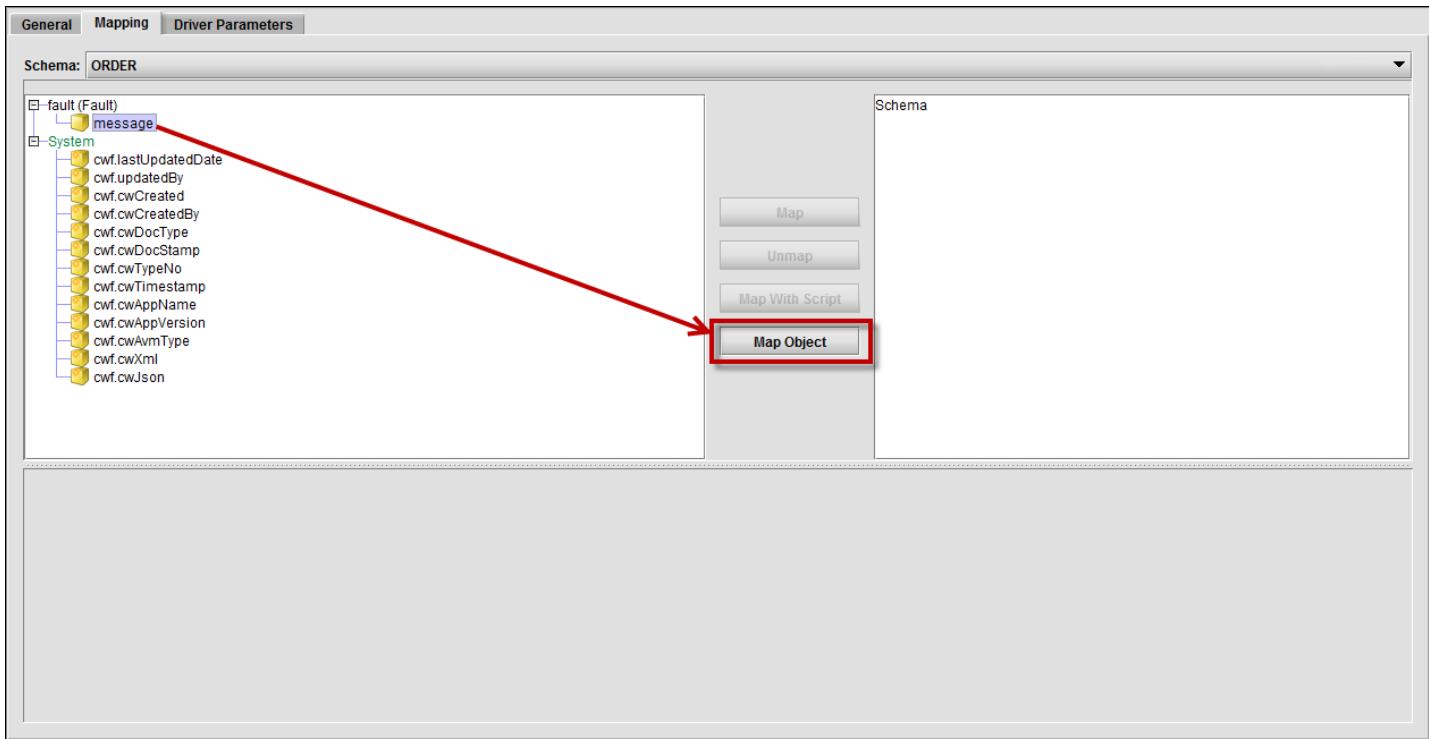
- o cwDBMapFromAction, which runs when data is retrieved from the database
- o cwDBMapToAction, which runs when the data object is saved to the database.

**Note:** It is mandatory to provide an implementation that returns the appropriate values that match the column data type.

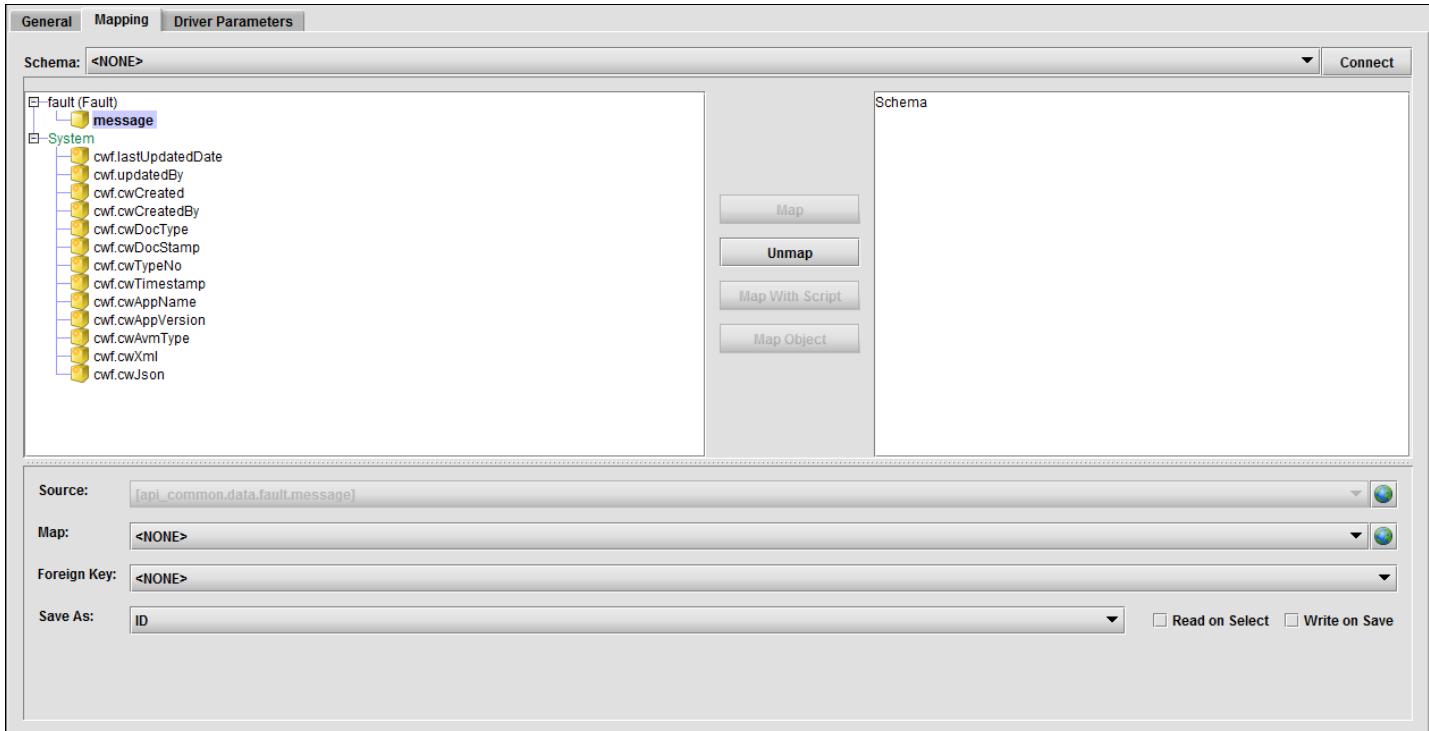
## Map an Object Variable

You can map an object variable as an object reference by completing these steps:

1. Select the object from the left side of the page, and then click the **Map Object** button.



2. A number of fields appear.



Specify the following fields:

Field	Description
<b>Source</b>	This field represents the source object variable that you want to map to your database table.
<b>Map</b>	Click the drop-down menu and select an object mapping from the list. If you do not select a map, the default mapping is used for the object. Only the object ID of the child object is saved to the parent data structure. The cwSetStructId and cwGetStructId methods set and get the object ID, which need to be overridden in the child object data structure.
<b>Foreign Key</b>	An object variable can be mapped as an object reference where the parent object reference, as a foreign key, is stored in the object variable data structure. A key column value of the parent object is stored in the child data object variable in the foreign key column. No column in the parent data structure exists that refers to the child object variable. However, the child object variable has a reference of the parent object. Click the <b>Foreign Key</b> field's drop-down menu and select from the list.
<b>Save As</b>	You can take an object variable and map it with a column by clicking this field's drop-down menu and select from one of the following options: <ul style="list-style-type: none"> <li>o <b>ID</b> When you map an object with this value, the setting saves an ID of the object variable in a column. The object ID is set and get using the cwSetStructId and cwGetStructId methods, respectively, which need to be overridden in the object variable data structure. You can select an object variable mapping from one of the existing mappings. If you do not select a mapping, the default mapping for the object variable is used. You can also specify write and read options. If you specify the <b>Read on Select</b> field for your object map, it reads the object variable with the parent data object. If you specify the <b>Write on Save</b> field for your object map, it writes the object when you save the parent data object.</li> <li>o <b>JSON</b> Setting the field to this value saves the object variable as a JSON serialized string into the column. No mapping is required, as the entire object is stored in a column.</li> <li>o <b>XML</b> Setting the field to this value is the same as <b>JSON</b>, except that this setting stores the XML serialized object as a string.</li> </ul>
<b>Read on Select</b>	Selecting this property allows reading the object with the parent data object.
<b>Write on Save</b>	Selecting this property allows writing the object when you save the parent data object.

**Note:** When a data structure is mapped as a reference key (that is, either a foreign key or an ID mapping), during a save or update operation, the reference key column automatically populates with the data structure object key.

### Expand Data Structure Variables When Mapping an Object Variable

When a data structure has a data structure variable, you can expand the data structure from the Mapping tab, even if it is not marked as a container.

In the following example, **address** is a data structure variable that is not a container in the **person** data structure. In this database map, the **address** data structure variable is expanded to show its variables.

General Mapping Driver Parameters

Schema: ORDER

Connect

person

- personid
- name
- age
- amount
- isPreferred
- orderDate
- total

address

- addressid
- street1
- street2
- city
- province
- zipcode

System

- cwf.lastUpdatedDate
- cwf.updatedBy
- cwf.cwCreated
- cwf.cwCreatedBy

Map

Unmap

Map With Script

Map Object

ORDER

PERSON (Primary)

- PERSONID
- NAME
- AGE
- AMOUNT
- ISPREFERRED
- ORDERDATE
- TOTAL
- ADDRESS
- ADDRESSID
- STREET1
- STREET2
- CITY
- PROVINCE
- ZIPCODE

Source: fcustomerprofile.person.address

Save As: ID

Map: <NONE>

Read on Select    Write on Save

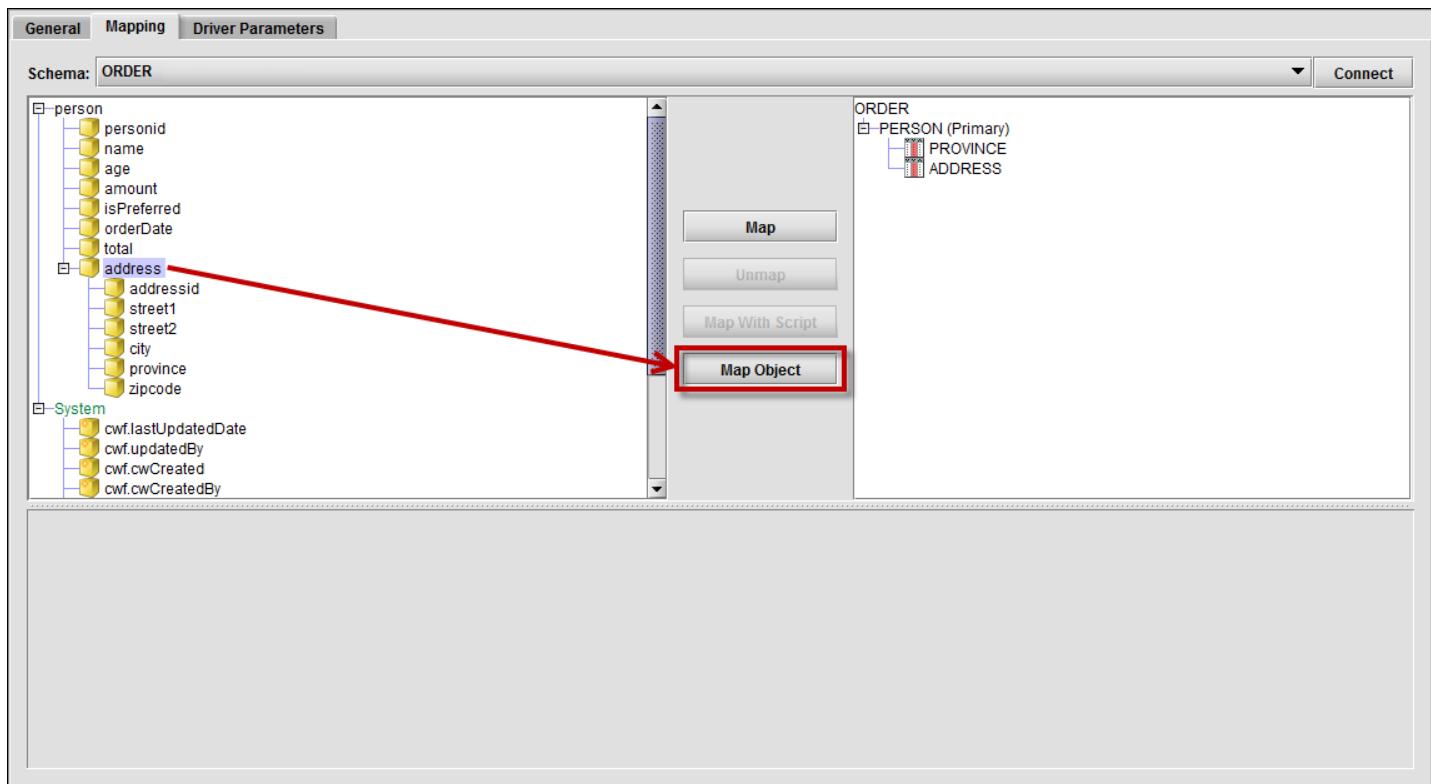
In this example, you can select **address** and provide object mapping using the **Save As** field. You can also choose any of the variables from the **address** data structure (for example, **city**, **province**, and so on) and map it with a column in the **PERSON** table.

## Conversion Map Support for an Object Map with a Foreign Key

There are instances where a data structure has multiple keys defined and it has a variable data structure that is mapped using a foreign key reference. If the foreign key reference uses only one column or variable to store the root data structure information, there is no need for a conversion map. The generated key from the root data structure has a composed key containing all key variable values. However, if the data structure variable has other variables that require the same value as the root data structure variable, you need a conversion map to copy the values.

To use a conversion map for an object map containing a foreign key, complete these steps:

1. Select the object from the left side of the page, and then click the **Map Object** button.



2. A number of fields appear.

General Mapping Driver Parameters

Schema: ORDER

Connect

ORDER

PERSON (Primary)

PROVINCE

ADDRESS

Map

Unmap

Map With Script

Map Object

Source: [customerprofile.person.address]

Map: <NONE>

Foreign Key: <NONE>

Conversion Map: <NONE>

Read on Select  Write on Save

Specify the following fields:

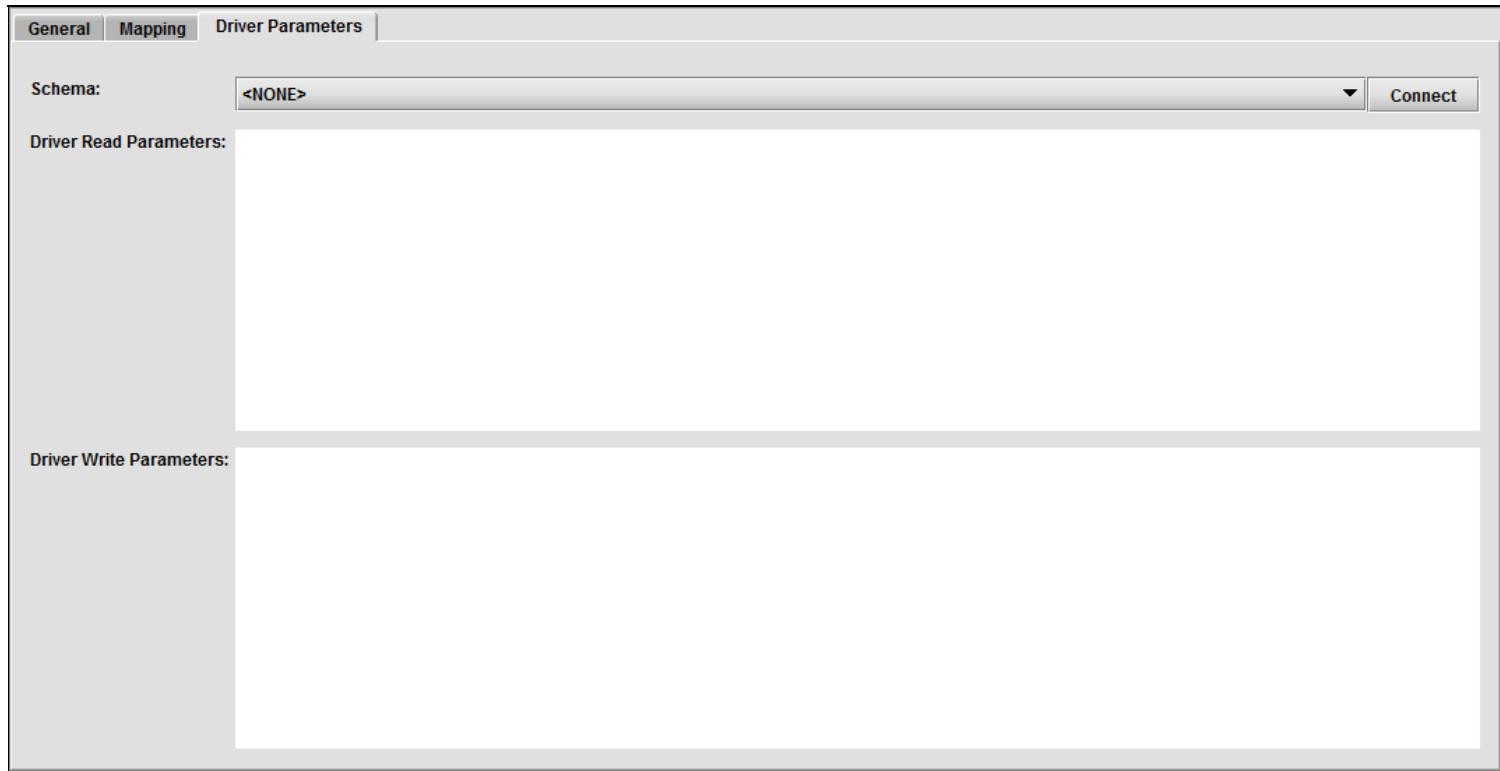
Field	Description
Source	This field represents the source object variable that you want to map to your database table.
Map	Click the drop-down menu and select an object mapping from the list. If you do not select a map, the default mapping is used for the object. Only the object ID of the child object is saved to the parent data structure. The cwSetStructId and cwGetStructId methods set and get the object ID, which need to be overridden in the child object data structure.
Foreign Key	An object variable can be mapped as an object reference where the parent object reference, as a foreign key, is stored in the object variable data structure. A key column value of the parent object is stored in the child data object variable in the foreign key column. No column in the parent data structure exists that refers to the child object variable. However, the child object variable has a reference of the parent object. Click the <b>Foreign Key</b> field's drop-down menu and select from the list.
Conversion Map	Click this field's drop-down menu and select the conversion map that you want. This field displays conversion maps that have the root data structure as its source. Its target is the child data structure with the first item selected.
Read on Select	Selecting this property allows reading the object with the parent data object.
Write on Save	Selecting this property allows writing the object when you save the parent data object.

#### Notes:

- Map conversion takes place during an insert operation only.
- When a data structure is mapped as a reference key (that is, either a foreign key or an ID mapping), during a save or update operation, the reference key column automatically populates with the data structure object key.

## Database Maps Driver Parameters Properties

The database map's Driver Parameters tab allows you to define its schema, and driver read and write parameter properties.



The following table describes the common fields for the database map's Driver Parameters tab:

Field	Description
<b>Schema</b>	This field is of String type and denotes the database schema's name. Click the drop-down menu and select a schema from the list, and then click the <b>Connect</b> button to connect to the database. This field can only be changed when Velocity Studio has a database connection.
<b>Driver Read Parameters</b>	This field represents the JDBC connection properties for read operations, which is shown as a list of name-value pairs for the specified JDBC database driver. The list of property names comes from the database driver, based on the selected schema. Validation occurs at runtime, to ensure that the read parameters are valid for the driver. These parameters can be inherited from the base map.  These parameters are set using a prepare statement before processing.
<b>Driver Write Parameters</b>	This field represents the JDBC connection properties for write operations, which is shown as a list of name-value pairs for the specified JDBC database driver. The list of property names comes from the database driver, based on the selected schema. Validation occurs at runtime, to ensure that the write parameters are valid for the driver. These parameters can be inherited from the base map.  These parameters are set using a prepare statement before processing.

## Audit Mapped Data Structures

---

You can track changes to mapped data structures by selecting an **Audit** checkbox. There are three checkboxes available:

- The global [Enable Audit](#) checkbox in the System Configuration application
- The [Audit Trail](#) checkbox from the database map's General tab
- The database map item's **Audit** checkbox from the database map's Mapping tab

To enable auditing for a specific map item, all three checkboxes must be checked.

### Notes:

- An object map with either an ID or foreign key mapping does not show the **Audit** checkbox. An audit can be enabled on either the default or selected map.
- An object map with either XML or JSON contains the audit feature and it creates an audit record for each variable.
- For event handling, you must create a new event handler for the [SYSTEM\\_AUDIT\\_TRAIL](#) event.

# JavaScript and Query Support

This page describes [JavaScript](#) and [query](#) support for database maps.

## JavaScript Support

The data object supports the following JavaScript functions that work with database maps:

- [insertInDB](#)
- [updateInDB](#)
- [saveToDB](#)
- [readFromDB](#)
- [readFromDBById](#)
- [readFromDBByReference](#)
- [deleteFromDB](#)
- [deleteFromDBByld](#)
- [selectFromDB](#)

See the [JavaScript documentation](#) for details.

### **insertInDB(*destination, connection*)**

This method Inserts the data object in the database using the *destination* parameter.

The *destination* parameter can be one of the following values for the destination schema:

- **Null:** The default database map is used. A error occurs if more than one default map is found.
- **Schema name:** Use the default database map for this schema. An error occurs if no default map is found.
- **Database map name:** Use this database map

The *connection* parameter can be a null or a database connection object. If it is null, the default database connection for the database map is used.

The following is an example of using this method:

```
var customerDs = new mynamespace.customer();
customerDs.name = "myname";
...
...
customerDs.email = "me@mysite.com";

customerDs.insertInDB("ORDER");
```

### **updateInDB(*destination, connection*)**

This method updates the data object in the database. See the [insertInDB](#) method for parameter details.

The following is an example of using this method:

```
var customerDs = new mynamespace.customer();
customerDs.custid = "10016";
customerDs.readFromDB("ORDER");

customerDs.name = "updatedname";

customerDs.updateInDB("ORDER");
```

### **saveToDB(*destination, cascade, connection*)**

This method is a generic one for either the Insert or Update operation. This method calls its respective method based on the object status. If the data object read occurs before it is saved, a call occurs to the updateInDB method. Otherwise, a call occurs to the insertInDB method for the new

data object.

When the *cascade* parameter is set to TRUE, it means that when the database operation is performed, it is also performed for the child data object if the object's **Write on Save** property from the Mapping tab is also set to true. When the *cascade* parameter is FALSE, the database operation for the child object is not performed.

The following is an example of using this method:

```
var customerDs = new mynamespace.customer();
customerDs.custid = "10016";
customerDs.readFromDB("ORDER");

customerDs.name = "updatedname";

// Calls updateInDB, as it read from db before save.
customerDs.saveToDB("ORDER");
```

See the [insertInDB](#) method for parameter details.

#### **readFromDB(*destination, cascade, connection*)**

This method reads the data object from the database. You must set a key column value to read the data object from the database. See the previous code example on using this method.

See the [insertInDB](#) and [saveToDB](#) method for parameter details.

#### **readFromDBById(*ids, name, destination, cascade, connection*)**

This static function reads the data object based on the IDs, name, and destination. It returns a list of data objects based on the selection criteria.

The *ids* parameter is an array of object that specifies the key column values.

The *name* parameter specifies the data object's metadata name. It is the qualified name with its namespace included (for example, mynamespace.customer) . An error occurs if the name is null or does not exist.

The following is an example of using this method:

```
var customerDs = CwObject.readFromDBById(["10016", "10020"], "mynamespace.customer", 'ORDER');
```

See the [insertInDB](#) method for other parameter details.

#### **readFromDBByReference(*values, columns, name, destination, cascade, connection*)**

This static function reads the data object based on the columns and their values, and gets both the data metadata name and destination schema name. This function generates an SQL SELECT statement with all the given column names in WHERE clauses (for example, WHERE col1=? AND col2=?), and passes all parameter values to the statement. An SQL exception occurs if the parameter is missing or there are incorrect column names.

The *values* parameter is an array of objects that contains values for the respective columns listed in the *columns* parameter. The *column* and *values* parameter counts need to match.

The *columns* parameter is an array of objects that contains the column names.

The following is an example of using this method:

```
var custListA = CwObject.readFromDBByReference([ "xyz" ], [ "name" ], "mynamespace.customer",'order'); //single column
var custListB = CwObject.readFromDBByReference([ "xyz", 21], [ "name", "age" ], "mynamespace.customer",'order'); //multiple column
```

See the [insertInDB](#) and [readFromDBById](#) methods for other parameter details.

#### **deleteFromDB(*destination, connection*)**

This method deletes the data object from the database. It returns a deleted record count when successful. You must provide a key column value to delete the data object from the database.

The following is an example of using this method:

```
var customerDs = new mynamespace.customer();
customerDs.id = "10013";
customerDs.deleteFromDB("ORDER", true);
```

See the [insertInDB](#) and [saveToDB](#) methods for parameter details.

#### **deleteFromDBByld(*id, name, destination, connection*)**

This static function deletes the data object based on the IDs, metadata name, and destination schema name. It returns a deleted record count when successful.

The following is an example of using this method:

```
var deletedCount = CwObject.deleteFromDBById(["10016", "10020"], "mynamespace.customer", 'ORDER');
```

See the [insertInDB](#) and [readFromDBByld](#) methods for parameter details.

#### **selectFromDB(*sql, params, name, destination, cascade, connection*)**

This static function reads the data from the database using the provided SQL query. The query can be written with a dynamic parameter that can also pass as the *params* argument. Both the metadata name and destination schema name are used to read the data object from the database.

The *sql* parameter is the SQL query. To get a result similar to `SELECT * FROM CUSTOMER WHERE CWDOCID=10011`, the *sql* parameter needs to be written as `SELECT {+*+} CWDOCID=?`. If there are multiples table mapped with the data structure, the table alias also needs to be specified (that is, `SELECT {+*+} T0.CWDOCID=?`). Internally, when the SQL generates, the actual column names replaces the `{+*+}` part. There is neither a table name, nor a WHERE clause.

If limited columns are only required, instead of using the `{+*+}` expression, column names can be specified. Column names need to match the exact order as the database map. Additionally, columns cannot be skipped.

As an example, if a table has 10 columns (`col01... col10`), and queries only need three columns, the SQL statement, `SELECT col01, col06, col03 FROM XYZ WHERE colum1 = 10016`, can be written as `SELECT col01, col02, col03, col04, col05, col06 FROM XYZ WHERE col01 = ?`, `["10016"]`.

**Note:** Some columns are not listed in the original statement. As columns cannot be skipped, the SQL specified all columns till the last one. In this example, `col06` is last one. As a result, there is no need to specify the remaining columns.

The *params* parameter passes parameters to the given SQL query. If there is no parameter required, this parameter can be null.

**Note:** The `{+*+}` expression always has one parameter. In the previous example, parameter is denoted as `['10016']`.

The following is an example of using this method:

```
var AddressList = CwObject.selectFromDB("SELECT {+*+} ID = ?", ["10016"], "mynamespace.address", 'order');
//single table in mapping

var custList1 = CwObject.selectFromDB("SELECT {+*+} T0.ID = ?", ["10016"], "mynamespace.customer", 'order');
//multiple table in mapping

var custList2 = CwObject.selectFromDB("SELECT col01, col02, col03, col04, col05, col06 FROM XYZ WHERE col01 = ?",
["10016"], "mynamespace.customer", 'order');
```

See the [insertInDB](#) and [readFromDBByld](#) methods for parameter details.

## **Query Support**

Scripts and script finders can use SQL queries. The script finder has various event methods, such as `cwOnFinderSel`, where you can write a script using the `selectFromDB` method to get the data object from the database and return it to the script finder.

## Conversion Maps

The Conversion Map is used to map one Document to another Document or Data Structure or one Data Structure to a Document. A reason why this mapping may be done includes, but is not limited to, sending a set of parameters to an external interface. Conversion Maps are required for definition of the result mapping in the Finder business rules with a result Document (refer to section on Result Map Properties of Finder Rule).

The following types of mapping are supported:

- Document to Document
- Document to Data Structure
- Data Structure to Document
- Data Structure to Data Structure, including a Data Structure with container nodes that contain separate top-level Data Structures

**Note:** The `mapToStructure()` method can be used to map the one data structure to another manually through a script. For example:

```
var targetDs = new DataStructure("cwl:participantOperationSearch");
this.mapToStructure(targetDs, "cwl:participantToOperationForEsc", false);
```

where second parameter `cwl:participantToOperationForEsc` is the name of the conversion map. This parameter can be null, if `this` (current source data structure) is of the same metadata type as `targetDs`, or there is a default conversion map for `this` and `targetDs`.

- An orders as either a source or target
- An order as an input or output to interface operations

**Note:** Deep-nested structures are also supported.

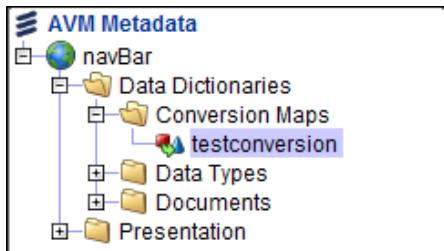
### Creating a Conversion Map

In the **Navigation** pane, **Conversion Maps** are found in the **Data Dictionaries** folder in each namespace.

To create a Conversion Map:

1. Right-click the **Namespace** and select **New**.
2. From the **New Metadata Objects** dialog, expand the **Data Dictionaries** folder and select the **Conversion Maps** and click **Next**.
3. In the new **Conversion Map** dialog, type a name for the conversion map.
4. In the **Source** drop-down select the Source Document or Data Structure from which you want to convert.
5. In the **Target** drop-down select the Document or Data Structure in the mapping which you want to convert.
6. Click **Finish**.

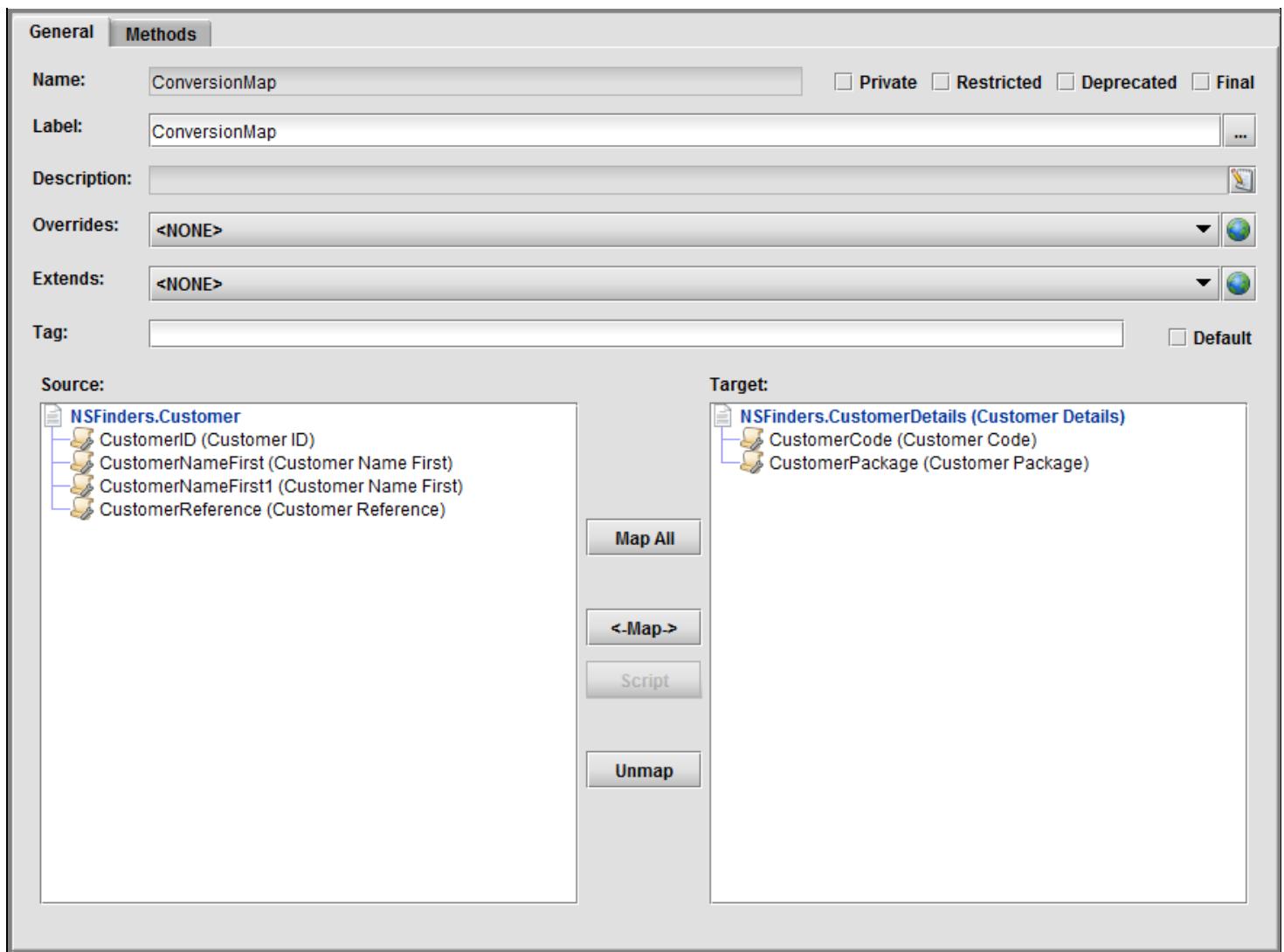
After the Conversion Map is created, the icon will be added under the **Conversion Maps** folder.



Library examples of the Conversion Map are available from the `cwf_pm` (Process Management) node on the **Library** tab.

### General Conversion Map Properties

The general Conversion Map properties are on the **General** tab.



The following table describes the fields:

Fields	Description
<b>Name</b>	Name of the metadata object (used in scripts; must conform to JavaScript naming conventions).
<b>Label</b>	Metadata object display label.
<b>Description</b>	Description of the metadata object for documentation.
<b>Overrides</b>	Metadata object to override. Used to customize system or library metadata objects.
<b>Extends</b>	Shows the base Conversion Map that is extended by this Conversion Map. Will have the base Conversion Map name as the value when the map was created by the <b>Extend...</b> command. Appears for Document-to-Document maps only.
<b>Tag</b>	Common name for a group of Conversion Maps.
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Final</b>	Metadata elements can be marked as final, which includes any Boolean final property that is available on any metadata element. Additionally, any metadata element that has the <b>Final</b> property set to true can neither be extended, nor overridden.
<b>Default</b>	If checked, defines the default Conversion Map for this pair of objects.

A derived Document from a library can be used as a source or target in a Conversion Map.

More than one Conversion Map can be defined between the same pair of objects. If the objects are Documents, one of the Conversion Maps must be defined as a default map by selecting the **Default** check box on the **General** tab. This map will be used by the runtime system when the Document-to-Document Conversion Map is not specified.

Several Conversion Maps can be combined into a group by specifying a common tag name for them in the **Tag** field. The tag name can be used in JavaScript functions instead of the Conversion Map name.

Similar to XSD, the WSDL files are stored as resources once imported into the Velocity Studio, and can be converted to Interface-related metadata objects.

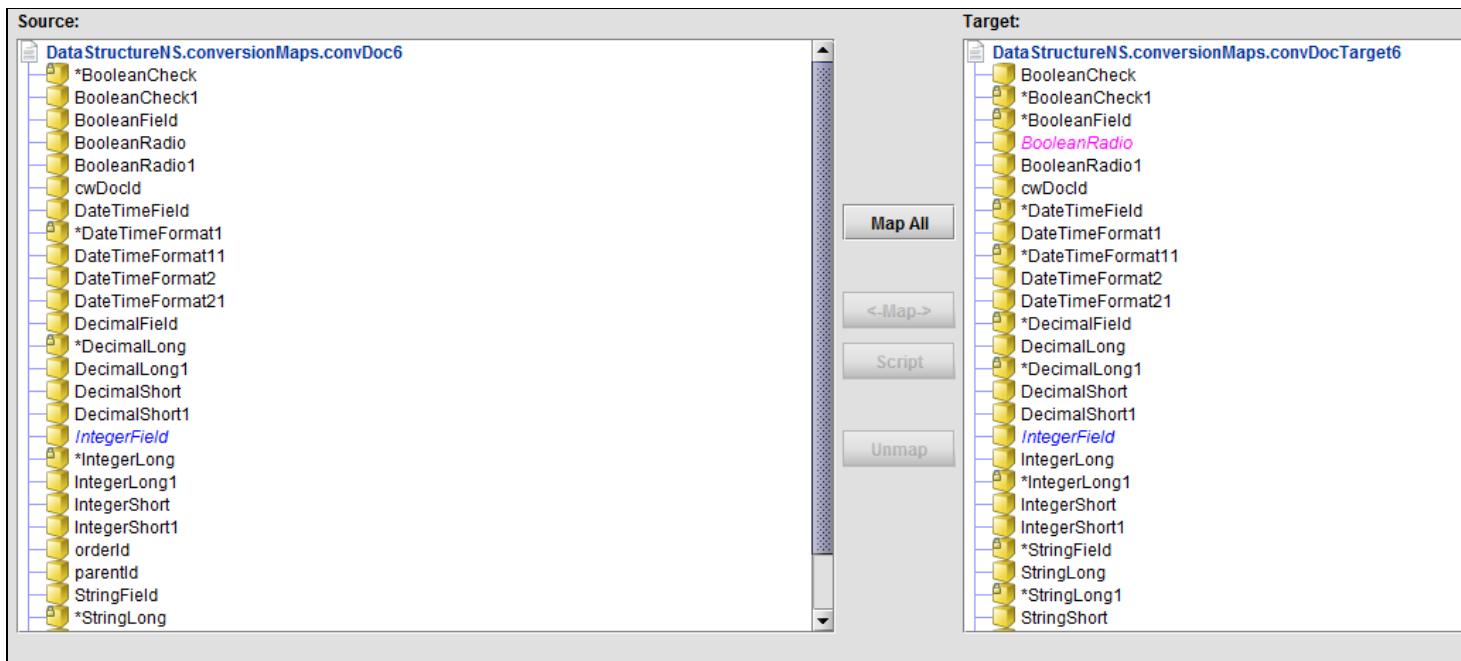
**Note:** The existence of duplicated conversion map tags are caught at design time, instead of at runtime. Any duplicated conversion map tags appear as errors in the Problems tab of Velocity Studio.

## Map Item Properties

The **Source** tree in the left area shows the elements in the source object. The **Target** tree in the right area shows the elements in the target object. Mapping one source element to multiple target elements is allowed, while mapping multiple source elements to a single target element is not allowed.

- To map elements in the source and target objects, highlight the element in both fields and click the **Map** button.
- The **MapAll** button is used to map all elements contained in the **Source** tree to those elements in the **Target** tree, provided the elements have identical names. Otherwise, the elements must be mapped one by one.
- The **Unmap** button is used to remove mapping between elements.

Once the mapping is done, the color of the elements change to blue. When you extend a conversion map, the map editor displays an asterisk (\*) in front of inherited mapped items from the base map.



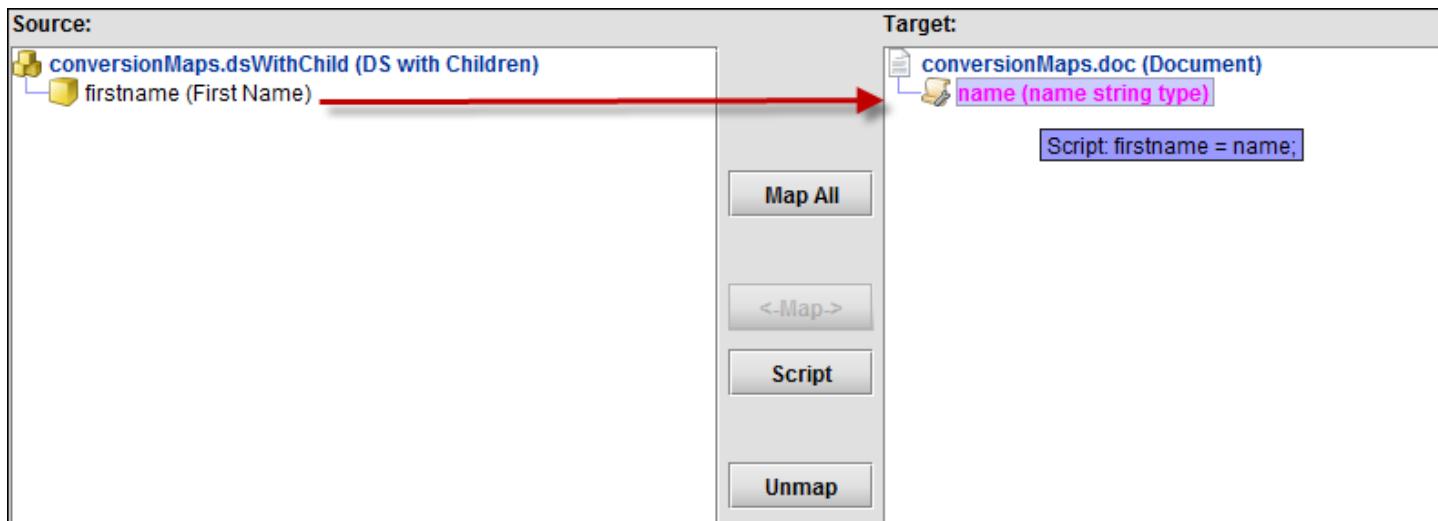
The source and destination elements may have different data types, but the types must be compatible, as shown in the table below (**Note:** l=length, p=precision).

Source/Dest	String	Number	Date	Time	Time stamp	Time Interval	Money	Boolean	Enumeration
<b>String</b>	$l(d) \geq l(s)$	No	No	No	No	No	No	No	No
<b>Number</b>	$l(d) \geq l(s) + 2$	$l(d) \geq l(s) \& p(d) \geq p(s)$	No	No	No	No	No	No	No
<b>Date</b>	$l(d) \geq 10$	No	Yes	No	Yes	No	No	No	No
<b>Time</b>	$l(d) \geq 8$	No	No	Yes	No	Yes	No	No	No
<b>Time stamp</b>	$l(d) \geq 18$	No	Yes	No	Yes	No	No	No	No
<b>Time Interval</b>	$l(d) \geq 10$	No	No	No	No	Yes	No	No	No
<b>Money</b>	$l(d) \geq 13$	$l(d) \geq 12 \& p(d) \geq 4$	No	No	No	No	Yes	No	No
<b>Boolean</b>	$l(d) \geq 1$	No	No	No	No	No	No	Yes	No
<b>Enumeration</b>	$l(d) = l(s)$	No	No	No	No	No	No	No	Same type only.

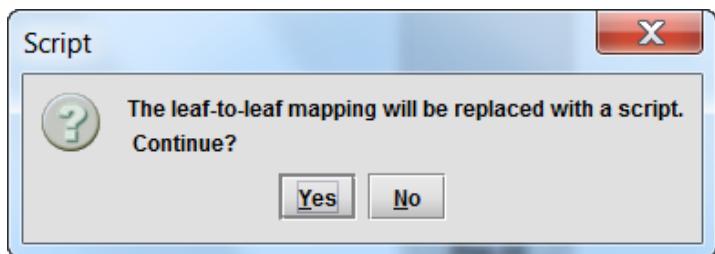
## Script Option

The **Script** option is used to compose a mapping script in JavaScript to cover additional mapping needs. For example, Element 1 added with value 5

needs to be mapped to Element 2. To create, edit, or remove a script for a selected element, the **Script** button must be clicked. The [JavaScript Editor](#) opens where the script can be composed. Click the **OK** button when the script is finished to return to the **General** tab. Elements that has a script attached to it display in magenta colour. The script mapping objects of an overriding conversion map are editable in the script editor.



A variable with a script mapping cannot have a leaf-to-leaf mapping or vice versa. The script is attached to the target leaf and the target leaf value can be calculated based on any or all variables of the source Document. A warning message appears on attempt to replace one type of mapping with another.



## Add Methods

From the Methods tab, you can add a document level mapping script.

**Note:** If a Document level mapping script is specified along with the variable-level mapping for a particular source and target Document pair, it is performed after the variable-level mapping is performed.

## View Properties of Conversion Map Objects

To view the element properties of conversion map objects, right-click a variable or tree node of a source or target object, and select **Properties**.

**AVM Metadata**

- + applicationUI (Application UI)
- + billingClient (Billing Client)
- + billingServer (Billing Server)
- + customerManagement (Customer Management)
- + Extensions
- + orderManagement (Order Management)
  - + Business Processes
  - + Data Dictionaries
    - + Conversion Maps
      - customerProfileToRequestMap
    - + Data Types
    - + Documents
    - + External Services
    - + Finders
    - + Orders

**General**   **Methods**

Name: customerProfileToRequestMap

Label: customerProfileToRequestMap

Description:

Overrides: <NONE>

Tag:

Source:

```
customerManagement.customerProfile (Customer Profile)
  + customerName
  + cwDocId
  + isPreferred
  + marketSegment
```

**Properties**

The element's Properties dialog appears. These properties are read-only.

**Properties**

**General**   **Elements**

Name: fName

Label:

Description:

Container  Data type (String, 15)  Array  Mandatory

Element:

**XML settings:**

XML name:

Format:

Length:

Include  Exclude  Include Always  Attribute

CDATA  Text

**Close**

## Orders

The Order is a major component in any OE application. An Order contains Documents and/or Collections. A Collection is used to hold multiple Document instances so that multiple line items can be stored in the Order, thus a Collection is similar to an array. The objects contained in the Order are called *Order Items* (that is, Documents, Collections, and Collection instances are all Order Items).

When defining an Order, you must determine:

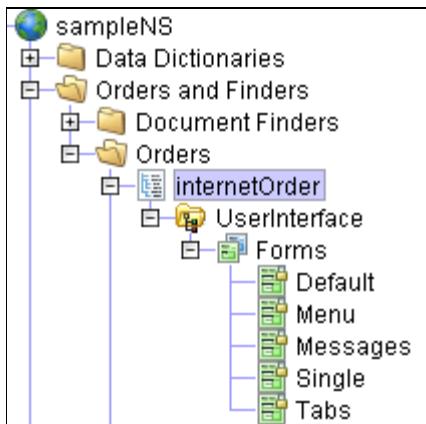
- Which Order Documents the Order must contain.
- Whether any Order Documents require multiple instances, in which case a Collection must be created under the Order.
- Whether the ordering process needs to be controlled by a Life Cycle.

### Creating an Order

To create a new Order, do one of the following:

- In the **Metadata** tab of **Navigation** pane, either:
  1. Right-click a Namespace and select **New ...**
  2. **New Metadata Object** wizard appears as a popup. Expand **Orders**, select **Order**, and then click the **Next** button.
  3. Step two of the wizard appears. Type in the name of Order (must conform to JavaScript naming conventions). For **Base User Interface**, choose **com.conceptwave.system.OrderUserInterface** to create a new Order. Then, click the **Finish** button.
- OR
  1. Right-click the **Orders** folder or the **Orders** folder in a Namespace, if it is present. Select **New Order**.
  2. Follow step 3 from above.

The newly created Order is added under **Orders** folder. Beneath the Order node is the Order's User Interface node, which subsequently contains the Forms of the Order.



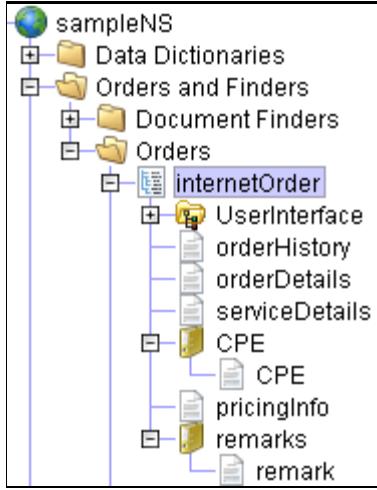
**Note:** Velocity Studio features require a separate license key that is activated in Velocity Studio.

### Create New Order Item

With an existing Order, the Order hierarchy can be built via creating Order Items, by right-clicking the Order and selecting one of the commands described below.

- **New Document**: Adds an Order Document as a child to this node.
- **New Collection**: Adds a Collection as a child to this node.

After selecting the pop-up menu command, **New Metadata Object** wizard appears as pop-up. Type in the name of the Order Item (must conform to JavaScript naming conventions), and populate other fields as necessary. Then click **Finish**. The newly created Order Item is added under the Item that is right-clicked. If this child node is a Collection, it can again be right-clicked on to further define the Order hierarchy. The figure below shows an Order containing multiple Collections as well as Documents.



### Extended Order Item Documents

When an Order Document is added under a collection, this node accepts a document of the declared document type or any extended type. At runtime it is possible to add a document of an extended type, by calling the `addDocumentInstance` method. From the example above, the "CPE" order collection will allow any extended document of CPE order item. The `addDocumentInstance` method is called to allow extended documents at runtime:

```

var extendedDoc = new Document("SampleNS:extendedCPE");
order.CPE.addDocumentInstance(extendDoc);

```

## Order General Properties

The general Order properties are defined on the **General** tab. The following image shows the General tab of the top-level Order.

The screenshot shows the 'General' tab of the Order properties dialog. The tabs at the top are 'General' (selected), 'Order items', and 'Methods'. The 'General' tab contains the following fields:

- Name:** phoneOrder  Private  Restricted  Deprecated  Virtual  Final
- Label:** Phone Order
- Description:**
- States:** <NONE>
- Instance key:** <NONE>
- Attachments**
- XSL:FO file:**
- Sync mode:** Default
- XML settings:**
  - XML name:**
  - Include namespace**

The following table describes the fields for Order; fields available to top-level Order metadata, Order Collection and Order Document are slightly different as indicated.

Fields	Root-Level Order	Order Collection	Order Document	Description
<b>Name</b>	Yes	Yes	Yes	Name of the metadata object (used in scripts; must conform to JavaScript naming conventions). Read-only after creation. To change it, use the <b>Rename</b> command in pop-up menu by right-clicking the metadata object.
<b>Private</b>	Yes	No	No	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	Yes	No	No	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	Yes	No	No	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Virtual</b>	Yes	No	No	If checked, it provides the ability to create methods in template metadata that does not have implementation, but can be used in template script and be bound to UI elements. The implementation is done in user metadata (see <a href="#">Virtual Field</a> for more information).
<b>Final</b>	Yes	No	No	Metadata elements can be marked as final, which includes any Boolean final property that is available on any metadata element. Additionally, any metadata element that has the <b>Final</b> property set to true can neither be extended, nor overridden.
<b>Label</b>	Yes	Yes	Yes	Metadata object display label.
<b>Description</b>	Yes	Yes	Yes	Description of the metadata object for documentation.
<b>States</b>	Yes	No	No	If the Order needs to be associated with a state during processing, a Life Cycle must be selected. The Life Cycle defines the Order states and state transitions. It must be defined in advance to be seen in this field.

<b>Instance key</b>	Yes	Yes	No	The selected child node to show as a root node of the Order or Collection instance. If selected, the child visual key will be displayed at runtime as the visual key of the selection instead of the root/Collection visual key. The visual key of this item should uniquely identify the Order or Collection instance.										
<b>Order item</b>	No	No	Yes	Mandatory. The Document that is referenced by this node.										
<b>Minimum, Maximum</b>	No	Yes	No	The <b>Minimum</b> and <b>Maximum</b> values represent the range of Collection instances that can be added under a Collection at runtime. Note: there is no limit to the <b>Maximum</b> number. However, if the <b>Minimum</b> number is 0, the Collection folder will not be seen in the OE client UI when the Order is activated. To add an Order Document instance into such a Collection, highlight its parent element and click the <b>Add</b> button in the OE client UI.										
<b>Mandatory</b>	No	Yes	Yes	Indicates that this Collection/Document is mandatory at runtime. If checked for Collection, at least one instance will appear when the Order is being created, even if the <b>Minimum</b> instance is set to zero if item is a Collection. Otherwise, if unchecked, the Collection instance folder's appearance will depend on the <b>Minimum</b> instance number. For example, if the <b>Minimum</b> instance is zero, the folder will not be seen in the UI when the Order is activated.										
<b>Attachments</b>	Yes	No	No	If checked, the attach icon is enabled on the OE client UI, allowing users to attach a file to the Order at runtime.										
<b>XSL:FO file</b>	Yes	No	No	Specifies the url for the XSL:FO file to be used for Order Printing, as defined in the Configuration application under Resources. <b>Note:</b> See Configuration User Guide, Web Resources for more information on defining resources. <b>Note:</b> See Velocity Studio User Guide, Order Printing 18 for more information on PDF printing.										
<b>Sync mode</b>	Yes	No	No	Defines a synchronization mode that is used for handling the simultaneous user access conflicts. See <a href="#">Metadata Settings</a> and <a href="#">Handling Simultaneous User Access Conflicts</a> for more information.										
<b>XML Settings</b>	Yes	Yes	Yes	<p>Specifies how to export/import orders to/from XML format. By default, all Order elements are exported/imported.</p> <table border="1"> <tr> <td><b>XML Name</b></td><td>XML tag name for this element. Default value is the element name.</td></tr> <tr> <td><b>Instance name</b></td><td>Visible for Collection items only. This is the XML tag name for the instance of this Collection. The default value is the Collection name. This field allows different XML tags for the Collection as a whole and its instances. If not specified, the Collection and the instances will have the same XML tag name.</td></tr> <tr> <td><b>Include / Exclude</b></td><td>When <b>Exclude</b> is selected for any Order Item, it excludes the element XML tag from the processing (that is, it does not generate the tag when writing XML and does not look for it when reading XML). However, all children of the Order Item with XML tags should exist.</td></tr> <tr> <td><b>Include / Exclude / Exclude children</b></td><td><b>Exclude children option</b> is available at Collection items only. When checked, this item, and all of its children, will be excluded when generating the XML or not processed when reading the Order from XML.</td></tr> <tr> <td><b>Include namespace</b></td><td>Visible for an Order root only. When checked, the <i>xmlns</i> attribute will be generated with the namespace value of the namespace this Order belongs to.</td></tr> </table>	<b>XML Name</b>	XML tag name for this element. Default value is the element name.	<b>Instance name</b>	Visible for Collection items only. This is the XML tag name for the instance of this Collection. The default value is the Collection name. This field allows different XML tags for the Collection as a whole and its instances. If not specified, the Collection and the instances will have the same XML tag name.	<b>Include / Exclude</b>	When <b>Exclude</b> is selected for any Order Item, it excludes the element XML tag from the processing (that is, it does not generate the tag when writing XML and does not look for it when reading XML). However, all children of the Order Item with XML tags should exist.	<b>Include / Exclude / Exclude children</b>	<b>Exclude children option</b> is available at Collection items only. When checked, this item, and all of its children, will be excluded when generating the XML or not processed when reading the Order from XML.	<b>Include namespace</b>	Visible for an Order root only. When checked, the <i>xmlns</i> attribute will be generated with the namespace value of the namespace this Order belongs to.
<b>XML Name</b>	XML tag name for this element. Default value is the element name.													
<b>Instance name</b>	Visible for Collection items only. This is the XML tag name for the instance of this Collection. The default value is the Collection name. This field allows different XML tags for the Collection as a whole and its instances. If not specified, the Collection and the instances will have the same XML tag name.													
<b>Include / Exclude</b>	When <b>Exclude</b> is selected for any Order Item, it excludes the element XML tag from the processing (that is, it does not generate the tag when writing XML and does not look for it when reading XML). However, all children of the Order Item with XML tags should exist.													
<b>Include / Exclude / Exclude children</b>	<b>Exclude children option</b> is available at Collection items only. When checked, this item, and all of its children, will be excluded when generating the XML or not processed when reading the Order from XML.													
<b>Include namespace</b>	Visible for an Order root only. When checked, the <i>xmlns</i> attribute will be generated with the namespace value of the namespace this Order belongs to.													

## Order Items

The **Order Items** tab contains the list of all the immediate children of the current Order Item, in the order in which they will be displayed in UI.

The arrow buttons located at the side of the list are used to move the selected element in the list to the desired location. The **Sort** button is used to sort the elements in alphabetical order.

A screenshot of a software interface showing the 'Order Items' tab selected. The tab bar also includes 'General' and 'Methods'. The main area displays a list of items under the namespace 'sampleNS.internetOrder':

- CPE
- orderDetails
- orderHistory
- pricingInfo
- remarks
- serviceDetails

The item 'serviceDetails' is highlighted with a blue selection bar. To the right of the list are two small arrow buttons: an upward-pointing arrow at the top and a downward-pointing arrow below it. At the bottom left of the list area is a 'Sort' button.

## Order Methods

The **Methods** tab is used to add Order-level scripts. There are different types of Methods that can be defined for the Order; they are all consolidated into this tab.

The screenshot shows the 'Methods' tab selected in a software interface. On the left, there's a tree view showing a folder named 'testOrder' containing several methods: 'addPerm', 'attachPerm', 'deletePerm', 'editablePerm', 'visiblePerm', and 'orderInt'. The 'orderInt' method is currently selected. The right pane contains fields for defining the method:

- Name:** orderInt
- Description:** (empty)
- Action Category:** (empty)
- Parameters:** A table with columns 'Name', 'Type', and 'Description'. It currently has one row with an empty 'Name' field and a blue upward arrow icon.
- Script:** A code editor containing the script: 

```
function orderInt() {  
}  
}
```
- Return:** com.conceptwave.system.Void
- Return Description:** (empty)

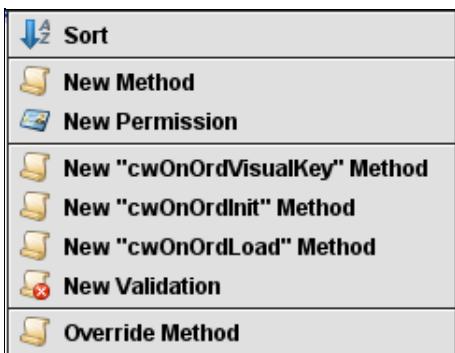
At the top right of the right pane, there are several small icons: a magnifying glass, a pencil, and four arrows (up, down, left, right) for sorting or filtering.

When you select the **Type** radio button, it means that the method can be invoked directly through the script and does not require an instance object.

To show all base methods, click the **Show base methods** (  ) button. Click the button again to hide all base methods. Base methods appear in blue font. Any method highlighted in blue indicates that you can override it. Methods in black font cannot be overridden.

If you have a long list of methods in the left pane, you can press the first letter of the method you are looking for to jump to the list of methods beginning with that letter. In this example, pressing the **D** key highlights the first instance of a method that begins with the letter D. The method displays in the right pane.

The Methods tab is a [script management screen](#) in Velocity Studio, which is composed of the **Method List pane** at the left which lists all scripts of the Order, and **Details pane** which displays the details of a script when selected at the Method List pane. To create a new Method, or to override an existing base Method, right-click the Order in the Method List pane. A pop-up menu appears.



The table below lists the type of Methods that can be added at the Order level.

#### New Methods

Method Type	Description
Script	Generic user-defined JavaScript function that can be defined in an Order and explicitly invoked by other scripts; the function is not triggered by other built-in means in the Order.
Permission	Permission Method that can be associated to determine permission for a certain property of the Order. (for example, Element properties). See section <a href="#">Permissions</a> for details.
Validation	Order validation script that determines the validity of runtime data values of the Order. Similar to <a href="#">Document Validation</a> methods, there are three different types of rules: information, warning and error. The user will be presented with a message indicating why the rule has fired. Validation rules are fired in two situations: <ul style="list-style-type: none"> <li>When the Order (the top level icon) is selected in the UI.</li> <li>When an Order becomes valid as the result of changes to the content in the UI.</li> </ul>

#### New System pre-defined Methods

Method Type	Script Name	Parameters	Return Type	Description
Visual Key	<code>cwOnOrdVisualKey</code>	None.	<code>String</code>	Script that shall return a string representing the visual key. The length of the string shouldn't be longer than 64 characters. If longer the value is truncated. If this Method is empty, the Order Label is used as visual key.
Initialization	<code>cwOnOrdInit</code>	None.	None.	Order Initialization script to set initial values for the Order. It is triggered when the Order is created.
Load	<code>cwOnOrdLoad</code>	None.	None.	Order Load script that is executed once after the Order is loaded from the database.

#### System-defined Methods that can be overridden

Method Type	Script Name	Parameters	Return Type	Description
View	<code>visiblePerm</code>	None.	<code>Boolean</code>	View Permission that specifies whether the participant is allowed to view the Order.

Permission				
Add Permission	<i>addPerm</i>	None.	<i>Boolean</i>	Add Permission that specifies whether the participant is allowed to create an instance of this Order.
Delete Permission	<i>deletePerm</i>	None.	<i>Boolean</i>	Delete Permission that specifies whether the participant is allowed to delete an instance of this Order.
Update Permission	<i>editablePerm</i>	None.	<i>Boolean</i>	Update Permission that specifies whether the participant is allowed to update the Order (for example, change Document Variables in the Order).  <b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.
Attach Permission	<i>attachPerm</i>	None.	<i>Boolean</i>	Attach Permission that specifies whether attachments are allowed to be attached to the Order.

For Method Types that can be created, you can create multiple Methods of the same Method type (for example, Validation 1, Validation 2...), each identifiable by unique Method name. However, for Methods that are to be overridden, you can only define one Method per base Method.

The system-defined Order-level Permission Set above (permView, permAdd, permDelete, permUpdate, permAttach), in essence, govern the Document's Permission. That is, whenever the Document is acted upon (create, read, update, delete...) by a participant, the corresponding Permission Method is invoked by AVM to determine whether the operation is allowed or not. Conversely, the "new" Permission Methods that can be added are not invoked unless explicitly associated with a Permission-related Property (for example, Element Properties in Document Variables).

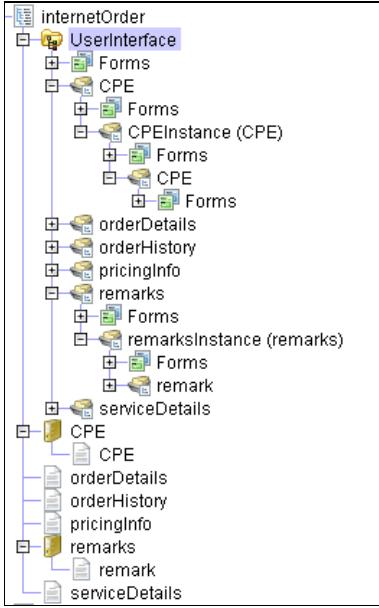
The Method created in Method List pane can be right-clicked, with various commands available; see [script management UI](#) for details.

## Order User Interface and Forms

Order Forms cohere to the User Interface generalization among metadata objects; that is, the Order contains a [User Interface](#), where [Forms](#) are contained within the User Interface. See [User Interface Modelling](#) on how you should use the User Interface object as *Controller* to connect the Order as *Model* to present Order Forms as *View* to users, adhering to the MVC architectural pattern.

However, because Order is a metadata object that is hierarchical, the Order User Interface is essentially a [Navigation Tree](#) metadata object, which is a derived User Interface object. Specifically, the system-base Order User Interface `com.conceptwave.system.OrderUserInterface` is extended from Navigation Tree (with Tabs).

The Order User Interface node sits beneath the root-level Order, and with its nature in Navigation Tree, the root Order User Interface node may have its children Tree nodes.



Thus, presentation of the Order is heavily based on the mechanics of the Navigation Tree metadata and its corresponding [runtime behaviour](#). You are expected to be familiar with Navigation Tree, or you should closely consult its documentation; this article only addresses the additional properties of Order User Interface and Form from their equivalent in Navigation Tree.

By right-clicking the root Order node or the root Order User Interface node, you may command the **Sync Navigation Tree**. The command creates the User Interface nodes hierarchy matching the order model structure. For example, when adding a new document to an order, the Order User Interface must be synchronized with the order model to include the new document's User Interface. Otherwise, the new document is not included in the Order User Interface. In the previous diagram, the Order User Interface is a Navigation Tree with nodes that mirror the order's structure.

Note that, as discussed in [Navigation Tree](#) documentation, the Order User Interface does not necessarily have to present the Order to user as defined by the structure of the Order model. While using the **Sync Navigation Tree** command provides the mirror structure of the Order model, you may add or delete any nodes in the Order User Interface tree, or alter properties in nodes, to present Order that may be more intuitive.

## Order User Interface

Extended from Navigation Tree metadata, Order User Interface has the [General](#), [Variables](#), [Tree Items](#) and [Methods](#) tabs; please see the respective section for details.

Specific to Order User Interface, however, is the additional User Interface Variables and Methods defined on top of its base User Interface object `com.conceptwave.system.NavigationTreeWithTabs`.

### Additional properties in General tab of root-level Order User Interface

There are no additional properties compared to the Root Node of Navigation Tree with Tabs.

### Pre-defined Variables of Order User Interface

Variable	Type	Description
<b>hoverMessages</b>	User Interface	An array of User Interfaces for presenting hover messages, which shows the Validation Errors.
<b>confirmTab</b>	String	Stores the selected tab for <i>continue</i> confirmation dialog.
<b>confirmSelection</b>	Tree Node	Array of Tree Nodes to store selected tree nodes for <i>continue</i> confirmation dialog.
<b>confirmMethod</b>	String	Stores the method to be invoked on <i>OK</i> of the confirmation dialog.
<b>validationErrorList</b>	Object	An array that stores the list of validation errors.

You are strongly encouraged to read [Scripting](#) on how to use User Interface Variables.

### Pre-defined Methods of Order User Interface

Most methods implement functionality on action Menu items such as **Add**, **Attach**, **Copy**, **Delete**, **Validate**, **Tasks**, **Task History**, etc.

**Note:** When library methods are overridden, it is recommended that the *super method* is invoked first and a custom script is written after the super call. If the super method is not invoked the Finder may not work as it is expected.

Method	Return	Description

<b>addAction</b>	<i>com.conceptwave.system.Void</i>	<p>Invoked when user clicks the Add menu. The top-level UI, ui_common.selectOptionalItemUI, is used and is displayed in the dialog.</p> <pre>function addAction(dialog) { // Metadata type method. Can be called by scripts.  var selNode = this.getSingleSelection(); if (selNode == null) return null; var addGroup = this.getAddGroup(selNode.model); if (addGroup == null    !(addGroup instanceof OrderGroup)    !addGroup.isAllowed("Add")) return null; if (addGroup.hasOptionalItems()){ var selectOptionalUI; var uiVers = this.getCompatibleVersion(); if (uiVers == 0) return null; else if (uiVers &gt;= 6) { if (CwMetadataObject.getMetadataObject("ui_common.selectOptionalItemUI") == null) return null; // UI-Common template is not loaded if (dialog) dialog.dialogWidth = 520; selectOptionalUI = new com.conceptwave.system.UserInterface("ui_common.selectOptionalItemUI",addGroup,this); return selectOptionalUI; } else { var doc = new Document("cwf_oe:OptionalItemSelect"); doc.parent = addGroup; selectOptionalUI = new doc.UserInterface(doc, this); } selectOptionalUI.owner = this; return selectOptionalUI; } else if (addGroup.isCollection) { var newItem = addGroup.newInstance(); if (newItem != null) { if (newItem.isSingleDocCollectionInstance &amp;&amp; newItem[0] != null) { newItem = newItem[0]; // select first document of the collection instance } var nodeToSelect = selNode.findNodeByModel(newItem); if (nodeToSelect != null) { this.displayNode(nodeToSelect); } } } } return null; }</pre>
<b>addTab</b>	<i>com.conceptwave.system.Void</i>	<p>Adds the <i>node</i> as a tab at the end of the <i>tabsArray</i>.</p> <pre>function addTab(tabsArray,node) { // Metadata type method. Can be called by scripts.  var tabUI = node.onNodeTab(); if (tabUI != null) { tabUI.parent = this; tabsArray[tabsArray.length] = tabUI; }</pre>
<b>addVisible</b>	<i>com.conceptwave.system.Boolean</i>	<pre>function addVisible(\$psCondition) { // Runs every time the permission is evaluated var permissionObject = this;  if (!\$psCondition) return false; var selNode = this.getSingleSelection(); return selNode != null &amp;&amp; this.getAddGroup(selNode.model) != null; }</pre>
<b>applySearchMap</b>	<i>com.conceptwave.system.Void</i>	<p>This function applies the assigned <b>Search Conversion Map</b> in General tab to populate the search Input Document value. Only applicable for Order Child node.</p> <pre>function applySearchMap() { // Metadata type method. Can be called by scripts.  var searchMap = this.metadata.getSearchConversionMapName(); if (searchMap != null &amp;&amp; this.parentNode != null &amp;&amp; this.parentNode.model != null &amp;&amp; this.model != null) { this.metadata.applyConversionMap(this.parentNode.model, this.model, searchMap); }</pre>
<b>applyTableDocMap</b>	<i>com.conceptwave.system.Void</i>	<p>This function applies the assigned <b>Table Conversion Map</b> in General tab to translate the Tree Node model to the Table Document.</p> <pre>function applyTableDocMap() { // Metadata type method. Can be called by scripts.  var tableDocMap = this.metadata.getTableDocConversionMapName(); if (this.model != null &amp;&amp; this.tableDoc != null &amp;&amp; tableDocMap != null) { var nodeID = this.tableDoc.id; var parentNodeID = this.tableDoc.parentId; this.metadata.applyConversionMap(this.model, this.tableDoc, tableDocMap); // restore IDs: node ID and parent node ID cannot be changed }</pre>

		<pre> if (nodeID != null) {     this.tableDoc.id = nodeID; } this.tableDoc.parentId = parentNodeID; } </pre>
<b>attachAction</b>	<i>com.conceptwave.system.Void</i>	<p>This function publishes a UI_GET_ATTACHMENT_FINDER event, which passes the order item as the event parameter, and then displays the event result in a popup.</p> <p>The default event handler located in UI Common returns an instance of ui_common.attachmentFinder.</p> <pre> function attachAction(dialog){      var selNode = this.getSingleSelection();     if (selNode == null    !(selNode.model instanceof OrderItem)    !selNode.model.attachments    !selNode.model.isAllowed("Attach")) {         return null;     }     if (dialog != null) {         dialog.title = Global.translateText("UU0008", null, [selNode.model.visualKey]);     }     var vers = this.getCompatibleVersion();     if (vers != 0 &amp;&amp; vers &lt; 6) {         // display attachment finder         var attachmentFinder = new Finder("cwf_oe:AttachmentFinder");         var finderUI = new attachmentFinder.UserInterface(attachmentFinder, this);         finderUI.ownerItem = selNode.model;         finderUI.initSearchDoc();         finderUI.searchAction();         return finderUI;     }     else         return publishEvent("UI_GET_ATTACHMENT_FINDER", FIRST_ONE, null, [selNode.model,this]); } </pre>
<b>attachVisible</b>	<i>com.conceptwave.system.Boolean</i>	<pre> function attachVisible(\$psCondition) { // Runs every time the permission is evaluated var permissionObject = this;  if (!\$psCondition) return false; var selNode = this.getSingleSelection(); if (selNode != null &amp;&amp; selNode.isOrderItem()) { return selNode.model.attachments &amp;&amp; selNode.model.isAllowed("Attach"); } return false; } </pre>
<b>confirmTabSelection</b>	<i>com.conceptwave.system.Void</i>	<pre> function confirmTabSelection() { // Metadata type method. Can be called by scripts.  if (this.confirmSelection != null) {     this.treeSel = this.confirmSelection;     this.confirmSelection = null;     this.showDetailAction(); } } </pre>
<b>confirmTreeSelection</b>	<i>com.conceptwave.system.Void</i>	<pre> function confirmTreeSelection() { // Metadata type method. Can be called by scripts.  if (this.confirmSelection != null) {     this.treeSel = this.confirmSelection;     this.confirmSelection = null;     this.showDetailAction(); } } </pre>
<b>continueConfirmation</b>	<i>com.conceptwave.system.Void</i>	<pre> function continueConfirmation() { // Metadata type method. Can be called by scripts.  if (this.confirmMethod != null) {     this[this.confirmMethod](); } return null; } </pre>
<b>continueConfirmation_message</b>	<i>com.conceptwave.system.String</i>	<pre> function continueConfirmation_message() { // Metadata type method. Can be called by scripts. return "UU0256:YN"; } </pre>
<b>copyAction</b>	<i>com.conceptwave.system.Void</i>	<p>Order's with editable tables calls the <i>searchAction()</i> method in cases when this Order is currently no displaying any data. A table element is notified to open the new row in edit mode.</p> <p>function copyAction() { // Metadata type method. Can be called by scripts.</p> <pre> var selNode = this.getSingleSelection(); if (selNode == null) return null; </pre>

		<pre> var copyItem = this.getCopyItem(selNode.model); if (copyItem != null &amp;&amp; this.getAddGroup(copyItem.parent) != null) {     var newItem = copyItem.copySubtree(null, true);     if (newItem != null) {         if (newItem.isSingleDocCollectionInstance &amp;&amp; newItem[0] != null) {             newItem = newItem[0]; // select first document of the collection instance         }         var nodeToSelect = this.getRoot().findNodeByModel(newItem);         if (nodeToSelect != null) {             this.displayNode(nodeToSelect);         }     } } </pre>
<b>copyVisible</b>	<i>com.conceptwave.system.Boolean</i>	<pre> function copyVisible(\$psCondition) { // Runs every time the permission is evaluated     var permissionObject = this;      var selNode = this.getSingleSelection();     if (!\$psCondition    selNode == null)         return false;     var orderItem = this.getCopyItem(selNode.model);     return orderItem != null &amp;&amp; this.getAddGroup(orderItem.parent) != null; } </pre>
<b>cwLeafInitAction\$CurrentForm</b>	<i>com.conceptwave.system.NavigationTreeWithTabs.currentForm(String)</i>	<p>The initialization method of Variable <i>currentForm</i> that specifies the Form to display in the detail Form Frame, which defaults to Form <i>Single</i>.</p> <pre> function cwLeafInitAction(document) { // Leaf initialization rule. Used to generate script     return "Single"; } </pre>
<b>deleteAction</b>	<i>com.conceptwave.system.Void</i>	<p>Invoked when the user clicks on the delete menu.</p> <pre> function deleteAction() { // Metadata type method. Can be called by scripts.      var selNode = this.getSingleSelection();     if (selNode == null)         return null;     var nextSel = selNode.parentNode;     while (nextSel != null &amp;&amp; nextSel.tableDoc.excluded) {         nextSel = nextSel.parentNode;     }     var objectToDelete = this.getDeleteItem(selNode.model);     if (objectToDelete == null)         return null;     var treeSelMode = this.treeSel[0];     if (treeSelMode == selNode) {         this.treeSel = new Array();         this.hideDetailAction();     }     else if (this.tabs != null &amp;&amp; this.selectedTab != null) {         if (this.tabs.length == 1) {             this.tabs = null;         }         else if (this.selectedTab == ("" + (this.tabs.length - 1))) {             this.selectedTab = "" + this.tabs.length - 2;         }     }     var nextItem = objectToDelete.deleteItem();     if (nextItem != null) {         selNode = this.findNodeByModel(nextItem);         if (selNode != null) {             nextSel = selNode;         }         if (nextSel != null) {             this.displayNode(nextSel);         }     } } </pre>
<b>deleteAction_Confirmation</b>	<i>com.conceptwave.system.String</i>	<pre> function deleteAction_Confirmation() { // Metadata type method. Can be called by scripts.      return "UU0172:YN"; } </pre>
<b>deleteVisible</b>	<i>com.conceptwave.system.Boolean</i>	<pre> function deleteVisible(\$psCondition) { // Runs every time the permission is evaluated     var permissionObject = this;      if (!\$psCondition)         return false;     var selNode = this.getSingleSelection();     return selNode != null &amp;&amp; this.getDeleteItem(selNode.model) != null; } </pre>
<b>displayNode</b>	<i>com.conceptwave.system.Void</i>	<pre> function displayNode(node) { // Metadata type method. Can be called by scripts.      var root = this.getRoot();     var selected = this.getSingleSelection();     if (node != null &amp;&amp; selected != node) { </pre>

		<pre> var treeNode = this.getClosestTreeNode(node); if (treeNode != null) { root.setSelection(treeNode); if (node.metadata.getDisplayInParent()) { root.selectedTab = this.getTabByNode(node); } } } } } } </pre>
<b>excludePerm</b>	<i>com.conceptwave.system.Void</i>	<p>Exclude Permission that specifies whether the treee node is to be displayed (false) or be hidden (true) such that the node's children will be displayed directly under the node's parent</p> <pre> function excludePerm(\$psCondition) { // Runs every time the permission is evaluated var permissionObject = this;  if(this.metadata.getExclude()) return true; // Exclude single document collection instance node: only if it has the child document return this.model instanceof OrderItem &amp;&amp; this.model.isSingleDocCollectionInstance &amp;&amp; this.model[0] != null; } </pre>
<b>expandAll</b>	<i>com.conceptwave.system.Void</i>	<pre> function expandAll() { // Metadata type method. Can be called by scripts.  // Helper method: expands the node and all children this.metadata.doExpandAll(this); } </pre>
<b>formatMessage</b>	<i>com.conceptwave.system.Message</i>	<pre> function formatMessage(message) { // Metadata type method. Can be called by scripts.  // Adds node visual key to the text messages of the child nodes var text = message.text == null ? "" : message.text; if (message.owner != this &amp;&amp; message.owner instanceof com.conceptwave.systemTreeNode) { text = "&lt;b&gt;" + message.owner.onNodeVisualKey() + "&lt;/b&gt;" + text; } message.text = "&lt;span style='white-space:nowrap;'&gt;" + text + "&lt;/span&gt;"; return message; } </pre>
<b>getNodeMessage</b>	<i>com.conceptwave.system.Object</i>	<p>Retrieves and returns an array of Validation Error messages for the node. If boolean parameter <i>includeChildren</i> is true, recursively retrieves node messages from descendant nodes.</p> <pre> function getNodeMessages(messagesArray,includeChildren) { // Metadata type method. Can be called by scripts.  if (messagesArray == null) { messagesArray = new Array(); } if (this.errorList != null) { // ValidationErrorList.toMessageArray() method returns // an array of com.conceptwave.system.Message objects var errorMessages = this.errorList.toMessageArray(); for (var i = 0; i &lt; errorMessages.length; i++) { errorMessages[i].owner = this; messagesArray[messagesArray.length] = errorMessages[i]; } } if (includeChildren == null    includeChildren){ if (this.loaded &amp;&amp; this.childNodes != null) { for (var i = 0; i &lt; this.childNodes.length; i++) { this.childNodes[i].getNodeMessages(messagesArray, true); } } } return messagesArray; } </pre>
<b>getSingleSelection</b>	<i>com.conceptwave.system(TreeNode</i>	<p>The method to return a single selection of tree node, in the event where multiple nodes are selected in the Navigation Tree.</p> <pre> function getSingleSelection() { // Metadata type method. Can be called by scripts.  // Gets single node selection: // tree selection or selected tab var root = this.getRoot(); if (root.tabs != null &amp;&amp; root.tabs.length &gt; 0 &amp;&amp; root.selectedTab != null &amp;&amp; root.tabs[root.selectedTab] != null) { return root.tabs[root.selectedTab].owner; } if (root.treeSel != null &amp;&amp; root.treeSel.length &gt; 0) { return root.treeSel[0]; } return null; } </pre>
<b>hideDetailAction</b>	<i>com.conceptwave.system.Void</i>	The method that hides the detail pane in the Navigation Tree.

		<pre>function hideDetailAction() { // Metadata type method. Can be called by scripts.  var root = this.getRoot(); root.detail = null; root.detailForm = null; root.detailMenu = null; root.tabs = null; root.selectedTab = null; root.currentForm = "Single"; root.detailNode = null; }</pre>
<b>initNode</b>	<i>com.conceptwave.system.Void</i>	<p>Implementation function that initializes the tree node.</p> <pre>function initNode() { // Metadata type method. Can be called by scripts.return this.cw\$super_initNode();  this.loaded = false; this.dirty = true; // init model if (this.model != null) { // add listener to the model Notifier.register(this.model, this, "onModelEvent"); } this.applySearchMap(); // init table document this.tableDoc = new Document(this.metadata.getTableDocType()); this.tableDoc.id = this.onNodeID(); if (this.parentNode != null) this.tableDoc.parentId = this.parentNode.tableDoc.id; this.tableDoc.isFolder = this.metadata.isFolder(); this.updateTableDoc(); // refresh privileges this.refreshPrivileges(); if (this.metadata.getAutoExpand()) this.expand(); }</pre>
<b>isDisplaying</b>	<i>com.conceptwave.system.Boolean</i>	<p>The method that determines if the node is being displayed or not.</p> <pre>function isDisplaying(node) { // Metadata type method. Can be called by scripts.  if (!node.tableDoc.visible) { return false; } var sel = this.getTreeSelection(); if(sel != null &amp;&amp; sel.length &gt; 0 &amp;&amp; sel[0] == node) { return true; } if (node.metadata.getDisplayInParent() &amp;&amp; node.parentNode != null) { return this.isDisplaying(node.parentNode); } if (node.tableDoc.excluded) { var displayInParentNodes = this.getDisplayInParentNodes(node); if (displayInParentNodes != null &amp;&amp; displayInParentNodes.length &gt; 0 &amp;&amp; node.parentNode != null) { return this.isDisplaying(node.parentNode); } } return false; }</pre>
<b>isNodeSelected</b>	<i>com.conceptwave.system.Void</i>	<p>The method that determines if the node is being selected or not</p> <pre>function isNodeSelected(node) { // Metadata type method. Can be called by scripts.  var sel = this.getTreeSelection(); if (sel != null) { for (var i in sel) { if (sel[i] == node) { return true; } } } return false; }</pre>
<b>onAutoSave</b>	<i>com.conceptwave.system.Boolean</i>	<pre>function onAutoSave() { // Metadata type method. Can be called by scripts.  var canContinue = true; if (this.detailNode != null &amp;&amp; this.detailNode != this) { // save selected node canContinue = this.detailNode.onAutoSave(); } // save order if "Auto save" set for the root if (canContinue &amp;&amp; this.metadata.isAutoSave()) { canContinue = this.saveNode(this, true, false); } }</pre>

		<pre>         return canContinue;     } } </pre>
<b>onClick</b>	<i>com.conceptwave.system.Void</i>	<pre> function onClick() { // Metadata type method. Can be called by scripts. } </pre>
<b>onHover</b>	<i>com.conceptwave.system.Void</i>	<pre> function onHover() { // Metadata type method. Can be called by scripts.      var root = this.getRoot();     var hoverNode = root.navTree.hover;     root.hoverMessages = null;     if (hoverNode != null) {         var messages = new Array();         hoverNode.getNodeMessages(messages, true);         if (messages.length &gt; 0) {             for (var i = 0; i &lt; messages.length; i++) {                 this.formatMessage(messages[i]);             }         }         root.hoverMessages = messages;     } } root.cwShowHover = root.hoverMessages != null; } </pre>
<b>onInit</b>	<i>com.conceptwave.system.Void</i>	<p>Invoked from the onInit() method of the Order.</p> <pre> function onInit() { // Metadata type method. Can be called by scripts.      this.initNode();     var docNode = this.findFirstDocumentNode(this);     if (docNode == null) {         docNode = this.findFirstNonExcludedNode(this);     }     if (docNode != null) {         this.displayNode(docNode);     } } </pre>
<b>onModelEvent</b>	<i>com.conceptwave.system.Void</i>	<p>The system method that is triggered when an action event occurs at the node's model, such as user updating the model in the Form. This event handling is enabled at node's initialization by registering to the model's notification service.</p> <pre> function onModelEvent(event) { // Metadata type method. Can be called by scripts.      var action = event.type;     if ((action == "updateState"    action == "updateStatus"    action == "update") &amp;&amp; !this.metadata.isFinderNode()) {         this.updateTableDoc();         this.markDirty();     }     else if (action == "validationErrors") {         // Validation notify: parameter is a list of validation errors         if (event.params != null &amp;&amp; event.params.length &gt; 0 &amp;&amp; event.params[0] instanceof ValidationErrorList) {             this.setValidationErrors(event.params[0]);         }     }     else if (this.enableEvents) {         // disable model events temporarily to avoid looping         this.enableEvents = false;         this.onNodeUpdate(this, action, event.source, event.params);         this.enableEvents = true;     } } </pre>
<b>onNodeDetail</b>	<i>com.conceptwave.system.UserInterface</i>	<p>The system method that is triggered when the node is selected in the tree, to return the detail User Interface of the model object.</p> <pre> function onNodeDetail() { // Metadata type method. Can be called by scripts.      // Returns the detail user interface that is displayed     // when the node is selected in the tree     if (this.modelUI == null) {         this.modelUI = this.metadata.createModelUI(this, this.model);     }     return this.modelUI; } </pre>
<b>onNodeDetailForm</b>	<i>com.conceptwave.system.String</i>	<p>The system method that is triggered when the node is selected in the tree, to return the name of the detail Form of the model object to display to the user.</p> <pre> function onNodeDetailForm() { // Metadata type method. Can be called by scripts.      // Returns the form name of the detail user interface     // that is displayed when the node is selected in the tree     return this.metadata.getDetailForm(); } </pre>
<b>onNodeEdited</b>	<i>com.conceptwave.system.Void</i>	<p>The system method that is triggered when the row that represents the node in Tree Form Element, the model of which is the Table Document presented as Table Tree, is editable</p>

		<p>and updated by user. Typically, logic may be inserted to update the node's model based on the updated Table Document.</p> <pre>function onNodeEdited(event) { // Metadata type method. Can be called by scripts.  // Table document has been edited in browser // Insert logic here: apply changes to the model }</pre>
<b>onNodeId</b>	<i>com.conceptwave.system.String</i>	<p>The system method that calculates the node identifier string.</p> <pre>function onNodeId() { // Metadata type method. Can be called by scripts.  // The method is called on node initialization to get an unique node ID // By default the node ID is a node path in the tree // WARNING: the node ID cannot be null, must be unique in the tree // and cannot be changed after var id = this.generateNodeId(); if (this.parentNode != null) { id = this.parentNode.tableDoc.id + "/" + id; } return id; }</pre>
<b>onNodeIcon</b>	<i>com.conceptwave.system.String</i>	<p>The system method that calculates the resource path of the tree node icon image. It can have overlay images, similar to those in Navigation Pane of Velocity Studio itself such as <i>pencil</i> overlay for dirty marking, or <i>green arrow</i> overlay for overridden objects. To overlay an image, simply return a list of images separated by comma in this method. As an example, <i>return this.cw\$super_onNodeIcon() + "/cwf/add.gif(top-right)"</i>; returns all icon images with the "add" icon overlay embedded at top-right.</p> <pre>function onNodeIcon() { // Metadata type method. Can be called by scripts.  var image = this.metadata.getDefaultImage(); if (this.isOrderItem()) { if (image == null    image.length == 0) { image = this.model.isDocument ? "/cwf/TreeDocUnknown.gif" : "/cwf/TreeFolderUnknown.gif"; } if (this.model.hasAttachment) { image += ",/cwf/TreeAttach.gif(top-right)"; } if (this.model.hasValidStatus) { image += ",/cwf/ov/ok.png(bottom-right)"; } else if (this.model.hasErrorStatus) { image += ",/cwf/ov/error.png(bottom-right)"; } if (this.model.dirty) { image += ",/cwfv/edit.png"; } } return image; }</pre>
<b>onNodeTab</b>	<i>com.conceptwave.system.UserInterface</i>	<pre>function onNodeTab() { // Metadata type method. Can be called by scripts.  var detailUI = this.onNodeDetail(); var detailForm = this.onNodeDetailForm(); if (detailUI != null &amp;&amp; detailForm != null) { var tab = this.tabUI; if (tab == null) { tab = new com.conceptwave.system.TabUserInterface(); tab.owner = this; } tab.model = this.tableDoc; tab.id = this.tableDoc.id; tab.label = this.tableDoc.label; if (this.metadata.getShowCount()) { tab.label += '(' + this.metadata.countVisibleChildren(this) + ')'; } tab.image = this.tableDoc.image; tab.detail = detailUI; tab.detail.parent = tab; tab.detailForm = detailForm; this.tabUI = tab; } else { this.tabUI = null; } return this.tabUI; }</pre>
<b>onNodeUpdate</b>		<p>The system method that is invoked to update the node as well as the overall Navigation Tree based on an action event; invokes by <i>onModelEvent</i> method.</p> <pre>function onNodeUpdate(node,action,param1,param2) { // Metadata type method. Can be called by scripts.</pre>

		<pre> ar isSelected = action == "select"; var reload = isSelected; if (param2 != null &amp;&amp; param2.length &gt; 0) { param2 = param2[0]; } if (param2 instanceof OrderItem) { if (action == "add") { node.metadata.addNode(node, param2); } else if (action == "delete"    action == "deleteOp"){ node.metadata.removeNode(node, param2); } } node.updateNode(true, isSelected, true); if (isSelected) { node.expand(); } node.refreshPrivileges(); node.markDirty(); var navTree = this.getRoot(); if (navTree != null &amp;&amp; navTree.isDisplaying(this)) { navTree.showDetailAction(); } } </pre>
<b>onNodeVisualKey</b>	<i>com.conceptwave.system.String</i>	<p>The system method that computes the node's Visual Key (that is, label of the node).</p> <pre> function onNodeVisualKey() { // Metadata type method. Can be called by scripts.  var label = null; if (this.model instanceof Order) { if (this.model.hasMethod("cwOnOrdVisualKey")) { label = this.model.cwOnOrdVisualKey(); } } if (label == null &amp;&amp; this.model instanceof OrderItem) { var visKey = this.model.getVisualKeyDocument(); if (visKey != null &amp;&amp; visKey.hasMethod("cwOnDocVisualKey")) { label = visKey.cwOnDocVisualKey(); } } if (label == null) { label = this.metadata.getDefaultLabel(); } return label; } </pre>
<b>onSelChange</b>	<i>com.conceptwave.system.Void</i>	<p>Invoked when the user makes a selection on this Order table.</p> <pre> function onSelChanged() { // Metadata type method. Can be called by scripts.  var newSelNode = this.getSingleSelection(); if (this.detailNode != null &amp;&amp; newSelNode != this.detailNode) { this.confirmSelection = new Array(); if(this.treeSel != null) { for (var i = 0; i &lt; this.treeSel.length; i++) { this.confirmSelection[i] = this.treeSel[i]; } } this.confirmMethod = "confirmTreeSelection"; if (!this.validateCurrentNode()) { return; } this.detailNode.onAutoSave(); } this.confirmSelection = null; this.confirmMethod = null; this.showDetailAction(); } </pre>
<b>onTabChange</b>	<i>com.conceptwave.system.Void</i>	<p>The method that is triggered when tab selection is changed among Tabset in the detail pane.</p> <pre> function onTabChanged() { // Metadata type method. Can be called by scripts.  var newSelNode = this.getSingleSelection(); if (this.detailNode != null &amp;&amp; newSelNode != this.detailNode) { this.confirmTab = this.selectedTab; this.confirmMethod = "confirmTabSelection"; if (this.validateCurrentNode()) { this.detailNode.onAutoSave(); this.confirmTab = null; this.confirmMethod = null; this.confirmTabSelection(); } } } </pre>

<b>onTimer</b>	<i>com.conceptwave.system.Void</i>	Generic timer method that is periodically invoked.  function onTimer() { // Metadata type method. Can be called by scripts.
<b>onValidate</b>	<i>com.conceptwave.system.Void</i>	The system method that is triggered when validation happens: <ul style="list-style-type: none"> <li>• On User Action method invocation, if its <b>validate</b> checkbox is selected</li> <li>• On selection changed in the navigation tree</li> <li>• On tabs selection changed.</li> </ul> By default, validation executes all validation methods for all Variables in the User Interface as well as validation methods on the model (if exists).  function onValidate() { // Metadata type method. Can be called by scripts.  if (this.detailNode == this) { this.cw\$super_onValidate(); } else if(this.detailNode != null) { this.detailNode.onValidate(); } }
<b>pdfAction</b>	<i>com.conceptwave.system.Void</i>	function pdfAction() { // Metadata type method. Can be called by scripts.  if (this.model instanceof Order) { var download = new com.conceptwave.system.DownloadPdf(null, this); download.fileName = "cwf/order" + this.model.id + ".pdf"; download.xml = this.model.toXML(); download.xsl = this.model.getXslFileName(); if (download.xsl == null    download.xsl == "") { Global.throwException("UE0307"); } download.mimeType = "application/pdf"; Global.doDownload(download); }
<b>pdfVisible</b>	<i>com.conceptwave.system.Boolean</i>	function pdfVisible(\$psCondition) { // Runs every time the permission is evaluated var permissionObject = this;  return this.model instanceof Order && this.model.getXslFileName() != null && this.model.getXslFileName().length > 0; }
<b>refreshNode</b>	<i>com.conceptwave.system.Void</i>	function refreshNode(reloadChildren,keepExpanded) { // Metadata type method. Can be called by scripts.  // Helper method: forces to refresh the node and its children // if reloadChildren is true, children of the node will be recreated // Set keepExpanded to true to re-expande node's children // that was expanded before refreshing this.updateNode(false, reloadChildren, keepExpanded); this.refreshPrivileges(); }
<b>refreshPrivileges</b>	<i>com.conceptwave.system.Void</i>	function refreshPrivileges() { // Metadata type method. Can be called by scripts.  // Helper method: refreshes visible and exclude privileges // for the node and its children recursively. // Calls visiblePerm and excludePerm for each node. var visible = this.visiblePerm(); if (visible == null) { visible = true; } var excluded = this.excludePerm(); if (excluded == null) { excluded = this.metadata.getExclude(); } if (visible != this.tableDoc.visible    excluded != this.tableDoc.excluded) { this.tableDoc.visible = visible; this.tableDoc.excluded = excluded; this.markDirty(); } var children = this.childNodes; if (children != null) { for (var i = 0; i < children.length; i++) { children[i].refreshPrivileges(); } } }
<b>saveAllAction</b>	<i>com.conceptwave.system.Void</i>	function saveAllAction() { // Metadata type method. Can be called by scripts.  if (!this.addNode(this, true, true)) { Global.showUserMessage("UE0118"); } }
<b>saveAllVisible</b>	<i>com.conceptwave.system.Boolean</i>	function saveAllVisible(\$psCondition) { // Runs every time the permission is evaluated var permissionObject = this;

		<pre>         return true;     } } </pre>
<b>saveNode</b>	<i>com.conceptwave.system.Boolean</i>	<p>function saveNode(startNode,validate,recursively) { // Metadata type method. Can be called by scripts.</p> <pre> var diffModelUI = startNode.modelUI != null &amp;&amp; startNode.model != startNode.modelUI.model; if (validate &amp;&amp; !this.validateTheNode()) { return false; } if (recursively &amp;&amp; startNode.loaded &amp;&amp; startNode.childNodes != null) { var len = startNode.childNodes.length; for (var i = 0; i &lt; len; i++) { if(!this.saveNode(startNode.childNodes[i], validate, true)) { return false; } } } if (diffModelUI) { startNode.modelUI.onAutoSave(); } if (startNode.model instanceof OrderItem &amp;&amp; (startNode.model.isDocument    startNode.model.isTop)) { startNode.model.save(); } return true; } </pre>
<b>selectTab</b>	<i>com.conceptwave.system.Void</i>	<p>The method that sets a tab selection based on <i>index</i>.</p> <p>function selectTab(index) { // Metadata type method. Can be called by scripts.</p> <pre> var root = this.getRoot(); var selectedIndex = "" + index; if (index != null &amp;&amp; root.tabs != null &amp;&amp; root.tabs[selectedIndex] != null) { root.selectedTab = selectedIndex; } } </pre>
<b>setNodeSelected</b>	<i>com.conceptwave.system.Void</i>	<p>The method that includes a particular <i>node</i> as a selected node at the Order Element.</p> <p>function setNodeSelected(node,state) { // Metadata type method. Can be called by scripts.</p> <pre> var newSelection = new Array(); var found = false; var root = this.getRoot(); if (root.treeSel != null) { for (var i in root.treeSel) { if (root.treeSel[i] == node) { found = true; if (state) break; else continue; } newSelection[newSelection.length] = root.treeSel[i]; } } if (state &amp;&amp; !found) newSelection[newSelection.length] = node; if ((state &amp;&amp; !found)    (!state &amp;&amp; found)) root.setSelection(newSelection); } </pre>
<b>setSelected</b>	<i>com.conceptwave.system.Void</i>	<p>The method that sets an array of <i>nodes</i> as the list of selected nodes at the Order Element, changing Root Node Variable <i>treeSel</i>. Method <i>onSelChanged</i> is invoked if parameter <i>fireEvent</i> is true or missing.</p> <p>function setSelection(nodes,fireEvent) { // Metadata type method. Can be called by scripts.</p> <pre> if (nodes == null) { nodes = new Array(); } else if (nodes instanceof com.conceptwave.systemTreeNode) { var sel = new Array(); sel[0] = nodes; nodes = sel; } var root = this.getRoot(); root.treeSel = nodes; // expand paths if (nodes != null) { for (var i = 0; i &lt; nodes.length; i++) { var parentN = nodes[i].parentNode; while (parentN != null) { parentN.expanded = true; parentN = parentN.parentNode; } } } </pre>

		<pre>         }         if (fireEvent == null    fireEvent) {           root.onSelChanged();         }       }     </pre>
<b>setValidationError</b>	com.conceptwave.system.Void	<p>Populates validation error list Variable <i>errorList</i> from parameter <i>validationErrorList</i>.</p> <p>function setValidationErrors(validationErrorList) { // Metadata type method. Can be called by scripts.</p> <pre> if (validationErrorList instanceof ValidationErrorResponse) {   // Use ValidationErrorResponse.getSubList() to get a subset of error messages   // that belong to the data object   // Parameters: 1 - data object(passing null returns all messages),   // 2 - true to include child objects messages   // 3 - true to include field errors   this.errorList = validationErrorList.getSubList(this.model, false, false); } else if (errorList == null) {   this.messages = null; } </pre>
<b>showDetailAction</b>	com.conceptwave.system.Void	<p>This method is triggered when the detail pane is to be displayed based on the node selection.</p> <p>function showDetailAction() { // Metadata type method. Can be called by scripts.</p> <pre> if (!this.isRoot()) {   // this is nested navigation tree - show detail action on the root level   this.getRoot().showDetailAction();   return; } this.hideDetailAction(); // clear detail variables if (this.treeSel != null &amp;&amp; this.treeSel.length &gt; 0) {   var selNode = this.treeSel[0];   var displayTabs = false;   var tabsArray = new Array();   var tabNodes = this.getDisplayInParentNodes(selNode);   if ((tabNodes != null &amp;&amp; tabNodes.length &gt; 0)    selNode.metadata.getShowTabsAlways()) {     // display selected tree node as at the first tab     this.addTab(tabsArray, selNode);     // add "display in parent" children     if (tabNodes != null) {       var len = tabNodes.length;       for (var i = 0; i &lt; len; i++) {         this.addTab(tabsArray, tabNodes[i]);       }     }     if (tabsArray.length &gt; 0) {       this.detailNode = tabsArray[0].owner;       this.detailMenu = tabsArray[0].owner.isRoot() ? tabsArray[0].detail : tabsArray[0].owner;       displayTabs = tabsArray.length &gt; 1    selNode.metadata.getShowTabsAlways();       if (!displayTabs) {         // display single detail form if only 1 tab         this.detail = tabsArray[0].detail;         this.detailForm = tabsArray[0].detailForm;       }     }   }   else {     this.detailNode = selNode;     this.detail = selNode.onNodeDetail();     this.detailForm = selNode.onNodeDetailForm();     this.detailMenu = selNode.isRoot() ? this.detail : selNode;   }   if (displayTabs) {     this.currentForm = "Tabs";     this.tabs = tabsArray;     this.selectedTab = "0";   }   else {     this.currentForm = "Single";     if (this.detail != null) {       this.detail.parent = this;     }   }   if (this.detailMenu != null) {     this.detailMenu.parent = this;   } } </pre>
<b>taskAction</b>	com.conceptwave.system.Void	<p>function tasksAction() { // Metadata type method. Can be called by scripts.</p> <pre> var vers = this.getCompatibleVersion(); if (vers != 0 &amp;&amp; vers &lt; 6) { </pre>

```

// display finder
var selectedItems = this.getSelectedOrderItems();
if (selectedItems == null || selectedItems.length == 0) {
    Global.throwException("UU0115");
}
var f = new Finder("cwf_pm:OrderItemWorklist");
if (selectedItems[0].isTop) {
    f.searchDocument.OrderId = selectedItems[0].id;
}
else {
    var ids = cwf_oe.getSelectedOrderIDs(selectedItems);
    if (!ids) {
        Global.throwException("UU0115");
    }
    f.searchDocument.orderIDs = ids;
}
f.search();
return f;
}
else
return publishEvent("WORKLIST_TASK_FINDER", FIRST_ONE, null, [null, this.model.id]);
}

```

taskHistoryAction	com.conceptwave.system.Void	<pre> function tasksHistoryAction() { // Metadata type method. Can be called by scripts.  var vers = this.getCompatibleVersion(); if (vers != 0 &amp;&amp; vers &lt; 6) { var selectedItems = this.getSelectedOrderItems(); if (selectedItems == null    selectedItems.length == 0) { Global.throwException("UU0115"); } var f = new Finder("cwf_pm:WorklistArchiveFinder"); if (selectedItems[0].isTop) { f.searchDocument.OrderId = selectedItems[0].id; } else { var ids = cwf_oe.getSelectedOrderIDs(selectedItems); if (!ids) { Global.throwException("UU0115"); } f.searchDocument.orderIDs = ids; } f.search(); return f; } else return publishEvent("WORKLIST_ARCHIVE_FINDER", FIRST_ONE, null, [null, this.model.id]); } </pre>
updateNode	com.conceptwave.system.Void	<p>Helper method to update the node in the Navigation Tree such as to update Table Document, reload children nodes, and keeping expanded states of the node and its children at refresh, if the corresponding boolean parameter is enabled.</p> <pre> function updateNode(updateTableDocument,reloadChildren,keepExpanded) { // Metadata type method. Can be called by scripts.  var expandedStates = {}; if (keepExpanded == null    keepExpanded) this.getExpandedStates(expandedStates); else expandedStates[this.tableDoc.id] = this.tableDoc.expanded;  if (updateTableDocument) { this.updateTableDoc(); } if ((reloadChildren == null    reloadChildren) &amp;&amp; this.isLoading()) { // reload children this.loadChildren(); } this.setExpandedStates(expandedStates); this.markDirty(); } </pre>
updateTableDoc	com.conceptwave.system.Void	<p>The helper method that implements the Table Document update. Invokes <i>applyTableDocMap()</i>.</p> <pre> function updateTableDoc() { // Metadata type method. Can be called by scripts.  this.applyTableDocMap(); this.tableDoc.label = this.onNodeVisualKey(); this.tableDoc.image = this.onNodeIcon(); if (this.tabUI != null) { // refresh tab label and image this.tabUI.label = this.tableDoc.label; if (this.metadata.getShowCount()) { this.tabUI.label += ' (' + this.metadata.countVisibleChildren(this) + ')'; } } </pre>

		<pre>         }         this.tabUI.image = this.tableDoc.image;     } } </pre>
<b>validateAction</b>	<i>com.conceptwave.system.Void</i>	<pre> function validateAction() { // Metadata type method. Can be called by scripts.  if (this.model != null &amp;&amp; this.model instanceof Order) { // Validate order (max errors per order) this.model.validate(5, false, false); } } </pre>
<b>validateCurrentNode</b>	<i>com.conceptwave.system.Boolean</i>	<pre> function validateCurrentNode() { // Metadata type method. Can be called by scripts.  if (this.detailNode != null &amp;&amp; (!this.detailNode.validateTheNode()    !this.validateTabs())){ // restore selection var sel = new Array(); sel[0] = this.getClosestTreeNode(this.detailNode); this.setSelection(sel, false); var tab = this.getTabByNode(this.detailNode); if (tab != null) { this.selectedTab = tab; } // display continue conformation this.continueConfirmation(); return false; } return true; } </pre>
<b>validateTab</b>	<i>com.conceptwave.system.Boolean</i>	<pre> function validateTabs() { // Metadata type method. Can be called by scripts.  if (this.tabs != null) { var tab; for (var i = 0; i &lt; this.tabs.length; i++) { tab = this.tabs[i]; if (tab.owner != null) { if (!tab.owner.validateTheNode()) return false; } } } return true; } </pre>
<b>validateTheNode</b>	<i>com.conceptwave.system.Boolean</i>	<pre> function validateTheNode() { // Metadata type method. Can be called by scripts.  var diffModelUI = this.modelUI != null &amp;&amp; this.model != this.modelUI.model; if (this.modelUI != null) { if (this.modelUI == this) { this.cw\$super_onValidate(); } else { this.modelUI.onValidate(); } if (!this.modelUI.isValid) return false; } if ((diffModelUI    this.modelUI == null) &amp;&amp; this.model instanceof OrderItem){ var err = this.model.validate(5,true,false); if (err != null) { return false; } } return true; } </pre>
<b>visiblePerm</b>	<i>com.conceptwave.system.Boolean</i>	<pre> function visiblePerm(\$psCondition) { // Runs every time the permission is evaluated var permissionObject = this;  if (this.model instanceof CwObject &amp;&amp; this.model.hasMethod("visiblePerm")) { return this.model.visiblePerm(); } return true; } </pre>

## Order Forms

Order Forms are similar to the Forms in [Navigation Tree with Tabs](#), with the following added metadata to implement Order presentation:

- In **Default** Form, the Tree Form Element for navigation remains, but is altered to show Validation Errors upon Hover at the Tree Form Element.
- In **Menu** Form, action menu items (Add, Attach, Copy, Delete, Validate, SaveAll, PDF, Tasks, Task History) are attached added to allow such management to the Order.
- Added **Messages** Form to show the Validation Errors upon Hover.

## Order Hover

There are four properties used to configure the hover option:

- Form
- Method
- Style
- Variable

By default, the Order User Interface has a hover form called Messages, which has an implemented onHover() method. This method computes what needs to be displayed in the hover form.

If you require overriding the behaviour of this method, do the following:

1. Override the onHover() method with the required logic.
2. Create a new hover form (for example, fHover), containing a layout element, and either a Label or HTML Content element.
3. **Override the Default** form and **Copy and Replace** the tree element inside, which becomes editable. For the hover form, select your Form (that is, fHover), Method, Style, and Variable.

**Note:** The hover action for the order tree sometimes slows down the tree selection and should be invoked, if required. By default, the order onHover() action is only used if the node has state messages, such as error, warning, or info. If the order onHover() action has been overridden, anything can be displayed in hover Form. In this case, the onHover() action is invoked on each mouse hover over the node.

## Best Practices for Order Structures

---

When modelling an order, follow these guidelines:

### Model Network-Based Products

Network based products, such as Frame Relay, ATM, and IP VPN, consist of a series of sites or nodes, and connections or links. Ensure following when modelling these types of orders:

- Order entry is simple and logical.
- Network components may be ordered, provisioned, and placed in-service independently from other components of the network. For example, the provisioning of a 200-node network may never actually complete due to change orders placed by the customer. It is essential that components may be placed in-service as they are completed.
- Network components should be able to be changed independently from other components of the network. For example, disconnection of PVC between two sites should not be encumbered by provisioning another PVC to one of the sites.
- The relationship of the components of the network is respected. For example, a disconnection order against a site should force the disconnection of all associated PVCs.
- Data entry, such as site information, should be limited to one per order.

From a design perspective, follow these guidelines for a selected order model:

- Represent nodes as a collection with 0 to N instances. For example:
  - 0: A new PVC to two existing sites.
  - 1: A new PVC to a new site, from an existing site.
  - 2 or more: new sites and PVCs.
- Represent links as a collection with 0 to N instances. These instances are linked only by reference to the nodes and are not explicit children of the nodes (since they must be children of two nodes). For example:
  - 0: A change order on a site that does not affect the connected links.
  - 1 or more: new PVCs.
- A link includes a reference to the two endpoints without requiring the definition of those endpoints, which means that they need not be located on the order. The following possibilities can be permitted (Frame Relay used as an example):
  - A PVC with an endpoint being a new site defined on the same order.
  - A PVC with an endpoint being a new site defined on another order, not yet completed.
  - A PVC with an endpoint being an existing site.
  - A PVC with an endpoint being a site under a different service (for example, Frame Relay to ATM interworking).
  - A PVC with an endpoint belonging to another customer.
  - A PVC with an endpoint being a gateway, such as the Internet, an interconnection to another service provider, and so on.
- An order should be able to be split, either physically or logically, so that the complete portions can be considered closed, while the outstanding portions can be managed under a new order. This splitting permits closing activities, such as billing activation, service turn-up, and so on that may be modelled once per order, to be performed on the completed components.
- An order that affects a node also auto-discovers the impact on the links:
  - A disconnection on the node automatically require a disconnection on any associated links.
  - A bandwidth change on a node to below that specified on a link illicits a warning.
  - Provide a wizard to automatically add the affected links.

### When a Link is Explicitly Ordered or Modelled

In some network-based products, a topology selection drives the network configuration. For example, if a topology of fully meshed is selected, links are assumed between each of the sites. Do these links (for example, PVCs) need to be physically ordered? The links are modelled if the answer is yes to any of the following questions:

- Can the topology be altered through the addition or removal of individual links, or both?
- Can the characteristics of the links vary, such as the bandwidth, class of service, and so on?
- Are there individual provisioning activities, such as configuration, required on each link?
- Are the links itemized billable entities?

- Are the links to be represented individually in the inventory system?

In the event that the links are physically ordered, provided wizards automatically create the relevant links based on topology selected by network.

## Manage Lists of Orderable Items

In many products, the order data includes potentially lengthy lists of items. Examples include telephone numbers, dialing plans, IP addresses, end users, time-of-day routing rules, and more.

There are two basic models for managing this data:

- On-order
- Off-order

**On-Order modelling** is the traditional mode. A collection is defined on the order to hold and manage the list of items:

- The order definition includes these items.
- The items are automatically loaded and displayed to the end user with other order data.
- The length of the order can be managed by selecting **minimized** under the collection folder, which displays the folder in **unexpanded** mode (the default is fully expanded).
- Long lists can be cumbersome as no filtering is provided.
- Actions are limited to the one-item-at-a-time operation as multi-select actions are not supported.
- Instances cannot be hidden elegantly; the item remains visible in the tree, although the details are blocked.
- Access to the data within business rules is simplified.
- System performance is compromised since the lists are retrieved on order load.

**Off-order modelling** consists of defining documents and finders to manage the list of items. The items are not part of the order. Buttons or menus, or both are defined to access the items through finders. In this model:

- Navigation through the order is simplified as data is accessed by buttons or menus using popup finders.
- Long lists of data are better managed through the finder filter capability.
- Individual items can be hidden through filters.
- Multi-select operations, such as delete, are permitted.
- Access to the data within business rules is complicated as finders must be invoked.
- System performance is improved since the lists are only retrieved when explicitly requested.
- Order copy, archiving, and deleting functionality must be scripted to accommodate the off-order data.

In general, the longer the list of items, the more suited off-order modelling will be to the situation.

## Attachments in Documents and Orders

In product version 5.x, the following metadata objects were used to add, load, and display attachments to document and order items:

- cwf\_oe.AttachmentFinder
- cwf\_oe.CWDOCATTACHMENT

The following table describes the metadata object pertaining to attachments in the product:

Metadata Object	Description
cwf.attachmentDoc	This object replaces the cwf_oe.CWDOCATTACHMENT document. It has the same structure (variable names, variable order, and mapping to database) as the old document to support backward compatibility. The document is located in system metadata and has no child UI.
ui_common.attachmentFinder	This object denotes the new attachment finder implementation. It has a child UI based on the ui_common.baseFinder user interface.
ui_common.attachmentSearch	This object is a search document that replaces cwf_oe.AttachmentSearch. The search document has no default child UI.
ui_common.attachmentUI	This object represents the top-level UI for the attachment document detail view.
ui_common.attachmentResult	This object denotes the top-level UI for the attachment finder's result table.

## Events

The following table describes events pertaining to attachments. Parameters that are bolded signify that they are mandatory:

Event	Parameters	Description
SYSTEM_ATTACHMENT_ADD	<ul style="list-style-type: none"> <li>• owner (order item)</li> <li>• description</li> <li>• <b>fileName</b></li> <li>• mimeType</li> <li>• note</li> <li>• <b>attachmentData</b></li> <li>• attachmentDoc</li> <li>• <b>attachmentId</b></li> </ul>	The upload servlet calls this event to store the uploaded file into the attachment document. There are three scenarios to consider: <ul style="list-style-type: none"> <li>• Only the attachmentId is provided</li> <li>• The attachmentDoc is provided</li> <li>• The attachmentDoc is NOT provided</li> </ul>
SYSTEM_ATTACHMENT_FILE_GET	<b>attachmentId</b>	This event loads the attachment file by the given attachmentId and returns array [attachmentFile, fileName, mimeType].
SYSTEM_ATTACHMENT_QUERY	<ul style="list-style-type: none"> <li>• <b>ownerId</b></li> <li>• topOrderId</li> </ul>	This event returns a list of cwf.attachmentDocs by the given parameters.
SYSTEM_ATTACHMENT_INFO_QUERY	<b>ownerId</b>	This event returns attachInfoArr. The return information is an array in the following format: [attachmentId_1, attachmentName_1, attachmentMimeType_1, ..., attachmentId_N, attachmentName_N, attachmentMimeType_N ]
SYSTEM_ATTACHMENT_REMOVE	<ul style="list-style-type: none"> <li>• <b>docId</b></li> <li>• topOrderId</li> </ul>	This event deletes all attachments for the given document or order. This event is called when deleting the object from the database.
UI_GET_ATTACHMENT_FINDER	<ul style="list-style-type: none"> <li>• <b>ownerItem</b> (order item)</li> <li>• parentUI (for example, order UI)</li> </ul>	This event creates, initializes, and returns the finder attachment UI for the given ownerItem.

## Get and Display the Attachment Finder UI

To get an instance of the attachment finder, the UI\_GET\_ATTACHMENT\_FINDER event should be fired with the following parameters:

```
[ownerObject, parentUI]
```

The following example retrieves an instance of the finder for the document, which can be an action in the document UI:

```
var finderUI = publishEvent("UI_GET_ATTACHMENT_FINDER", FIRST_ONE, null, [this.model,this]);
return finderUI;
```

To see an example of this action, see com.conceptwave.system.OrderUserInterface > attachAction from Velocity Studio's Library tab.

## Customize the Default Behaviour

The UI Common template provides the default handler for the event to return the ui\_common.attachmentFinder implementation.

In your metadata, you can provide your own handler to return the old attachment implementation (cwf\_oe.AttachmentFinder) or any own extended finder.

## Life Cycles

---

Life Cycle objects, also known as State Machines, define a set of states and the directed transitions between them. The list of all state codes is defined by an enumeration data type, known as the state codes list, which is selected in the **Enumeration** field on the **General** tab.

The state nodes specify the states of some object. It can be an Order, Order Item, Document that is used as trouble ticket or any other business entity that has a notion of state. It is important to put emphasis on the fact that the Life Cycle diagram is attached to a data element so a single Order, Order Item or Document may use different Life Cycle diagrams in different situations. State nodes may contain an optional state script and Decision Tree.

Transitions define a possible switch from one state to another. Transitions from a given state are an ordered set of elements. They may contain an optional condition script, action script and Decision Tree.

The actions which the Life Cycle diagram performs are coded into the state and transition action scripts defined in the state and link property dialogs (refer to sections on Life Cycle State Properties and on Life Cycle Link Properties). This guarantees high performance and is a preferable solution when these actions rarely change.

The runtime scripts, state script, transition condition script and transition action script take the following parameters:

- *Document*: The Document that contains the state code field.
- *Item*: The field name of the state code field in the Document.
- *Input*: The input message.
- *theTree*: The Decision Tree for the state or transition action script.

The Decision Tree runtime object list takes the following parameters:

- *Document*: The Document that contains the state code field.
- *Item*: The field name of the state code field in the Document.
- *Input*: The input message.

The Life Cycle diagram is used in two ways:

1. As an Order Life Cycle Diagram to specify a set of valid transitions and optional actions from one Order state to another. The LifeCycle is set to associate with an Order, in the **States** field on the **General** tab of the Order properties, enforcing an ordering process following the defined state flow.
2. As a State Machine that performs actions based on the object state and an incoming message connected with any object that has a notion of state.

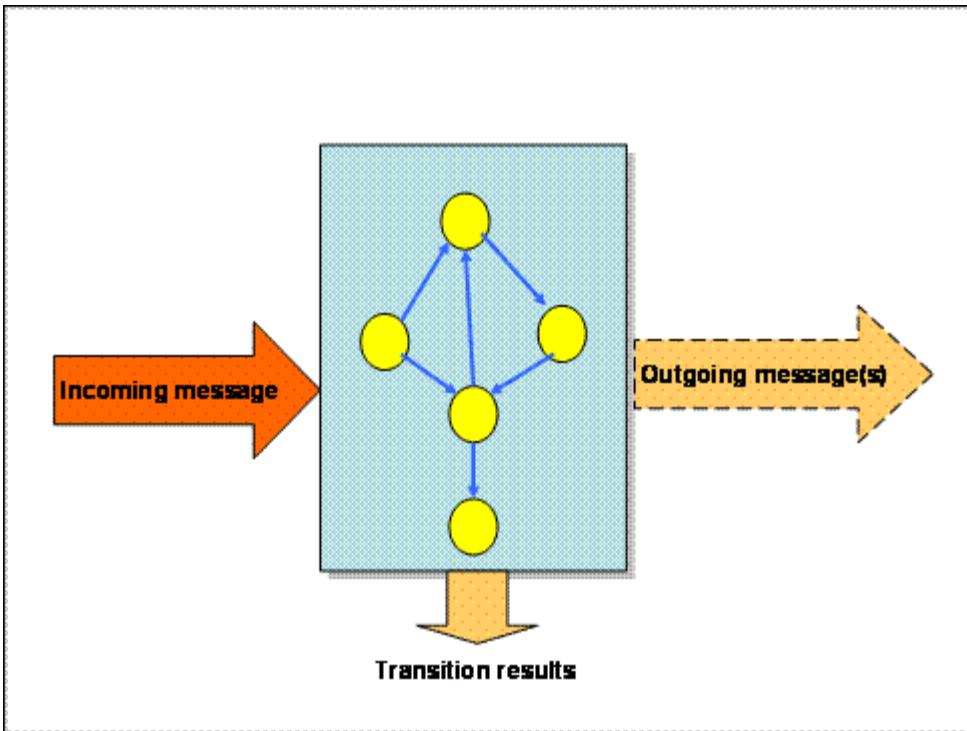
### Order Life Cycle Diagram Behaviour

The Life Cycle diagram used in an Order is called an Order Life Cycle Diagram. Their state codes list should be based on the system type *All States* and it should contain the state code NEW. The AVM automatically sets the code of any newly created Orders to NEW.

The Order Life Cycle Diagram defines whether the transition from one Order state to another is allowed. Their states use a property that is not related to the State Machine model, **Allow invalid state**, that specifies whether or not an Order can go into a given state if it is invalid.

### State Machine Behaviour

The State Machine can be invoked automatically by the AVM or by a script through the Document method runStateMachine(String fieldName, Document message, String stateMachineName). It always works in the context of a given incoming message.



The State Machine performs its actions as scripts so it can generate zero, one or more output messages to different internal or external consumers.

The processing of an incoming message is an atomic transaction that returns a value called the *transition result*. The State Machine as a whole is performing asynchronously, each incoming message processed in the moment it comes.

The algorithm for processing a single message is:

1. Find the state in the State Machine which corresponds to the current enumeration value (state code). If such a state does not exist throw a runtime exception.
2. Run the state script (if any). If the script returns a new state code find the corresponding transition from the current state to the new state and continue to 4. If no corresponding transition is found throw a runtime exception.
3. Run the condition scripts of the transitions from the current state in the order they are defined and take the first one which returns true (there may not be one).
4. If no transition is found then return null.
5. Run the transition action script and store the returned value into the transition result. If there is no script then set the transition result to null.
6. Set the enumeration data to the new state code as defined by the transition.
7. Return the transition result.

## Creating a Life Cycle

The **Life Cycle** element is created based on a series of user-defined Order states that can optionally be used by any Order. Before a Life Cycle element can be created, the user-defined Order states enumeration must be created as a **Data Type** element under the **Data Dictionaries** folder, based on the system-defined data type *State* from the cwf namespace (set the **Base type** to *State* or *All States* on the **General** tab).

Enumeration values are required when defining such state elements, in which at least a NEW state must be included for *Order* state enumerations. Neither a NEW or END is required for non-*Order* state enumerations.

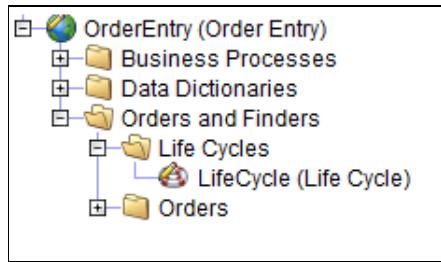
The figure below shows an example of enumeration values defined for the data type OrderPhase. In this example, seven states are defined: CAN, NEW, PRV, REV, END, REJ and TMP.

Enumeration:	Code	Description	Active
	CAN	Cancel	<input checked="" type="checkbox"/>
	NEW	New	<input checked="" type="checkbox"/>
	PRV	Provision	<input checked="" type="checkbox"/>
	END	End	<input checked="" type="checkbox"/>
	REJ	Reject	<input checked="" type="checkbox"/>

To create a Life Cycle:

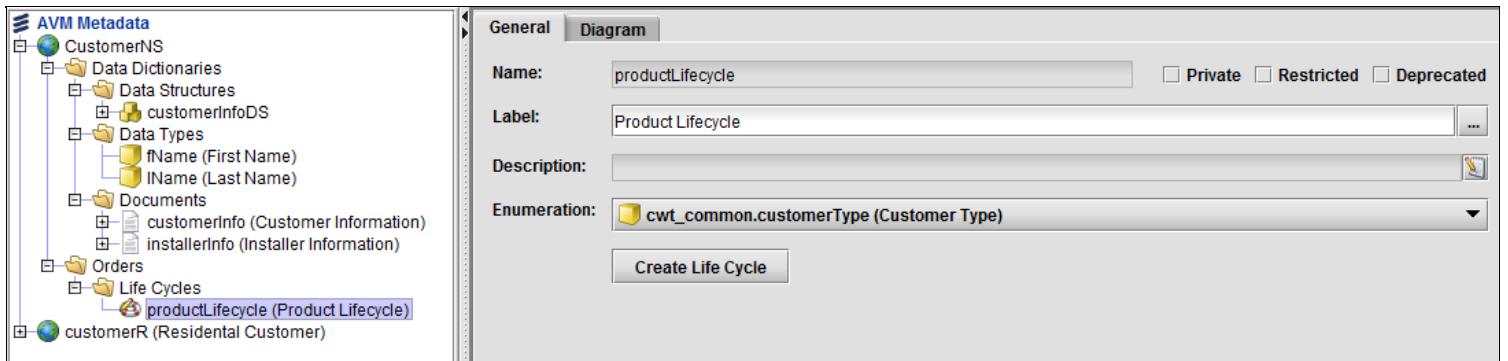
1. Right-click the **Life Cycles** folder in the **Orders** folder in the **Navigation** pane of Velocity Studio.
2. Select the **New Life Cycle** command. Give your Life Cycle a **Name** and **Label**, and then select your **Enumeration** datatype that you previously created. Click the **Finish** button to complete the process.
3. The **General** and **Diagram** tabs will appear where the Life Cycle properties are defined.
4. Click the **Create Life Cycle** button from the **General** tab to create the Life Cycle.

After the Life Cycle is created, the icon will be added under **Life Cycles** folder (see figure below).



## Life Cycle General Properties

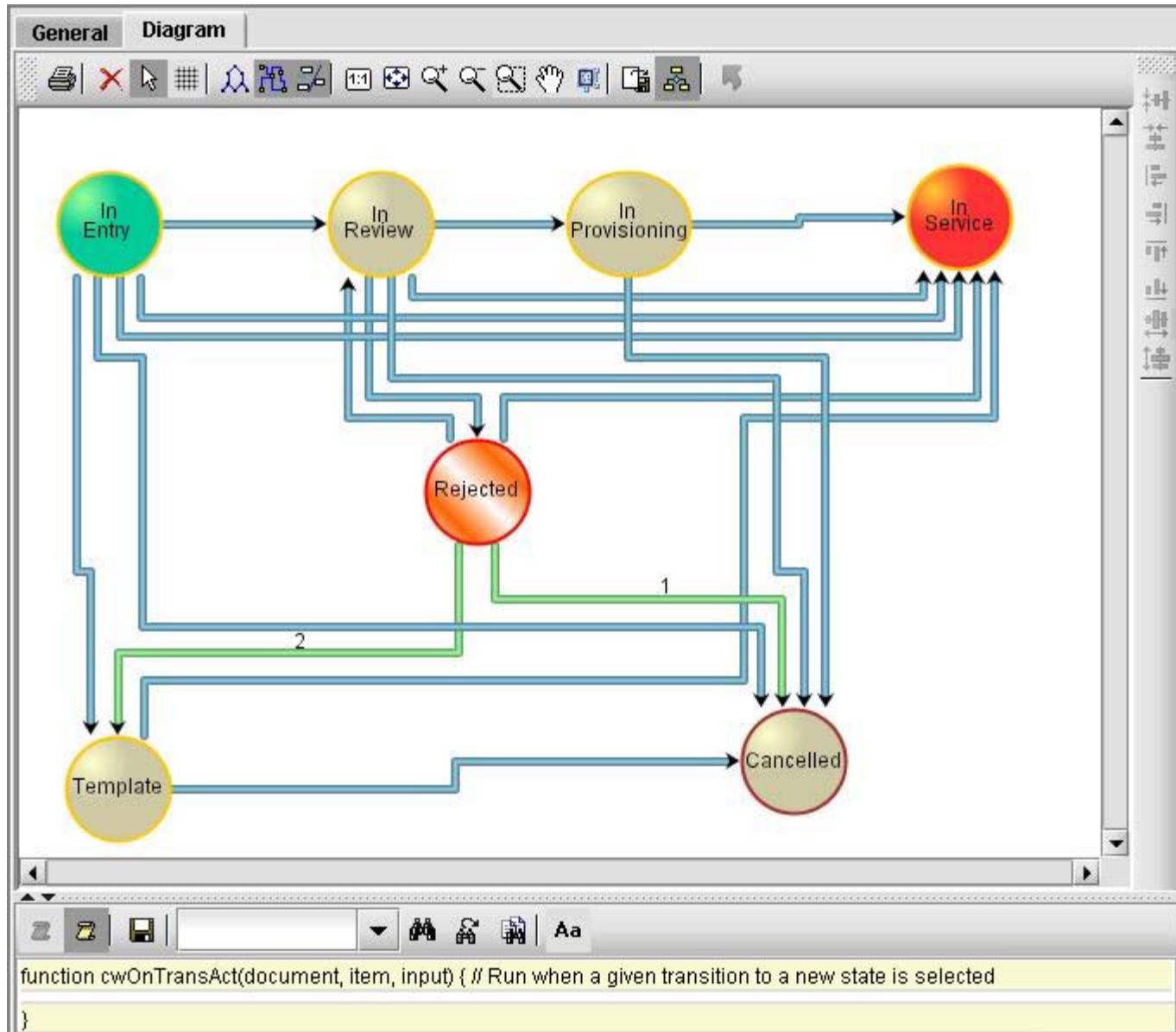
The general Life Cycle properties are defined on the **General** tab (see figure below).



The **Enumeration** field is mandatory and shows a list of user-defined state data elements, allowing you to decide which state element is appropriate for this Life Cycle entity.

## Life Cycle State Diagram

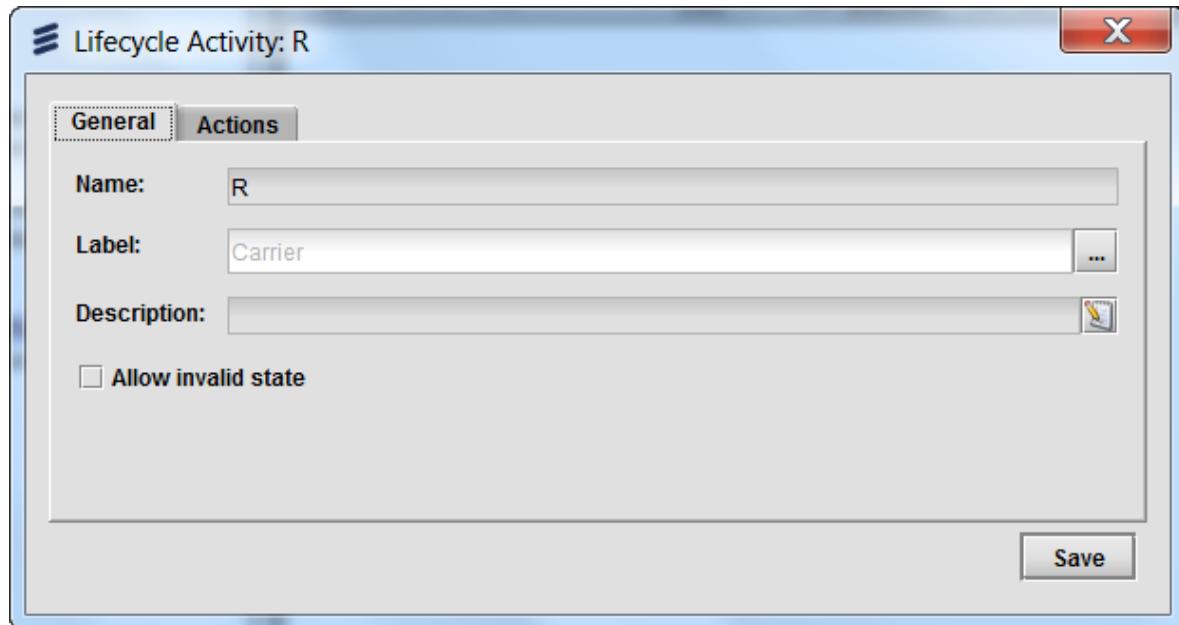
The **Diagram** tab (see figure below) provides a graphical editor for drawing a Life Cycle state flow diagram. Once the **Enumeration** field in the **General** tab is filled in with a state element, the analogous icons will be presented automatically in the editor.





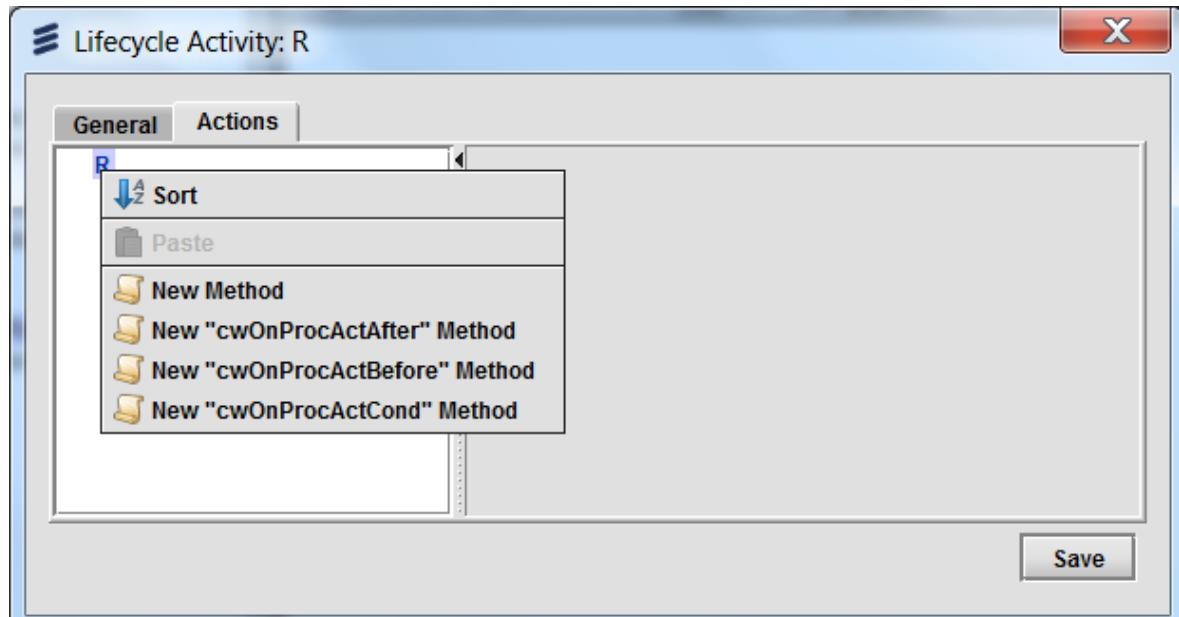
## Life Cycle State Properties

When the user right-clicks on a **state** icon in the **Diagram** tab and selects the **Properties** command, the Life Cycle State **Process property** dialog will open where the state can be edited. The table that follows provides a description of the fields.

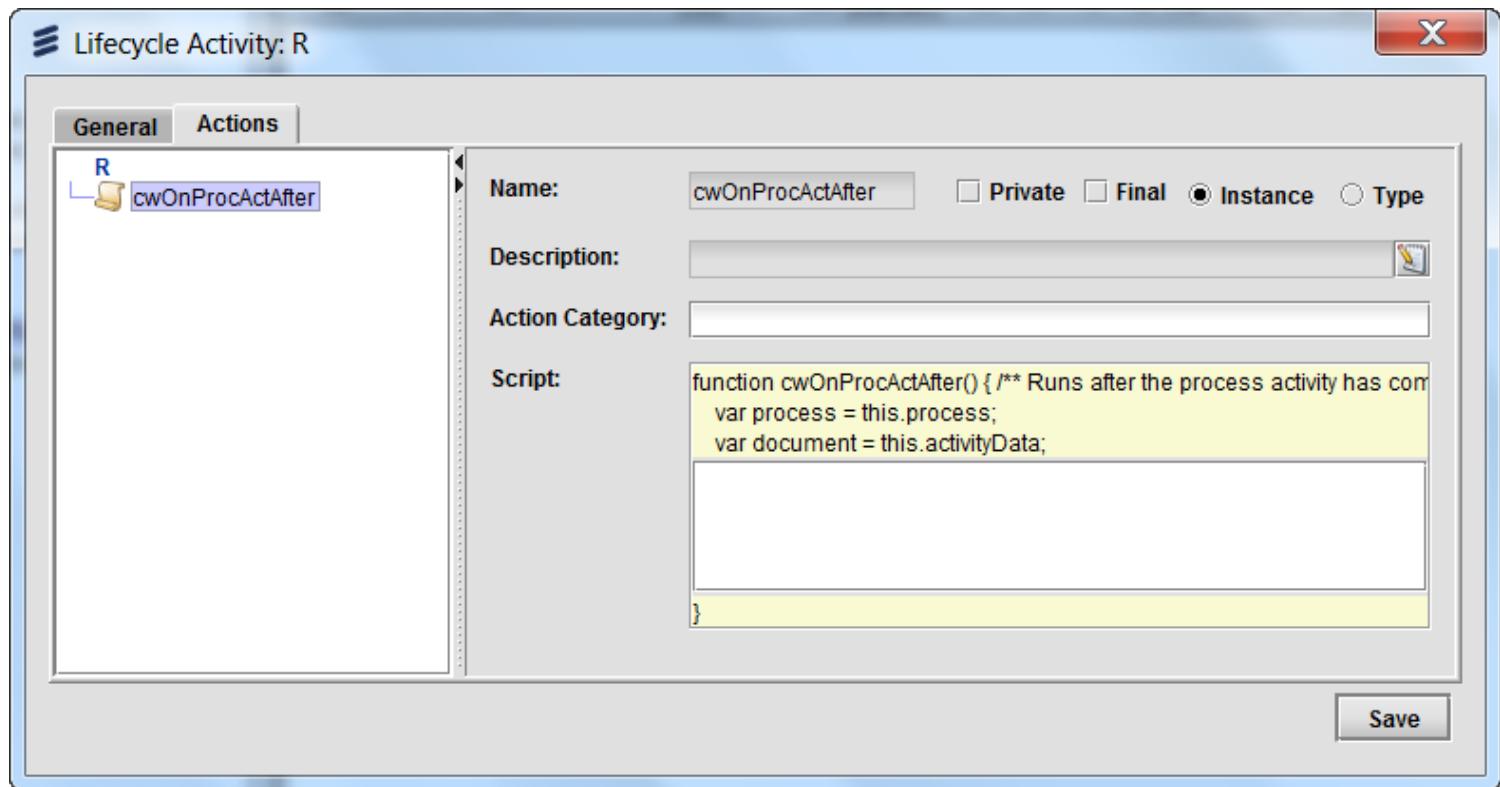


Field	Description
Name	Read-only. The state code defined in the enumeration.
Label	Read-only. The state description defined in the enumeration.
Description	Descriptive text for documentation purposes.
Allow invalid state	This attribute only applies to Order Life Cycle Diagrams. The attribute can be set to any node, except the state with the NEW code, to allow the Order state to be changed without concern over incorrect status.
Actions	The Actions tab houses state scripts.

The Action folder enables you to create or select from various default methods relating to the state by right-clicking the State Code (that is, Prov).



The Action properties folder appears, which allows you to create or change the script.



Normally at runtime, when an Order state is changed according to an Order Life Cycle Diagram definition, the system checks two things before it changes the state:

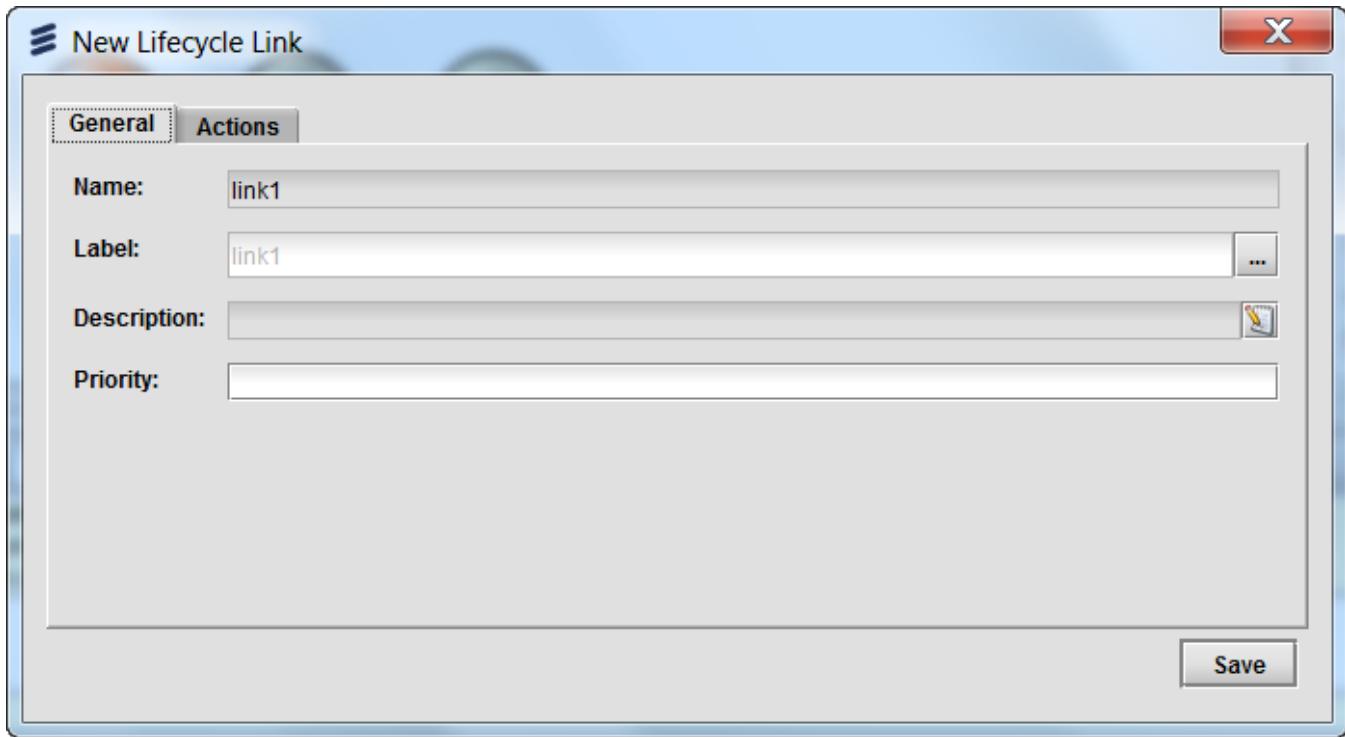
1. Whether the current Order is valid.
2. Whether the current Order is complete.

If either of these conditions fails, an internal error message box will pop up and the Order state will not be changed to the next state in the state flow. However, if the **Allow invalid state** check box is checked, the invalid or incomplete status will be ignored and the Order will move to the next state.

## Life Cycle Link Properties

To indicate directions from one state to another, a link or transition needs to be generated between states. To create such a link, follow these steps:

1. Click the **first state** icon to highlight it (the original state).
2. Place the mouse cursor over the **second state** icon (the next state) and right-click.
3. Select the **Create Link** command; the New Life Cycle Link dialog appears.



The following table describes the fields:

Field	Description
Name	The field indicates the name of the link and it is read-only.
Label	The value of this field is read-only.
Description	This field contains the descriptive text for documentation purposes.
Priority	This field indicates the priority of the link.

4. Click the Save button.

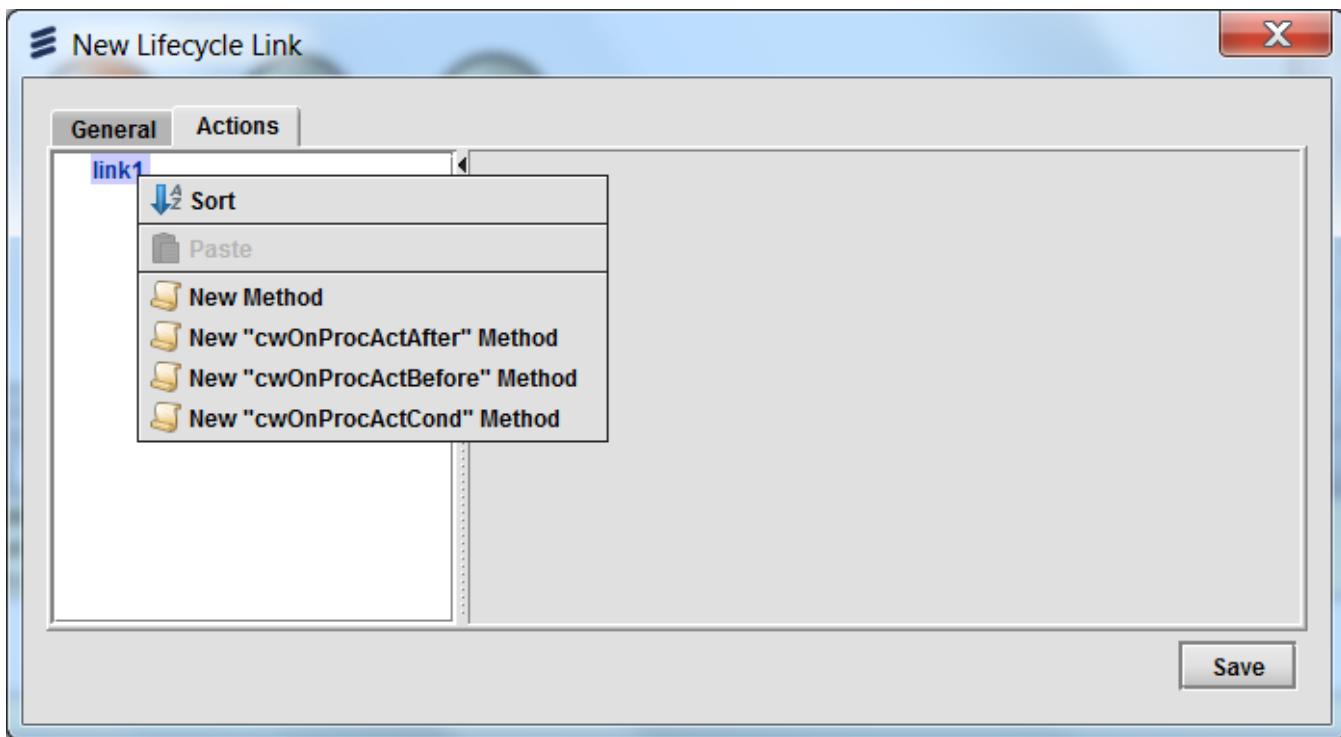
A link from a state to itself can also be created. To do so, click the **state** icon to highlight it and then place the mouse cursor over the same **state** icon. Right-click and select the **Create Link to Itself** command. A link will be created from the state to itself.

Bi-directional links are allowed between a pair of user created states, for example, between the states In Review and Rejected in the figure above. The **start state** can have incoming and outgoing links, including links to itself. The **end state** can only have incoming links, which excludes linking to itself.

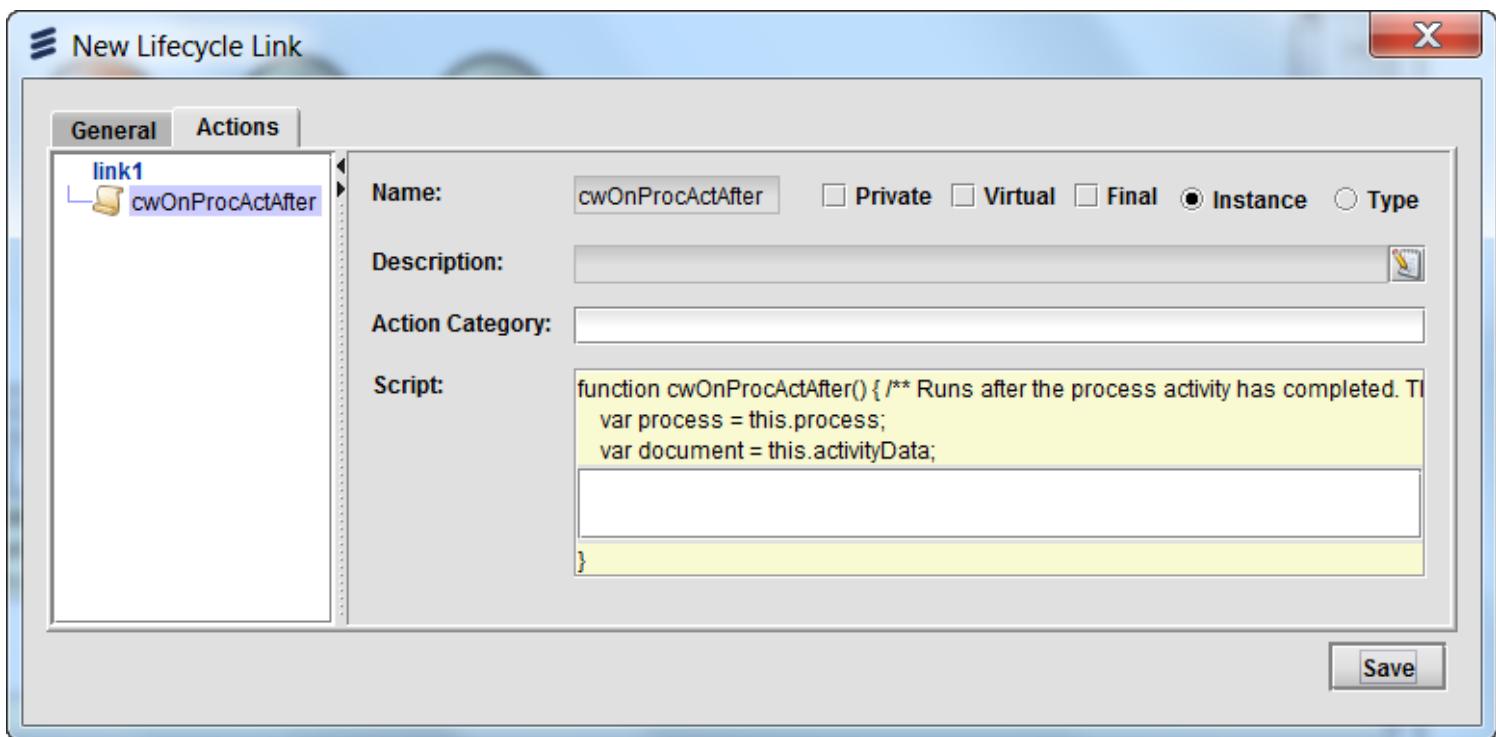
Multiple links are allowed to go out from one state to other states. The **Priority** field in the New Life Cycle link dialog allows you to assign a sequential number to the link. When more than one transition from the same state is possible, the one with the lowest priority number will be performed.

### Add Method for the Link

To add a method for the activity, click the Actions tab and right-click the state. You can select a predefined method or [create a new](#) one.



The Action property allows you to create or change the script.



The following table provides a description of the fields:

Field	Description
Name	Mandatory. Visible on the <b>General</b> tab. Unique name for the link.
Label	Mandatory. Visible on the <b>General</b> tab. Visible name for the link.
Description	Visible on the <b>General</b> tab. Descriptive text for documentation purposes.
Priority	Visible on the <b>General</b> tab. Sequential number that defines the transition order when multiple links go out from one state to other states. The default value is 0 and it is not shown. When more than one transition from the same state exists, the one with the lowest number will be performed. If several transitions have the same number the transition is unpredictable.
Action	Visible on the <b>Action</b> tab. Action script. The script will be executed if the transition specified by the link is executed.

<b>Script</b>	Visible on the <b>Action</b> tab. Condition script that will be used to decide which condition can be used for transition from the current to the next state. This happens when the state machine knows the next state (for example, the state is returned from a script) or when an Order with a Life Cycle diagram requests a change of state and needs to understand if it is allowed to transition to it.
---------------	---

The links with a script defined are shown in green color on the diagram. The links with no scripts defined are in blue color. If a link priority number is greater than 0, it will be shown as the link label on the diagram.

# Finders

Finders are used for searching, accessing, adding, updating and deleting data from different data sources such as databases and external systems. Information that is returned by the Finder object is displayed using the Finder User Interface. Contained within the Finder User Interface are several Forms that enable you to customize the display of the data at runtime. You can customize the default Finder Forms, by [extending](#) or [overriding](#) them. This enables you to control every aspect of the User Interface presentation, from the layout of the search screen to the object menu of the Finder.

Before defining a Finder, the user should determine the following:

- What the user is looking for. This will determine which [Finder type](#) to use.
- What are the criteria, if any, on which to search. This will define the search Document type for the select operation.
- What fields should be included in the search results. This will define the result Document type for the select operation.

## Types of Finders

---

The following types of Finders are supported:

- [Document Finders](#) that allow searching for a single Document type mapped to the database as a single table, set of database tables or database view.
- [Order Finders](#) that allow searching for Orders of one or many types stored in the database.
- [SQL Finders](#) that allow a database search with user-specified SQL statements.
- [Script Finders](#) that execute scripts to implement the requested operation.
- [Interface Finders](#) that link to external interfaces that implement the Finder operations.
- [Rule Finders](#) that allow the use of existing business rules for the Finder definition.

Finders in general perform select, get, add, update and delete operations. Different Finder types support subsets of these operations, but all Finders support the select operation.

Finder type	select	get	add	update	delete
Document	x	x	x	x	x
Order	x	x			
SQL	x	(Optional)			
Script	x	(Optional)			
Interface	x	(Optional)	(Optional)	(Optional)	(Optional)
Rule	x				

## Creating a New Finder

To create a new Finder, follow these steps:

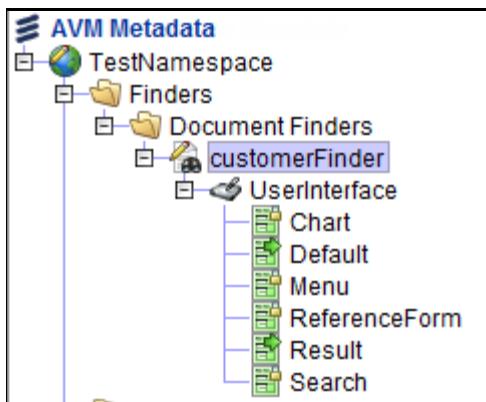
1. In the Navigation Tree, right-click the Namespace.
2. Select **New** from the dialog.
3. In the **New Metadata Object** wizard, expand the **Finders** node, and select one of the following Finders:
  - o **Document Finder**
  - o **Interface Finder**
  - o **Order Finder**
  - o **Rule Finder**
  - o **SQL Finder**
  - o **Script Finder**
4. Click the **Next** button.

## Additional Finder Fields

Depending on the type of Finder that you selected, there may be additional fields.

- All Finder names must conform to JavaScript naming conventions.
- For the Base Controller fields, you are recommended to select the `com.conceptwave.system.FinderUserInterface`; see [Finder User Interface and Forms](#) for details.
- Continue by configuring the Finder and then click **Finish**.
- **Document Finder**
  - **Name:** Type the name of Finder.
  - **Base Controller:** Select the data type.
  - **Select Output:** Choose the Output Document property of the Finder, in **General tab** when the Finder is created.
- **Interface Finder**
  - **Name:** Type the name of Finder.
  - **Base Controller:** Select the data type.
  - **Interface:** Choose the Interface property of the Finder, in **General tab** when the Finder is created.
- **Order Finder**
  - **Name:** Type the name of Finder.
  - **Base Controller:** Select the data type.
- **Rule Finder**
  - **Name:** Type the name of Finder.
  - **Base Controller:** Select the data type.
- **Script Finder**
  - **Name:** Type the name of Finder.
  - **Base Controller:** Select the data type.
- **SQL Finder**
  - **Name:** Type the name of Finder.
  - **Base Controller:** Select the data type.

The newly created Finder is added by type under the corresponding Finder folder, which is under the **Finders** folder.



**Note:** The Finder displays its Forms under the Finder User Interface object. You can customize your Finder Form, by [extending](#) or [overriding](#) the default Finder Forms. You can control the User Interface presentation for the following Forms: Chart, Default, Menu, ReferenceForm, Result and Search.

## Finder General Properties

The general Finder properties are defined on the **General** tab.

The screenshot shows the 'General' tab of the Finder configuration interface. The 'Name' field is set to 'testFinder'. The 'Label' field contains 'Test Finder'. The 'Description' field is empty. The 'Overrides' dropdown is set to '<NONE>'. The 'Volatility (min)' field is set to '60'. The 'Select input' and 'Select output' fields both show '<NONE>'. There are checkboxes for 'Private', 'Restricted', 'Deprecated', 'Virtual', and 'Final', none of which are checked. A toolbar at the top includes tabs for General, Search, View, Advanced, and Methods, along with several status icons.

The following table describes the common fields for the **General** tab of the Finder:

Field	Description
<b>Name</b>	Mandatory. Unique name of the Finder within the namespace (used in scripts; must conform to JavaScript naming conventions).
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Virtual</b>	If checked, it provides the ability to create methods in template metadata that does not have implementation, but can be used in template script and be bound to UI elements. The implementation is done in user metadata (see <a href="#">Virtual Field</a> for more information).
<b>Final</b>	Metadata elements can be marked as final, which includes any Boolean final property that is available on any metadata element. Additionally, any metadata element that has the <b>Final</b> property set to true can neither be extended, nor overridden.
<b>Label</b>	Mandatory. Display label to identify the Finder.
<b>Description</b>	Description of the Finder for documentation.
<b>Overrides</b>	Specifies an overridden Finder. For example, a Document Finder can override another Document Finder from a library when the output Document of the overriding Finder is the same or the output document from the base Finder is overridden.  <b>Note:</b> When specifying a finder to override, an input document must be inherited from the overridden finder.
<b>Volatility (min)</b>	When this Finder is used for a code table, specifies the update period in minutes. Setting the volatility to zero forces the system to read from the database each time that a search is performed.
<b>Select Input</b>	This field is optional for all Finder types except Order Finders. It is mandatory for Order Finders.  It is required when the Finder needs search criteria for the select operation. For Document and Rule Finders, the <b>Select input</b> Document field is used as a search criteria and must be mapped to the database. This field is the <b>Input document</b> when it becomes read-only based on Finder type. When used in the UI, the Finder search criteria are entered using the form specified in the <b>Search Form</b> field of the Finder's <a href="#">View tab</a> .
<b>Select Output</b>	Mandatory. Finders must always define a <b>Select output</b> Document. The result of the Finder select operation is a list of

runtime **Select output** Document instances that contain the data matching the search criteria. For Document and Rule Finders, the **Select output** Document must be mapped to the database. This field is **Output document** when it becomes read-only based on Finder type.

When used in the User Interface, the Finder results are displayed using the form specified in the **Result Form** field of the Finder's [Views tab](#).

## General Properties for Finder Types

The fields displayed below the **Select output** field are dependent upon the type of Finder with which you are working. The following sections describes the additional property fields in the **General** tab for each Finder type.

### Document Finder

A Document Finder is used for finding Documents mapped to the database. The Finder returns the **Select output** Document containing the Documents corresponding to the search criteria, if any.

**Note:** The General tab of the Document Finder does not have any additional fields.

The screenshot shows the 'General' tab of a Document Finder configuration screen. The tab bar includes 'General', 'Search', 'View', 'Advanced', and 'Methods'. The 'General' tab is active. The configuration fields are as follows:

Name:	testDocumentFinder	<input type="checkbox"/> Private	<input type="checkbox"/> Restricted	<input type="checkbox"/> Deprecated	<input type="checkbox"/> Virtual	<input type="checkbox"/> Final
Label:	Document Finder	...				
Description:	<input style="width: 15px; height: 15px;" type="button" value="..."/>					
Overrides:	<NONE>	<input style="width: 15px; height: 15px;" type="button" value="..."/>				
Volatility (min):	60					
Select input:	<NONE>	<input style="width: 15px; height: 15px;" type="button" value="..."/>				
Select output:	TestNS.finderdoc (Finder Doc)	<input style="width: 15px; height: 15px;" type="button" value="..."/>				

### Order Finder

An Order Finder must have a **Select input** Document defined and at runtime it must not be empty. This restricts the number of Orders found, because the select operation for Order Finders requires many resources and can affect the system performance negatively.

General Search View Advanced Methods

Name:	testOrderFinder	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated <input type="checkbox"/> Virtual <input type="checkbox"/> Final
Label:	Test Order Finder	...
Description:	<input type="text"/>	
Overrides:	<NONE>	...
Volatility (min):	60	
Select input:	<NONE>	...
Select output:	api_common.searchContext	...
<input type="checkbox"/> All documents must exist		
Order(s):	<input type="checkbox"/> Name <input type="checkbox"/> TestNS.testOrder	

The following table describes the additional fields for this Finder:

Field	Description
All documents must exist	If checked, all Documents that are used to construct the Finder result set have to be present to create the result Document instance. When unchecked, some Documents that are used to construct the result may be missing. When this is the case, the result Document is still created with the missing fields set to null. The default is unchecked.
Order(s)	Displays a list of Order types with a check box beside each type. When all the check boxes are unchecked, all Order types are included in the search. When one or more check box is checked, only the checked Order types are included in the search. The default is all check boxes are unchecked.

## Rule Finder

A Rule Finder is based on a Business Rule that must be first defined in the **Rules** folder under the **Business Processes** folder (Library > cwf\_pm (Process Management Product)). The expressions from rule instances of the specified Business Rule, defined on the Business Rule **Instances tab**, are used as search criteria on the Rule Finder invocation. In the Rule Finder metadata, this Business Rule must be selected in the **Rule** field. The **Input Document** and **Output Document** fields in the Rule Finder are read-only. The optional **Input Document** on the Rule **General** tab appears as the **Input Document** of the Finder. The **Rule document** that is defined on the Rule **General** tab appears as the **Output Document** of the Finder.

General View Advanced Methods

Name:	testRuleFinder	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated <input type="checkbox"/> Virtual <input type="checkbox"/> Final
Label:	Test Rule Finder	...
Description:	<input type="text"/>	
Overrides:	<NONE>	...
Volatility (min):	60	
Rule:	<NONE>	...
Input document:	<NONE>	...
Output document:	<NONE>	...

The following table describes the additional field for this Finder:

Field	Description
Rule	Mandatory. A Business Rule from the <b>Rules</b> folder under the <b>Business Processes</b> folder (Library > cwf_pm (Process Management Product)).

## Interface Finder

An Interface Finder is based on an Interface that must be first defined in the **Interfaces** folder under the **External Services** folder (Library > cwf). At creation of Interface Finder, the Interface is selected. The Interface Finder has additional optional fields **Select**, **Get**, **Add**, **Delete** and **Update** where you can select a list of Operations defined for the Interface. Each Operation may or may not have input and output Documents defined in the **Input** and **Output** fields on the **General** tab. If you select an Interface that only has Operations with no documents, the lists is empty.

The **Select** operation must be assigned. Only an Operation that contains an output Document can be selected. The input and output Documents of the **Select** operation appear as the **Input Document** and **Output Document** (read-only) of the Finder.

**Note:** Interface finders can only be used with interface operations that have Documents as inputs or outputs.

General	Search	View	Advanced	Methods
Name: testInterfaceFinder	<input type="checkbox"/> Private	<input type="checkbox"/> Restricted	<input type="checkbox"/> Deprecated	<input type="checkbox"/> Virtual
Label: Test Interface Finder	<input type="checkbox"/> ...			
Description: <input type="text"/>	<input type="checkbox"/>			
Overrides: <NONE> <input type="button" value="..."/>				
Volatility (min): 60				
Interface: <NONE> <input type="button" value="..."/>				
Select: <NONE> <input type="button" value="..."/>				
Get: <NONE> <input type="button" value="..."/>				
Add: <NONE> <input type="button" value="..."/>				
Delete: <NONE> <input type="button" value="..."/>				
Update: <NONE> <input type="button" value="..."/>				
Input document: <NONE> <input type="button" value="..."/>				
Output document: <NONE> <input type="button" value="..."/>				

The following table describes the additional fields for this Finder:

Field	Description
Interface	Mandatory. Interface type to use for Finder operations from the <b>Interfaces</b> folder under the <b>External Services</b> folder..
Select	Mandatory. Interface Operation for selecting (searching) results.
Get	Interface Operation used to get the detail Document.
Add	Interface Operation used to add a new Document.
Delete	Interface Operation used to delete a Document.
Update	Interface Operation used to update a Document.

## Script Finder

A Script Finder contains additional **Select** and **Get** fields. These fields allow you to specify Documents for select and get operations with additional

JavaScript support.>/

**Note:** Script Finders do not specify search criteria operations.

The screenshot shows the 'General' tab of a Script Finder configuration dialog. The fields include:

- Name: testScriptFinder
- Label: Test Script Finder
- Description: (empty)
- Overrides: <NONE>
- Volatility (min): 60
- Select input: <NONE>
- Select output: api\_common.data.object
- Get input: <NONE>
- Get output: <NONE>

Checkboxes for Private, Restricted, Deprecated, Virtual, and Final are present at the top right. A '...' button is also visible.

The following table describes the additional fields for this Finder:

Field	Description
Get input	Input Document for get operation.
Get output	Output Document for get operation.

Use the **Methods** tab to define the **Select** and **Get** scripts.

The screenshot shows the 'Methods' tab for defining a 'Select' script named 'cwOnFinderSel'. The script code is:

```
function cwOnFinderSel(objectList) { // Runs when 'select' operation is invoked
    var finder = this; // Deprecated
    // 'document' is the document containing the reference. Applicable for UI
    var document = this.refFinderDocument; // Deprecated
    var searchData = this.searchDocument; // The search criteria document.
```

## SQL Finder

A SQL Finder defines a database Finder that performs select and get operations by specifying select and get SQL statements.

**Note:** SQL Finders do not specify search criteria operations.

General View Advanced Methods

Name:	testSQLFinder	<input type="checkbox"/> Private	<input type="checkbox"/> Restricted	<input type="checkbox"/> Deprecated	<input type="checkbox"/> Virtual	<input type="checkbox"/> Final
Label:	Test SQL Finder	...				
Description:						
Overrides:	<NONE>					
Volatility (min):	60					
Select input:	<NONE>					
Select output:	api_common.searchContext					
Get input:	<NONE>					
Get output:	<NONE>					
Logical connection:	<NONE>					
						Connect

The following table describes the additional fields for this Finder.

Field	Description
Get input	Input Document for get operation.
Get output	Output Document for get operation.
Logical connection	Database schema. Use the <b>Connect</b> button to select the database schema.

Use the **Methods** tab to define the **Select** and **Get** scripts.

Name: cwOnFinderSQLSel

Description:

SQL:

```
select c.firstName, c.lastName from customer c;
```

## Select SQL statement

The select statement is written in SQL with additional syntax that defines the product-specific behavior. It consists of conditional and unconditional segments.

Conditional segments are enclosed in curly brackets and may contain parameter placeholders. The conditional segments are included in the runtime SQL statement only if the value of the corresponding search Document leaf is not null or if they are marked as always included (refer to <option> A below). The conditional segments contain designators in square brackets that define the properties of the segment (see below).

Unconditional segments contain the text between the conditional segments and are always included in the runtime SQL select statement. They should not contain parameter placeholders, although the system does not perform a check for parameter placeholders. For example:

```
select <unconditional segment> {<conditional segment>}<unconditional segment>{<conditional segment>}...
select <unconditional segment> {<conditional segment>}<conditional segment>{<conditional segment>}...
```

Immediately following the opening curly bracket, a conditional segment must specify the segment designator in square brackets. The general format of the segment designator is:

[<parameter number>,<test mask>,<set mask>,<options>] where:

- <parameter number> is a number from 0 to 127 or an asterisk.
- <test mask> is a positive integer number.
- <set mask> is a positive integer number.
- <option> is any combination of the uppercase letters I, E, A, P, N, B.

The segment designator should contain at least one part (not including the select statement). The <parameter number> must be defined first, while the remaining parts of the segment designator can be in any order. The missing parts of the designator do not need to be marked by commas.

The designator defines the Finder behaviour as follows:

- If the <parameter number> has a value from 0 to 127, the segment will be included only if the search Document leaf with the corresponding number has a non-null value.

◦ Example:

```
select ... {[3]...} ... {[0]...} ...
```

The first conditional segment will be included if the fourth field of the search Document is not null; the second segment will be included if the first field is not null.

- If the <parameter number> is \*, the segment is included only if the search Document has at least one non-empty leaf. Such segments cannot contain a parameter placeholder.

◦ Example:

```
select ... {[*]} where } ...
```

The segment will not be included if the whole search Document is empty.

- Each conditional segment may contain only one parameter placeholder. If it contains a parameter placeholder, the placeholder will be bound to the corresponding search Document leaf as specified by the <parameter number>.

◦ Example:

```
select ... {[3]} and T1.USERID = ?} ...
```

In runtime, the parameter placeholder will be bound to the value of the fourth field of the Document.

- A conditional segment allows you to use a parameter name along with the <parameter number>. It means that either a leaf position or a leaf name can be specified in the segment designator.

**Note:** Similar to the <parameter number>, the parameter name should be used as the first part, if it exists.

◦ Example:

```
select ... {[::address]...} ... {[::firstName]...} ...
```

The parameter name is recognized by the colon (:) prefix and converted into a leaf position at compilation time.

- The operation IN is recognized in the segment and the placeholder is properly replaced by a list of values only if the keyword in is preceded and followed by a space. In this case the placeholder should exist and should not be put in parentheses.

◦ Example:

```
select ... {[3]} and TABLE1.COLUMN2 in ?} ...
```

If the fourth parameter of the search Document is A,B the runtime SQL statement will be select ... and TABLE1.COLUMN2 in ('A','B') ...

- If the select SQL statement contains the string {+\*+}, it is replaced by the list of all columns mapped to DB.

◦ Example:

```
select {+*+} from t1 where TYPE = 3
```

- The segment may check bits of the internal mask using the <test mask> number. If the internal mask has at least one bit of the <test mask> set, the segment is included. The <test mask> is checked after all <set mask> operations are performed.

- o Example:

```
select ... {[T6] TABLE1.COLUMN IS NULL} ...
```

The segment will be included if the second or third bit of the internal mask is set.

- The SQL Finder maintains a 31 bit internal mask initially set to 0. Each segment can set bits of this mask by specifying a <set mask> number.

The segments that are skipped because of the null search Document leaves do not set the internal mask.

- o Example:

```
select ... {[12,S4] T1.C1 = ?} ...
```

If the thirteenth field of the search Document is not null, the third bit of the internal mask will be set to 1.

- If <option>I is specified in the designator, the segment must contain a parameter placeholder. In this case the value of the corresponding search Document leaf replaces the placeholder in the SQL statement. The value must be a number or string, if the value is a string it is delimited by single quotation marks.

- o Note: For backwards compatibility with AVM 3.0, the designator of format +n+ is supported and has the equivalent functionality to the segment designator [n,I].

- o Example: select ... {[0,I] T1.C1 = ?} ...

If the first field of the Document contains the string value ABC, the runtime SQL statement will be select ... T1.C1 = 'ABC' ...

- If <option>E is specified in the designator, the segment must contain a parameter placeholder. In this case the value of the corresponding search Document leaf replaces the placeholder in the SQL statement. The value must be a number or string, if the value is a string it is not delimited by single quotation marks. This option can be used to extend the SQL statement.

- o Example:

```
select ... {[5,E] order by ?} ...
```

If the sixth field of the search Document has the value COLUMN1, the runtime SQL statement will be select ... order by COLUMN1 ...

- If <option>A is specified in the designator, the segment is always present independently of the parameter value. Such a segment behaves as an unconditional segment except that it may have placeholder. Just as an unconditional segment, it may check the internal mask and can be skipped correspondingly.

- o Example:

```
select ... {[5,A] or TABLE1.COLUMN <> ?} ...
```

The segment will always be present and the placeholder will be bound to the value of the sixth field of the search Document.

- If <option>P is specified, the test mask operation checks for all bits set. If the internal mask has all bits of the <test mask> set, the segment is included.

- o Example:

```
select ... {[T9,P] TABLE1.COLUMN3 IS NOT NULL} ...
```

The segment will be included if the first and the fourth bits of the internal mask are set.

- If <option>N specifies that the segment will be replaced by the keyword null if the condition for its inclusion is not true. This is usually used in the column list of the select statement.

- o Example:

```
select TABLE1.C1, {[1,N]TABLE1.C2} ...
```

If the second field of the search Document is null, the runtime select statement will be select TABLE1.C1,null ...

- If <option>B is specified and the corresponding search Document field is Boolean, the segment will not be included if the field value is false.

- o Example:

```
select ... {[14,B] or TABLE1.COLUMN <> ?} ...
```

The conditional segment will not be included if the fifteenth field of the Document has a value of false or is null.

**Note:** The conditional segments should not contain the symbol ? except when used for the parameter placeholder. The select SQL statement should not contain the symbols '{' and '}' except as conditional segment delimiters.

## Referencing the DB mapped table

If the select SQL statement contains the string '{\*+}', it is replaced by the list of all columns mapped to the database.

For example:

```
| select {*} from t1 where TYPE = 3
```

## SQL statement example

The following is an SQL statement example:

```
| select TBL1.A,TBL1.B from TBL1 {[T1],TBL2} {[*]where} {[0,S2]TBL1.A > ?} {[T3,P]and} {[1,S1]TBL1.K = TBL2.K and TBL2.B < ?}
```

## Support for Conditional Clauses for Multiple Deployed Applications

The SQL finder contains conditional clause support. This support is based on your system configuration for multiple deployed applications.

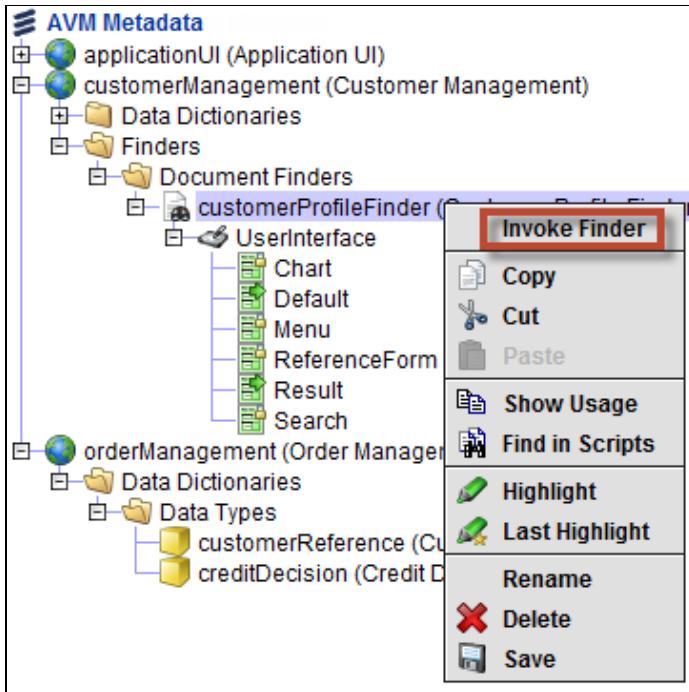
The conditional clause supports the \$multiApp system variable, indicating that multiple application deployment is configured on the system. The following is an example:

```
| SELECT {+*+} FROM DUAL {[:_multiApp]} WHERE APP_NAME = ?
```

## Invoke a Finder

Using the Invoke Finder dialog, you can test the functionality of your finder to ensure that it is configured correctly. To invoke your finder, complete these steps:

1. Right-click your finder and select **Invoke Finder** from the menu.



2. The Invoke Finder dialog appears, which allows you to specify your search criteria and invoke the finder. The top pane essentially contains one input field for each field in the **Select input** document defined on the **General** tab. You can either enter your search criteria values manually, or you can import these values from either a text or XML file.

Invoke Finder 'Customer Profile Finder'

**Input: Customer Profile (Customer Management)**

Cusomer Name:	<input type="text"/>
Market Segment:	<input type="text"/>
Is Preferred:	<input type="text"/>

**Output: Customer Profile (Customer Management)**

Cusomer Name	Market Segment	Is Preferred	System document ID

Export into:  View  File

**Buttons:**

- Validate
- Import Text
- Import XML
- Export XML
- Reset
- View
- Export Text
- Export XML
- Invoke
- Close

You can clear your search criteria fields at anytime by clicking the **Reset** button.

3. To export or import your search criteria from either a text or XML file, you can do one of the following:
  - o Click the **Export XML** button to export the defined search criteria values to an XML file for later use.
  - o Click the **Import Text** button to import data from a text file. The Import Text dialog opens and allows you to navigate to where your data is located. By clicking the **Open** button, the Import Settings dialog opens where the text file settings can be defined for the fields that are enabled. These settings determine how the file is processed.
  - o Click the **Import XML** button to import your data from an XML file. The Import Document from XML dialog opens to where the file is selected.
4. Before invoking the search, click the **Validate** button to validate the search criteria values. If there are any errors, a **Validation Errors** dialog opens.
5. After you have entered your search criteria and have validated them, click the **Invoke** button to invoke the search.
6. Your search results appear in the bottom section of the Invoke Finder dialog. There is one column for each field in the **Select output** document defined on the **General** tab. The results appear as a list of document instances. You can preview the result documents by highlighting a row and clicking the **View** button.

Invoke Finder 'Customer Profile Finder'

Input: Customer Profile (Customer Management)				
customerName:	<input type="text"/>			
marketSegment:	<input type="text"/>			
Is Preferred:	<input type="text"/>			
<input type="button" value="Validate"/> <input type="button" value="Import Text"/> <input type="button" value="Import XML"/> <input type="button" value="Export XML"/> <input type="button" value="Reset"/>				
Output: Customer Profile (Customer Management)				
customerName	marketSegment	Is Preferred	cwDocId	
Steve	Residential	<input checked="" type="checkbox"/>	204000	
Adam	Commercial	<input checked="" type="checkbox"/>	203000	
Eve	Residential	<input checked="" type="checkbox"/>	203001	
Harry	Residential	<input checked="" type="checkbox"/>	205000	
<input type="radio"/> View <input type="radio"/> File		<input type="button" value="View"/>	<input type="button" value="Export Text"/>	<input type="button" value="Export XML"/>
		<input type="button" value="Invoke"/>	<input type="button" value="Close"/>	

7. You can export your results to a file. To preview the data and file structure before exporting, select the **View** radio button, and click either the **Export Text** or **Export XML** button depending on the export format. The **Export Data Preview** dialog opens.
8. Once you are satisfied with the file structure, select the **File** radio button, and click either the **Export Text** or **Export XML** button to export the document data into a text or XML file.

**Note:** Only one record can be exported at a time, each to a separate file.

## Finder Search

The **Search** tab is used for defining mappings between search criteria and result Documents fields and the fields of the Documents that keep the search and result data in the database. The purpose of the mappings is to handle different scenarios and simplify the user's work. For example, a single input and output Document can be used to search Orders. This is done by mapping the fields in the input and output Documents to the corresponding fields in each search target. If either the input or output Document is the one that is stored in the database, and is used to hold the Order Entry data, the mapping is circular in nature from the Document to itself.

### Note:

- The **Search** tab is not available in Rule Finders or the SQL Finders. The mapping of search Document to result Document is mandatory for Order and Document Finders.
- In the result Document mapping for Order Finders, the field that is a generated key in the result Document should not be mapped.

The screenshot shows the 'General' tab of the Finder Search interface. On the left, there is a tree view of namespaces and documents. The 'cw\_l\_worklist' namespace is expanded, showing its contents. The 'Documents' folder under it contains several items like 'actIndex (Activity Index)', 'appName', 'AssignedDate (Assigned)', etc. Below this, there are sections for 'Effort', 'Flags (Task Type)', 'MetadataVer (Metadata Version)', 'Operation (Task)', 'Orderid (Order ID)', 'orderitemid (Order item ID)', 'OrderK (Details)', 'Participant (Participant type)', 'Priority', 'revisionNo (Process Revision)', 'SenderId (Process ID)', 'SenderType (Sender Type)', 'StartDate (Started)', and 'userid (User ID)'. The right side of the interface is divided into three main sections: 'Search criteria items', 'User Defined Search items', and 'Sort'. The 'Search criteria items' section contains a table with columns 'Item', 'Operation', 'Map To', and 'No Case'. It has three rows: 'Flags (Task Type)' with operation '=', 'Map To' 'Flags (Task Type)', 'disable' with operation '=', 'Map To' 'disable', and 'userid (User ID)' with operation '=', 'Map To' 'userid (User ID)'. Below this is a row of buttons: 'Unmap', 'Map All', and 'Map'. The 'User Defined Search items' section contains a table with columns 'Item', 'Operation', and 'Static Value'. It has one row: 'Flags (Task Type)' with operation '==' and static value '2'. Below this is a row of buttons: 'Add' and 'Remove'. The 'Sort' section contains a table with columns 'Item' and 'Descending'. It has one row: 'createDate' with 'Descending' checked. Below this is a row of buttons: 'Add' and 'Remove'.

The **Search** tab is divided into these sections:

- **Document tree:** Shows all the Documents that can be used for mapping within their namespaces in a tree structure. Each **Documents** folder and **Orders** folder can be expanded to show the available Documents and its variables.
- **Search criteria items \*table:** Shows the list of fields in the **Select input** Document defined on the **General** tab.
- **Search result items table:** only in **Order Finder**. Shows the list of fields in the **Select output** Document defined on the **General** tab.
- **User-Defined Search items** table: Allows you to search for a null or static value.
- **Sort table:** Specifies the sort order of the results.

**Note:** All items on the **Search** tab appear in alphabetical order.

The following table describes each column in the **Search criteria items** section:

Field	Description	Assigned by
Item	Items in the <b>Search criteria items</b> table are the variables of the <b>Select input</b> Document defined on the <b>General</b> tab.	System
Operation	The usage of the search criteria item in the search. A value can be selected from the combo box by clicking in the field. The choices include the following: <ul style="list-style-type: none"><li>• equal</li><li>• not equal</li><li>• greater than</li><li>• greater than or equal</li><li>• less than</li><li>• less than or equal</li><li>• contains</li><li>• ends with</li><li>• starts with</li></ul>	User

	<ul style="list-style-type: none"> <li>• like</li> <li>• in</li> <li>• not in</li> </ul> <p>The field can also be left blank. In this case, the corresponding row is ignored.</p>	
<b>Map To</b>	The Document field to which the search criteria item or search result item is mapped.	User
<b>No Case</b>	This field allows you to turn off case-sensitivity. This checkbox can only be selected for the following operations:	User
	<ul style="list-style-type: none"> <li>• Contains</li> <li>• Starts with</li> <li>• Like</li> <li>• In</li> <li>• Not in</li> </ul>	

The following table describes each column in the **Search result items** section:

Field	Description	Assigned by
<b>Item</b>	Items in the <b>Search result items</b> table are the variables of the <b>Select output</b> Document defined on the <b>General</b> tab.	System
<b>Map To</b>	The Document field to which the search criteria item or search result item is mapped.	User

The **Search result items** section appears in Order finders only:

<b>Search result items:</b>	
Item	Map To
BAN	
Date of Birth	Date of Birth
firstName	First Name
lastName	Last Name

The following table describes each column in the **User-Defined Search Items** section:

Field	Description	Assigned by
<b>Item</b>	Items in the <b>User-Defined Search Items</b> table are the variables of the <b>Select output</b> Document defined on the <b>General</b> tab.	System
<b>Operation</b>	The usage of the search criteria item in the search. A value can be selected from the combo box by clicking in the field. The choices include the following: <ul style="list-style-type: none"> <li>• is null</li> <li>• is not null</li> <li>• == (equal to static value)</li> </ul>	User
<b>Static Value</b>	This field allows you to set a static value for your item (for example, the variableName item == 2, where 2 is set in the <b>Static Value</b> field).	User

The following are XML examples after you have defined your search criteria and verify your XML:

**Example:** Static value

```
<search type="sitem">
  <mappedDocument>doc_cwf_oe.CWPROCESSEVENTLOG</mappedDocument>
  <mappedPath>doc_cwf_oe.CWPROCESSEVENTLOG/leaf_NodeId</mappedPath>
  <operation>==</operation>
  <staticValue>UI</staticValue>
</search>
is not null example -
<search type="sitem">
  <mappedDocument>doc_cwf_oe.CWPROCESSEVENTLOG</mappedDocument>
  <mappedPath>doc_cwf_oe.CWPROCESSEVENTLOG/leaf_STACK_TRACE</mappedPath>
  <operation>is not null</operation>
</search>
```

**Example:** is not null operator

```
<search type="sitem">
  <mappedDocument>doc_cwf_oe.CWPROCESSEVENTLOG</mappedDocument>
  <mappedPath>doc_cwf_oe.CWPROCESSEVENTLOG/leaf_STACK_TRACE</mappedPath>
  <operation>is not null</operation>
</search>
```

**Note:** Document and path elements are no longer mandatory. The staticValue element is only used for static values.

The **Unmap**, **Map All**, and **Map** buttons appear below the **Search criteria items** and **Search result items** tables depending on the Finder type selected in the **General** tab. When an items table is empty it means that part of mapping is not applicable for the selected type of Finder. When the buttons do not appear but the items table contains fields, it means the default mapping is always used.

Field	Description
<b>Unmap</b>	The <b>Unmap</b> button is used to unmap fields individually. To unmap fields, highlight the field in the table and click the <b>Unmap</b> button. The field in the <b>Map To</b> column is removed.
<b>Map All</b>	The <b>Map All</b> button is used to map all identical fields simultaneously. The identical fields are based on the variable's data type. To map all fields in a single step, highlight the Document in the browser and click the <b>Map All</b> button. All fields are added under the <b>Map To</b> column in the same order as items in the <b>Item</b> column.
<b>Map</b>	The <b>Map</b> button is used to map fields one by one. To map fields one by one, highlight one field in the table, and another one in the Document from the tree, then click the <b>Map</b> button. The field from the Document are added into the <b>Map To</b> column.

The sort order of the results is defined in the **Sort** table. To define the fields to sort the results by, click the **Add** button. The **Add Fields** dialog opens where the fields from the result Document are displayed. For each Sort field, the sort order can be specified in the **Descending** column. If the check box option is selected, the sort order is descending; otherwise the sort order is ascending. The order in which the results are sorted depends on the order of the fields in the **Sort** table. The first field is the first sort column; the second field is the second sort column. Use the up and down arrows, located to the right of the **Sort** table, to move the highlighted field.

## Finder View

Finder views are defined on the **Views** tab. A view specifies one distinctive way to represent a Finder in the User Interface. Each view is defined by the Finder Result Form specified in the view.

At runtime, the Finder views has the following behaviour:

- Upon initial display, the default view is displayed in the User Interface.
- Views are selected (or evaluated, when the view is a single form), based on the privileges of the view form.
- The Finder displays an error for a message Security violation. Views are not permitted if no Result Form is available, or if the Finder has a search Document. In this case, the **Show Search Form** property is checked for the view and no search form is available.
- The view menu of the Finder lists all views available for the user. The user can change the view by selecting a different view from the menu. If there is only one view it is not listed.

Name	Search Form	Result Form	Auto Search	Show Search
testDocFinderView	Default	Result	<input type="checkbox"/>	<input type="checkbox"/>

## Configuration

The **Search Form** and **Result Form** displays forms that appears in the Input and Output Document's User Interface, respectively. Overridden or extended Forms must be available or you cannot select a Form from either of these lists.

The following checkboxes appear on the View tab:

Field	Description
Allow custom views	This checkbox defines whether runtime custom views are allowed to be defined for this Finder. The runtime custom views are created by the user and stored in the database. They store information about the form that is used, the fields that are displayed and the content of the search criteria Document.  <b>Note:</b> At runtime, all Custom View Names should be unique for all views for a given user.  Users can share runtime custom views in the database if a view is defined as a shared view. This means that one user can create a view that can be used by other users (by default, the runtime views are private; they can be seen and used only by the user who created them).
Share	This checkbox defines whether the custom views can be shared. When checked, this option may create a serious security issue if the

<b>Views</b>	Finder has search criteria that are automatically populated with user-specific data such as user ID, and user profile data. These Finder views should not be shared and the <b>Share Views</b> property should be unchecked.
<b>Allow saving search criteria</b>	This field is set to true by default for backward compatibility. When selected, this setting allows you to save your search criteria, particularly if your finder either uses predefined search criteria or criteria set by a script from creating favorites or custom views with a saved search.

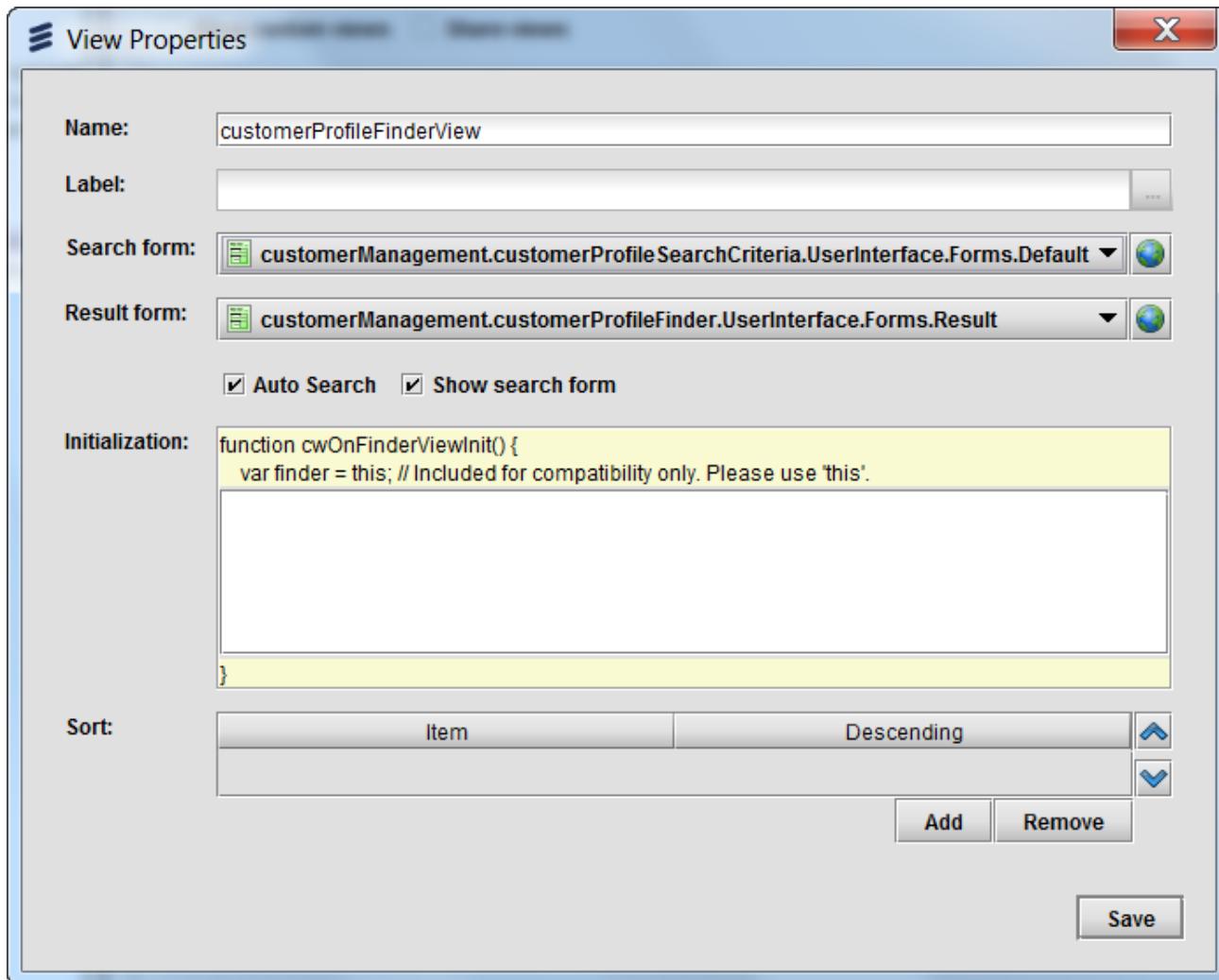
## Create a Finder View

To create a Finder view, follow these steps:

1. From the Finder View tab, click the **Add** button. The **New Finder View** wizard opens.
2. Type the **Name** of the Finder View.
3. Configure the properties of the new View.
4. Click **Finish**. The new View is then added to the Views list.

To update a view, highlight the view item in the list, then double-click it or click the **Properties** button. The Finder View Properties dialog opens. Continue by changing the view properties.

To delete a view, highlight the desired view item in the list, and click the **Remove** button.



The View Properties dialog contains the following fields:

Field	Description
<b>Name</b>	Mandatory. The name of the view.
<b>Label</b>	Mandatory. The label of the view to display at runtime.
<b>Search UI</b>	This field allows you to specify the user interface from which to get the search. This field lists all top-level user interfaces, the finder

user interface, and the input document user interface.

If this field is null, the forms that exist under the finder's input document populate the **Search form** field. Otherwise, the forms that exist under the selected UI populate the **Search form** field.

**Note:** Any non-null selection in the **Search UI** field nullifies the value in the **Search event** field.

To select a local form, do the following:

1. Find and select the finder's user interface in the **Search UI** field.
2. The **Search form** displays the finder's local forms, except for these forms:
  - o Default
  - o ReferenceForm
  - o Result
  - o Search

**Note:** The base finder's Search form is used in navigation trees and it displays the search form of the current finder view. This form cannot be used as a search form to avoid infinite recursion.

To select a top-level UI form, select the top-level UI from the **Search UI** field first, and then select the **Search form** from the list.

**Notes:**

- Helpful finder user interface variables include the following:
  - o `this.search`  
This variable keeps the search UI instance. It can be equal to `this` if the local form is used.
  - o `this.searchForm`  
This variable is the string name of the current search form
  - o `this.searchDoc`  
This variable is the current search document or data structure. Use the variable to build finder's local forms.
- In the case of a top-level form when the search UI is created, the input document or data structure of the finder is passed as a model to the top-level UI. To build a search form, override the `model` variable and set the type to the input object.
- In case of local form of the finder, by default, all search fields are disabled for script and SQL finders. By default, the `editablePerm` method of the script or SQL finder returns FALSE (that is, not-editable). To fix this case, you can either set the **Editable** property to **TRUE** on the field level or override the finder's `editablePerm` method and return TRUE.

<b>Search form</b>	Optional. The Form that appears in the search input section of the Finder Form. Choose one of the User Interfaces available in <b>Select Input Document</b> in General tab (or choose <NONE>).
<b>Search event</b>	This field allows you to select the search event for your finder. Any non-null selection in the <b>Search UI</b> field nullifies the value in the <b>Search event</b> field.  See the <b>Note</b> section that follows regarding precedence if both the form and event are specified.
<b>Result UI</b>	This field allows you to specify the user interface from which to get the result.  If this field is null, the finder's own forms populate the <b>Result form</b> field. Otherwise, the forms that exist under the selected UI populate the <b>Result form</b> field.  <b>Note:</b> Any non-null selection in the <b>Result UI</b> field nullifies the value in the <b>Result event</b> field.
<b>Result form</b>	Mandatory. The Form that appears in the search result section of the Finder Form. Choose from one of the User Interfaces available in this Finder. The default is its <b>Result Form</b> (<This Finder>.UserInterface.Forms.Result), which has a Table that displays all the returned results after the Finder has completed its search.  <b>Notes:</b> <ul style="list-style-type: none"><li>• You can change this field to show results in your own Form, but your Form must be located within this Finder's User Interface (that is, the Form should not be located in output Document).</li><li>• See the <b>Note</b> section that follows regarding precedence if both the form and event are specified.</li></ul>
<b>Result event</b>	This field allows you to select the result event for your finder. Any non-null selection in the <b>Result UI</b> field nullifies the value in the <b>Result event</b> field.

	See the <b>Note</b> section that follows regarding precedence if both the form and event are specified.
<b>Auto Search</b>	If checked, the search operation is performed before the view is shown for the first time. The value is displayed in the <b>Auto Search</b> column in the <b>View</b> list on the <b>Views</b> tab.
<b>Show search form</b>	If checked, the search form will be initially displayed when the view is shown for the first time. The value is displayed in the <b>Show Search</b> column in the <b>View</b> list on the <b>Views</b> tab.
<b>Initialization</b>	JavaScript script that is performed before the view is shown for first time.
<b>Sort</b>	List that defines how the result records are sorted.

**Note:** The **Search form** and **Result form** properties always takes precedence if both the form and event are specified. When only event is specified, the following actions can occur:

- The event handler can return the full name of the user interface to use and, in the user interface, have an initialization script for the `resultForm` variable.
- The event handler can return the full name of the user interface to use and the name of the form. The following is an example:

```
return "cwa_admin.eventLogResult/ResultEditable"
```

Where `ResultEditable` is the name of the form to use under `eventLogResult`.

## Finder Advanced Properties

Additional Finder properties are defined on the **Advanced** tab and enable you to customize the Finder configuration.

General    Search    View    Advanced    Methods

Help:

Max rows: 9999    UI query timeout (seconds): 90

Detail form: <NONE>

NLS\_COMP: <NONE>    NLS\_SORT:

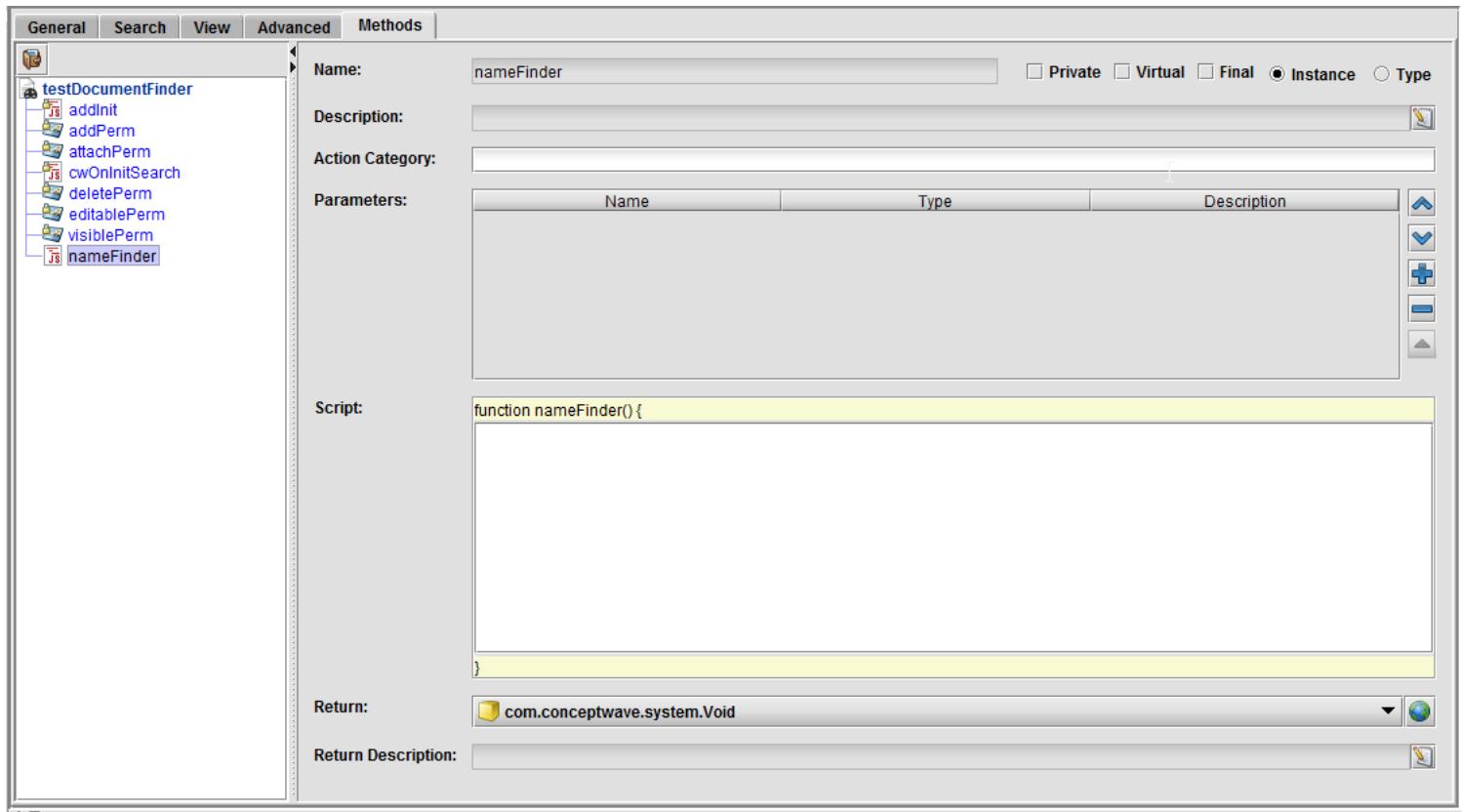
Auto-map values from parent when used in a reference field

Field	Description
Help	Ability to configure context sensitive user help that is available at runtime.
Max rows	Maximum number of rows to fetch (refer to <a href="#">Customizing the Worklist</a> for Worklist Finder search behaviour). The range of this field is from 100 to 9999. If this field is not set (by default) or the number set is smaller than 100, 100 is taken as the maximum number of rows to fetch.
UI query timeout (seconds)	Maximum number of seconds to wait for the Finder query to return results. The default is 90 seconds.
Detail Form	<p>Specifies the Form of the output Document to display when a result Detail is to be shown. Choose one of the User Interfaces available in <b>Select Output Document</b> in General tab (or choose &lt;NONE&gt;).</p> <p><b>Note:</b> Make sure that you do have overridden or local Forms in your output Document. Otherwise, only the generic base Forms inherited from the base Document of your output Document are available here (for example, <i>com.conceptwave.system.DocumentUserInterface.Forms.Default</i>).</p>
NLS_COMP	<p>This property supports Oracle linguistic sorting (for example, sorting customer names that are written in both uppercase and lowercase letters, and require being sorted in a list on the UI) and specifies the collation behaviour of the database session. An example of Oracle linguistic sorting is as follows:</p> <div style="border: 1px dashed #ccc; padding: 5px; margin-top: 10px;">ALTER SESSION SET NLS_COMP=LINGUISTIC;</div> <p>This property is available for SQL, document, and order finders. Click the drop-down menu and select from one of the following values:</p> <ul style="list-style-type: none"><li><b>BINARY</b> Comparisons in the WHERE clause and in SQL blocks are binary unless you specify the <b>NLS_SORT</b> parameter.</li><li><b>LINGUISTIC</b> Comparisons for all SQL operations in the WHERE clause and SQL blocks should use the linguistic sort specified in the <b>NLS_SORT</b> parameter.</li><li><b>ANSI</b> This setting is for backwards compatibility. In general, this property is set to <b>LINGUISTIC</b>.</li></ul>
NLS_SORT	<p>This property supports Oracle linguistic sorting (for example, sorting customer names that are written in both uppercase and lowercase letters, and require being sorted in a list on the UI). This property denotes the original value of the Oracle database collating sequence for ORDER BY queries (for example, BINARY_CI). An example of Oracle linguistic sorting is as follows:</p> <div style="border: 1px dashed #ccc; padding: 5px; margin-top: 10px;">ALTER SESSION SET NLS_SORT=BINARY;</div> <p><b>Notes:</b></p>

	<p>If the value is BINARY, the collating sequence for ORDER BY queries is based on the numeric value of characters (that is, a binary sort that requires less system overhead).</p> <ul style="list-style-type: none"> <li>• If the value is a named linguistic sort, sorting is based on the order of the defined linguistic sort. Most languages supported by the NLS_LANGUAGE parameter also support a linguistic sort with the same name.</li> <li>• Setting NLS_SORT to anything other than BINARY causes a sort to use a full table scan, regardless of the path chosen by the optimizer. BINARY is the exception because indexes are built according to a binary order of keys.</li> <li>• For a document finder, linguistic sorting from the database works.</li> <li>• The linguistic sorting on the UI (in a browser), when you click a column or define a custom view, is based on the values selected in the Nls_Comp and Nls_Sort properties.</li> </ul>
<b>Auto-map values from parent when used in a reference field</b>	This checkbox serves as a flag within a finder that allows control over whether automatic mapping of the finder search document data is done in a reference finder. By default, this property is checked. When this checkbox is selected, reference finders have their search document value filled with the parent finder's search document value if 1. They are the same type of document 2. A conversion map exists for the two types of documents.

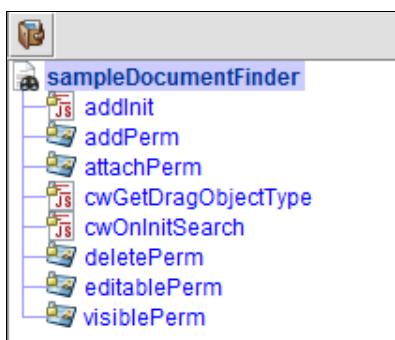
## Finder Methods

The **Methods** tab centralizes all of the Finder methods into one tab. You can define different types of methods for the Finder.

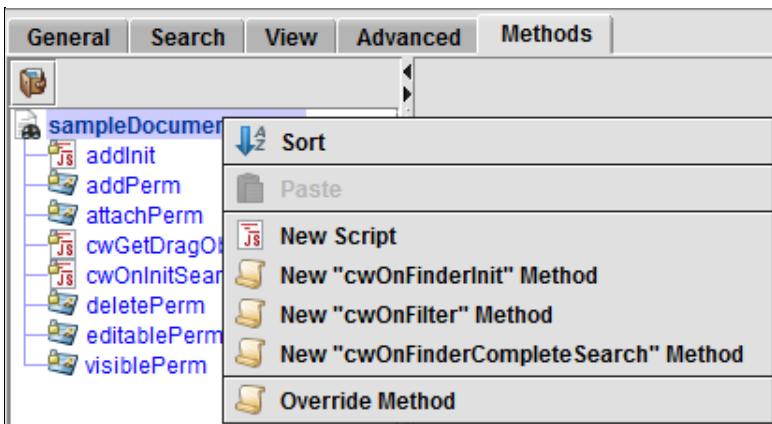


The Methods tab is a [script management screen](#) in Velocity Studio, which is composed of the **Method List pane** at the left which lists all scripts of the Finder, and the **Details pane** which displays the details of a script when selected at the Method List pane.

To show all base methods, click the **Show base methods** (  ) button. Click the button again to hide all base methods. Base methods appear in blue font, as shown in the following figure. Any method highlighted in blue indicates that you can override it. Methods in black font cannot be overridden.



To create a new Method, right-click the Finder in the Method List pane and select a method from the pop-up menu.



When you select the **Type** radio button, it means that the method can be invoked directly through the script and does not require an instance object.

If you have a long list of methods in the left pane, you can press the first letter of the method you are looking for to jump to the list of methods beginning with that letter. For example, pressing the I key highlights the first instance of a method that begins with the letter I. The method displays in the right pane.

This list represents the New Methods available that can be added at the Finder level. Please refer to the function headers for comments on input parameters and return types.

Method Type	Script Name	Description
<b>General Method</b>	user specified.	Generic user-defined JavaScript function that can be defined in a Finder and explicitly invoked by other scripts; the function is not triggered by other built-in means in the Finder.
<b>Initialization</b>	<i>cwOnFinderInit</i>	Finder Initialization script to set initial values for the Finder. It is triggered when the Finder is created.
<b>Result Filter</b>	<i>cwOnFilter</i>	Runs the script on each result row and removes it if the script returns false. A dependency exists between the <i>cwOnFilter</i> and <i>cwOnFinderCompleteSearch</i> scripts. The <i>cwOnFilter</i> is only run if the <i>cwOnFinderCompleteSearch</i> script exists.
<b>Complete Search</b>	<i>cwOnFinderCompleteSearch</i>	Runs when the finder search operation is completed. A dependency exists between the <i>cwOnFilter</i> and <i>cwOnFinderCompleteSearch</i> scripts. The <i>cwOnFilter</i> is only run if the <i>cwOnFinderCompleteSearch</i> script exists.
<b>Get script</b>	<i>cwOnFinderGet</i>	<i>Available only in Script Finder.</i> Script that is executed when get operation (that is, get from result documents) is invoked.
<b>Select script</b>	<i>cwOnFinderSel</i>	<i>Available only in Script Finder.</i> Script that is executed when select operation (that is, search) is invoked.
<b>Get SQL</b>	<i>cwOnFinderSQLGet</i>	<i>Available only in SQL Finder.</i> SQL statement that is executed when get operation (that is, get from result documents) is invoked.
<b>Select SQL</b>	<i>cwOnFinderSQLSel</i>	<i>Available only in SQL Finder.</i> SQL statement that is executed when select operation (that is, search) is invoked.

The list represents the system-defined Methods available. By default, the base Finder's equivalent method is invoked if any of these Methods is not overridden (that is, *cw\$super\_<method name>()*).

Method Type	Script Name	Description
<b>Add Permission</b>	<i>addPerm</i>	Add Permission that determines whether the participant is allowed to create a new instance of the output Document (using Detail section of the Finder <b>Default Form</b> ). Controls the visibility of Add button in object <b>Menu Form</b> .
<b>Attach Permission</b>	<i>attachPerm</i>	Attach Permission that determines whether attachments are allowed to be attached to the output Documents.
<b>Delete Permission</b>	<i>deletePerm</i>	Delete Permission that determines whether the participant is allowed to delete an instance of the output Document from search result (using Detail section of the Finder <b>Default Form</b> ). Controls the visibility of Delete button in object <b>Menu Form</b> .
<b>Get Drag</b>	<i>cwGetDragObjectType</i>	When this method is called on a finder, the finder will return the output document's type. The purpose of

<b>Object Type</b>		this method is to allow metadata to provide a different type for the object than what is used by the product.
<b>Edit Permission</b>	<i>editablePerm</i>	<p>Editable Permission that determines whether the participant is allowed to edit an instance of the output Document from search result (using Detail section of the Finder <b>Default Form</b>). Controls the visibility of Copy and Update buttons in object <b>Menu Form</b>.</p> <p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.</p>
<b>To String</b>	<i>toString</i>	<i>Reserved. Do not use.</i>
<b>View Permission</b>	<i>visiblePerm</i>	View Permission that determines whether the participant is allowed to view the Finder. Controls the visibility of the whole Section Stack of the Finder.

For the general Methods, you can create multiple Methods in the same Finder, each identifiable by unique Method name. However, for other system-defined Methods, you can only define one Method each. For scripts that are already defined, they appear in the Method List pane and no longer be available at the right-click context pop-up menu. The Method created in Method List pane can be right-clicked, with various commands available. For more information see [script management UI](#).

## Finder User Interface and Forms

The Finder User Interface behaves similarly to other User Interface metadata objects such as Documents and Orders. When configuring Finders, there are three system-defined User Interfaces to choose from and each of these User Interfaces has a set of default Forms.

User Interface	Description	Forms
<code>com.conceptwave.system.FinderUserInterface</code>	The generic Finder presentation, which has a <i>List View</i> that presents the Input Document as search border on top (if defined) and Output Document as search results in a Table. See <a href="#">Finder Variables</a> and <a href="#">Finder Methods</a> for more information.	<a href="#">Chart</a> , <a href="#">Default</a> , <a href="#">Menu</a> , <a href="#">ReferenceForm</a> , <a href="#">Result</a> , <a href="#">Search</a>
<code>com.conceptwave.system.OrderFinderUserInterface</code>	<p>This User Interface is similar to the <code>com.conceptwave.system.FinderUserInterface</code>, but the Output Document displayed is an Order Document. Using the Order Document ID, all associated orders (with the same Document ID) are displayed in the Detail View.</p> <p>For example, by configuring the <code>showDetailAction</code> method you can determine which Order Documents are called and sets it to display in the Detail View of the User Interface.</p> <pre>var order = Order.getOrderById(this.result.selected[0].cwDocId) this.detail = new order.metadata.UserInterface(order, this);</pre>	<a href="#">Chart</a> , <a href="#">Default</a> , <a href="#">Menu</a> , <a href="#">ReferenceForm</a> , <a href="#">Result</a> , <a href="#">Search</a>
<code>com.conceptwave.system.SplitFinderUserInterface</code>	<p>Similar to <code>com.conceptwave.system.FinderUserInterface</code>, but in the Detail View, the Output Document Form is displayed at the bottom in a separate pane, with the Finder's Search Border and result Table still displaying on top.</p> <p>This User Interface does not have the following Forms: Chart, ReferenceForm, Search.</p>	Default, Menu, Result

Clicking the finder's UserInterface in the left tree shows the following information in the General tab:

The screenshot shows the 'General' tab selected in a configuration dialog. The fields are as follows:

- Name:** UserInterface
- Extends:** com.conceptwave.system.FinderUserInterface (Finder User Interface)
- Overrides:** <NONE>
- Help:** Test help
- Timer duration (minutes):** (empty input field)
- Validate On Update:**

Form	Description
<b>Name</b>	Name of the finder's UserInterface.
<b>Extends</b>	The user interface that the finder extends.
<b>Overrides</b>	The default value is NONE. Click the drop-down menu and select the user interface to override.
<b>Help</b>	Enter help text for the finder, if required.
<b>Timer duration (minutes)</b>	Specify the number of minutes that this finder times out in this field.
<b>Validate on update</b>	By default, this checkbox is selected, meaning that the finder functions as usual. When this property is unchecked, the product only calls the save method on the metadata document without validating.

**Note:** The finder result is only the data document and does not have controller associated with it.

## Extend a Finder User Interface and Invoke Its Methods

You can extend a finder User Interface and invoke its methods. The following is an example:

1. Create a User Interface that extends com.conceptwave.system.FinderUserInterface and add a permission method in this User Interface.
2. Create a finder and extend its User Interface to the User Interface that you created in the previous step.
3. Set the **Visible** property of one of the User Interface elements to the permission method in step 1.
4. The permission method is invoked when displaying the finder User Interface.

## Finder User Interface Forms

By default there are several Forms that display in the Finder User Interface:

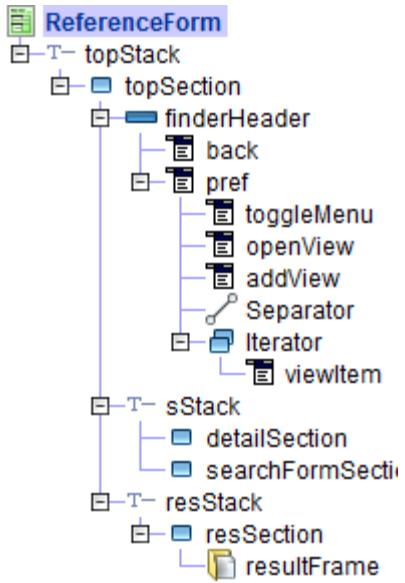
Form	Description
<a href="#">Chart</a>	Defines different chart types that can be used for graphical presentations of the Finder data.
<a href="#">Default</a>	This Form is the main form that renders the Finder and includes the Search, Result, Detail and Chart sections that are shown or hidden appropriately based on user's navigation at the Finder. It also displays the preference menu items.
<a href="#">Menu</a>	Defines the object menu options such as Search , Select, Add, Copy, Update, Delete.
<a href="#">Search</a>	Defines the appearance of the Search at runtime. It includes a Form Frame that displays the content of the search.
<a href="#">Result</a>	Defines the appearance of the Results at runtime. It includes a table form element that displays the content of the result.

You can define the layout of the Finder, using the Forms and their [extend](#) and [override](#) functions. For example, you can add additional menu items or alter existing functionality (show results in other Form Elements other than Table).

When associating a Finder into your application User Interface, create the [User Action Method](#) at the User Interface of the application Page or Form, and associate that method using a Form Element such as [Menu](#) or [Button](#).

## Finder ReferenceForm Form

The Reference Form element uses the **ReferenceForm** to display data. When a user clicks the Reference element icon for a Finder in the User Interface, the *onRefClick* method is triggered and defines the ReferenceForm to be used in the Finder. The ReferenceForm's content is similar to the Default Form, but has different property settings for the topStack element.



## Finder User Interface Variables

The Variables tab displays the system-generated variables for the Finder User Interface that you have selected.

The Finder Variable details appear in the detail pane when a Variable entry is highlighted in **Variables list**. The Finder Variable properties are listed in the [chart that follows](#). Some of the properties appear in the columns of the **Variables list** table.

- The **Type** column is populated with the primitive data type of the Variable's selected Data Type (for example, String), its actual data type (for example, com.conceptwave.system.String), and its length, if it is specified.
- The **Methods** checkbox is checked whenever the Finder Variable contains one or more Methods.
- The **Vis**, **Opt** and **Edit** fields denote the **Visible**, **Optional**, and **Editable** Element properties, respectively. For more information about Variables see, [Document Variables](#).

The screenshot shows the 'Variables' tab of the Finder User Interface configuration. The top part is a table titled 'Variables' with columns: Name, Type, Methods, Vis, Opt, and Edit. The 'Methods' column contains checkboxes. The 'Vis', 'Opt', and 'Edit' columns contain dropdown menus with options like 'Array', 'Constant', 'Audit', and 'Validate'. The bottom part is a detailed view for the selected variable 'cwShowHover'. It includes fields for Name, Description, Data type (set to 'com.conceptwave.system.Boolean'), Type error code, Mandatory error code, and Element properties ('<Inherited>'). A table at the bottom lists properties with columns 'Name' and 'Value'.

Name	Type	Methods	Vis	Opt	Edit
cwShowHover	com.conceptwave.system.Boolean(Boolean)	<input checked="" type="checkbox"/>			
confirmObject	com.conceptwave.system.Object(Object)	<input type="checkbox"/>			
model	com.conceptwave.system.Model(Model)	<input type="checkbox"/>			
cwSelectCallback	com.conceptwave.system.String(String)	<input type="checkbox"/>			
detail	com.conceptwave.system.UserInterface(User Interface)	<input type="checkbox"/>			
detailForm	com.conceptwave.system.String(String)	<input type="checkbox"/>			
result	com.conceptwave.system.Document(Document)	<input type="checkbox"/>			
resultForm	com.conceptwave.system.String(String)	<input type="checkbox"/>			
disableValidation	com.conceptwave.system.Boolean(Boolean)	<input type="checkbox"/>			
cwHasFavorite	com.conceptwave.system.Boolean(Boolean)	<input type="checkbox"/>			
chart	com.conceptwave.system.Boolean(Boolean)	<input type="checkbox"/>			
header	com.conceptwave.system.String(String)	<input type="checkbox"/>			
tableState	com.conceptwave.system.String(String)	<input type="checkbox"/>			
views	com.conceptwave.system.FinderViewIteratedItem(Finder View Iterated Item)	<input type="checkbox"/>			
finder	com.conceptwave.system.Finder(Finder)	<input type="checkbox"/>			
search	com.conceptwave.system.UserInterface(User Interface)	<input type="checkbox"/>			

**General**   **Methods**

**Name:** cwShowHover    Array    Constant    Audit    Validate

**Description:**

**Data type:** com.conceptwave.system.Boolean

**Type error code:**    **Mandatory error code:**

**Element properties:** <Inherited>

Name	Value

The rows in the **Variables list** table can be reorganized using the up and down arrow buttons located beside the table. Highlight an item, then click either the up arrow or down arrow button to move it. They can also be sorted alphabetically by clicking the **Sort** button .

The **Add** button is used to add a Data Type as a Document Variable in the **Variables list**. When you click the **Add** button, the **Add Element** dialog opens.

The dialog contains a list of Data Types under all namespaces in the metadata. To add Variables, highlight the elements, and click the **Save** button. The dialog closes and the selected elements are added as Variables in the **Variables list**.

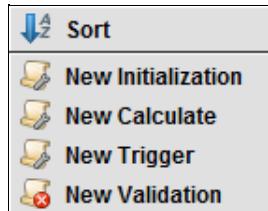
To remove Variables from the table, highlight one or more rows in the **Variables list** and click the **Remove** button . A Variable that is selected as a system-generated key cannot be removed.

These buttons beside the list follow the [Manage Array Elements](#) presentation convention in Velocity Studio. See this section for details about the aforementioned and additional buttons.

The following system variables are generated for the *com.conceptwave.system.FinderUserInterface*.

Variable	Return Type	Description
<b>confirmObject</b>	Object	Refer to Confirm JavaScript Extension.
<b>model</b>	Model	Data Finder (equivalent to Finder JavaScript Extension).
<b>detail</b>	User Interface	Detail document of the Finder. It is populated when a selection is performed on the Finder and the <code>showDetailAction</code> method is invoked.
<b>cwShowHover</b>	Boolean	Instructs the Web browser to show hover data when the method is set to true for table's hover method.
<b>detailForm</b>	String	The form to use for the detail view (includes any form under the output document).
<b>result</b>	Document	An array of documents from the database or a query.
<b>resultForm</b>	User Interface	Determines the form name to be displayed. The results Frame uses this variable as a form property.
<b>header</b>	String	The section header label of the Finder. This can be changed using views.
<b>tableState</b>	String	Used for views. It keeps track of the current sorting field visibility of the table. This variable cannot be changed.  To customize your view on a finder, see the <a href="#">TableState object</a> for details.
<b>views</b>	User Interface	The views available for this Finder.
<b>isEditable</b>	Boolean	Set to true when the Finder table or detail form is editable.
<b>search</b>	User Interface	The Search document's User Interface.
<b>SearchForm</b>	String	Search form name used by the Finder.
<b>isSearchFormVisible</b>	Boolean	Keeps track of the visibility search form section of the search form.
<b>invokeContext</b>	Object	Internal functionality of the Finder.
<b>isSearchButtonVisible</b>	Boolean	Determines whether the search menu is visible or not.
<b>chart</b>	Boolean	When set to true shows the Chart Form. False hides the Chart.
<b>result</b>	Document	The results list of the document.

The Methods tab is a [script management screen](#) in Velocity Studio, which is composed of the **Method List pane** at the left which lists all scripts of the Finder Variable, and **Details pane** which displays the details of a script when selected at the Method List pane. To create a new Method, or to override an existing Method of its base Document, right-click the Finder in the Method List pane. A pop-up menu appears.



**Note:** This creates a Method at the Finder Variable level, impacting only the Finder Variable. To create Methods at the Finder level, go to [Finder Methods](#).

## Variable Methods

This table lists the type of Methods that can be added or overridden at the Document Variable level.

Method Type	Description
 Validation	Finder Variable validation script that determines the validity of runtime data values of the Finder. It is triggered with Finder method <code>.validate</code> , and by default, the generated Finder Form's save menu button triggers such validation. See the <a href="#">phasedValidation configuration variable</a> for details.

The following lists the system-defined Methods that can be overridden.

Method Type	Script Name	Parameters	Return Type	Description
<b>Initialization</b>	<code>cwLeafInitAction</code>	None	Document Variable	<p>Finder User Interface Variable Initialization script is to set initial values for the Finder Variable.</p> <p>It is triggered when the Document is created. This Method is not executed when the Document is read from the database or an external system interface.</p>
<b>Calculate</b>	<code>cwLeafCalculateAction</code>	None	Document Variable	<p>Finder User Interface Variable Calculate script is to compute a value for the calculated Finder Variable. A calculated Finder Variable is a Variable that is not stored in the database or sourced from external sources, but their values are computed based on values of other Finder Variables. Variables with calculate Method are read-only fields in the UI at runtime.</p> <p><b>Note:</b> Because the value of the calculated field is computed every time the field is accessed, it may be accessed many times for different purposes by AVM at runtime. Unwise and excessive use of calculated fields may introduce serious performance problems. It is recommended to limit the Method to simple calculations based on Finder Variable values, and not invoke database access operations or external interface calls.</p>
<b>Trigger</b>	<code>cwLeafTriggerAction</code>	None	None	<p>Finder User Interface Variable Trigger script is executed when the content of a Finder Form field is changed (and loses focus) at runtime. Trigger execution may come automatically from the UI or can come from a user script at runtime. When a trigger execution request is received, all triggers for Variables that has been changed are executed.</p> <p><b>Note:</b> As a Trigger Method is triggered at UI, a server round trip is required to send updates to the client; this is done in AJAX, with only changed variables updated to browser with no page refresh. Regardless, Trigger Methods should be used sparingly to avoid incurring this overhead.</p>

For Method Types that can be created, you can create multiple Methods of the same Method type (for example, Validation 1, Validation 2...), each identifiable by unique Method name. However, for Methods that are to be overridden from base Document, you can only define one Method per base Method. All of the system-predefined Methods `com.conceptwave.system.Finder` have empty scripts, which means the default Method is to do nothing.

For scripts that are already overridden, they appear in the Method List pane and no longer be available at the right-click context pop-up menu.

The Method created in Method List pane can be right-clicked, with various commands available; see [script management UI](#) for details.

## TableState Object

The TableState object is scriptable and is intended to be used when you need to customize the table view through scripts. The object parses the JSON string of the table state and provides an API to change the following column states:

- Hide or show columns
- Set width
- Set column order
- Set one or more sort columns

### Get the TableState Object

To get the TableState object, use the following finder User Interface method:

```
var stateObj = this.getTableState();
```

The `getTableState()` method returns the parsed TableState object or null if the table state is not received from the Web browser.

You can also create it manually and pass the stored JSON string state:

```
var stateObj = new TableState(storedTableStateString);
```

Then, you can serialize the changed state back to the JSON string as follows:

```
var tableStateString = stateObj.toString();
```

### Setting TableState

To apply the changed table state, call the method of the finder User Interface:

```
this.setTableState(stateObj);
```

**Note:** If you need to restore or change the table state on the finder User Interface initialization, call the `super.onInit()` method first, and then change the state:

```
// Example of changing table state in onInit in finder UI
// call super.firstL, as it will clear tableState var
this.cw$super_onInit();

var state = new TableState();
state.addField("OrderId", true, null, null);
state.addField("custName", true, null, null);
state.addField("orderType", true, null, null);
state.addField("getDate", false, null, null);
state.addField("curEditor", false, null, null);
state.addField("OrderStatus", true, null, null);
state.addField("ezStatus", false, null, null);
state.addField("solution", false, null, null);
state.addSorting("OrderId", true);
state.addSorting("orderType", false);
this.setTableState(state);
```

### TableState API

The following table shows the main methods of the TableState object:

Method Name	Description
<code>fromString(storedStateString)</code>	Parses the stored JSON table state into the TableState object.
<code>toString()</code>	Serializes the TableState object back to the JSON string.
<code>addField(String name, Boolean visible, String width, Integer order)</code>	Adds the column state. The width and order are optional.
<code>getFieldsCount()</code>	Returns the number of field states in the TableState object.
<code>getFieldName(index)</code>	Returns the field name at the specified index.
<code>setVisible(String name, boolean visible)</code> <code>setWidth(String name, String width) // width can be null</code> <code>setOrder(String name, int order)</code>	Sets the corresponding property of the column.
<code>isVisible(String name)</code> <code>getWidth(String name) // width can be null</code> <code>getOrder(String name)</code>	Gets the corresponding property of the column.

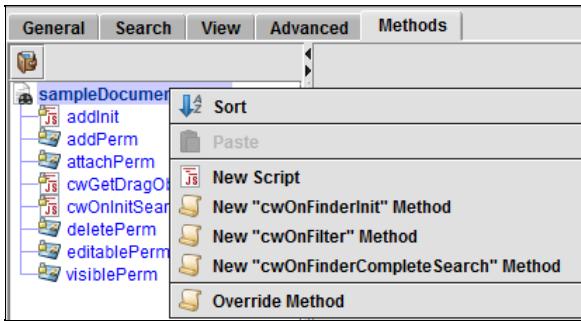
You can use the following methods for sorting:

- `setSorting(String name, Boolean ascending)`
- `addSorting(String name, Boolean ascending)`
- `getSortCount()`
- `getSortField(index) // returns column name`
- `getSortDir(index) // return true or false`

See the Javadoc API section for a full list of methods.

## Finder User Interface Methods

The Methods tab is a [script management screen](#) in Velocity Studio, which is composed of the **Method List pane** at the left, and lists all scripts of the Finder, and **Details pane**. This displays the details of a script when selected at the Method List pane. To create a new Method, right-click the Finder in the Method List pane and select a Method from the pop-up menu.

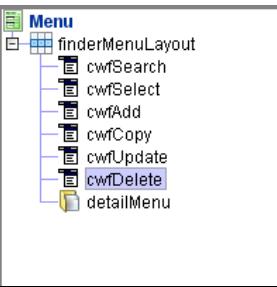
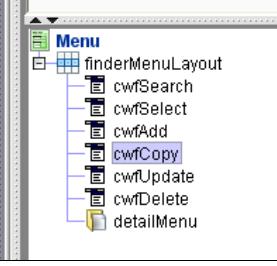


If you have a long list of methods in the left pane, you can press the first letter of the method you are looking for to jump to the list of methods beginning with that letter. For example, pressing the **I** key highlights the first instance of a method that begins with the letter I. The method displays in the right pane.

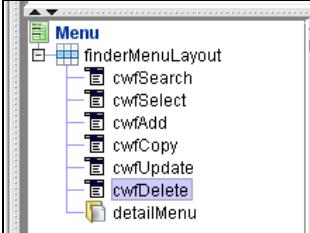
The following table lists the types system-defined Methods that can be overridden at the Finder User Interface level.

**Note:** When library methods are overridden, it is recommended that the *super method* is invoked first and a custom script is written after the super call. If the super method is not invoked the Finder may not work as it is expected.

Method	Usage	Code	Parameters	Description
addAction	Invoked when user clicks on the Add menu	<pre>var finder = this.model.metadata;  if(finder.isViewEditable()){     this.searchAction();      addParam = new Document(this.model.selectOutputDocumentMetadataName);      var toAdd = finder.operations.add.invoke(addParam,         this.invokeContext);      this.addInit(toAdd);      this.result.updateList; //force update } else{     var finder = this.model.metadata;      var toAdd = addParam ? addParam : new         finder.operations.add.input.UserInterface(null, this);      this.isSearchButtonVisible = false;     this.addInit(toAdd);     this.detail = toAdd; }  // Notify listeners  Notifier.notify(this.model, "add", toAdd.model);</pre>	addParam; type Document	<p>Finders with non-editable tables create a new instance of a Document providing the Finder's output document name. The Finder's add operation is then called with the new instance and <i>addInit()</i> is called if any script is provided. The search menu of the Finder is hidden and the new document is set as this Finder's detail variable.</p> <p>Finders with editable tables call the <i>searchAction()</i> method in cases when this Finder is currently not displaying any data. A new document is then created and the table element is notified to open the new row in edit mode.</p>
addCustomView	Invoked when the menu addView is clicked.	return cwf_oe.addToFavorites(true);	N/A	Creates a new instance of <i>cwf_oe.Favorite</i> document and presents it on a dialog popup for users to create a new custom view for this Finder
addInit	Invoked by <i>addAction()</i>	N/A	N/A	May be used to initialize values of a newly added document in this Finder.
canDelete	Used as the visible permission property of the <i>cwfDelete</i> menu.	<pre>if(!this.model.deletePerm())     return false;  if(this.result.selected != null &amp;&amp; this.result.selected.length == 1){     if(!this.result.selected[0].deletePerm())</pre>	\$psCondition; type Boolean; Outcome of the privileges operation	Determines the user's ability to remove data from this Finder using the delete menu. If this Finder's delete

	 <pre> Menu └── finderMenuLayout     ├── cwfSearch     ├── cwfSelect     ├── cwfAdd     ├── cwfCopy     ├── cwfUpdate     └── cwfDelete     └── detailMenu </pre>	<pre>         return false;     }      return this.detail!=null    (this.result!=null &amp;&amp; this.result.selected!=null); } </pre>		<p>permission returns <i>false</i>, the delete menu does not show for this Finder. Otherwise, if the user is currently viewing the result table and a row is selected, then the delete menu is visible. If the Finder's delete permission returns <i>true</i> but the current selected document's delete permission returns <i>false</i>, this delete menu is hidden.</p>
<b>canEditInTable</b>	Invoked when the addAction is called.	return this.model.isViewEditable;	\$psCondition; type Boolean; Outcome of the privileges operation	Accesses the current Result Form (based on the current view displayed for this Finder) and determines the editable property of the Finder table within that Result Form.
<b>changeFinderView</b>	Invoked when the user clicks on a view listed under the finder preferences menu (if any)	<pre> if (this.model != null &amp;&amp; this.model instanceof Finder &amp;&amp; this.model.metadata != null &amp;&amp; view != null) {     this.model.setViewView(view);     this.header = view;      var finder = this.model;     this.detailForm = finder.getDetailForm();      this.initSearch(); } </pre>	view; type String; the name of the view selected	Utilizes the view parameter as a view name and tries to find and set as the current view of the Finder. The Search Form, Result Form, Detail Form are set based on the current view. The Search Form is also initialized. The state of the table for this Finder, including the order of the columns as well as the sorting of the data, is set.
<b>copyAction</b>	Invoked when the user clicks on the copy menu.	 <pre> Menu └── finderMenuLayout     ├── cwfSearch     ├── cwfSelect     ├── cwfAdd     ├── cwfCopy     ├── cwfUpdate     └── cwfDelete     └── detailMenu </pre> <pre> var finder = this.model.metadata; if (this.result.selected!=null &amp;&amp; this.result.selected.length == 1) {     var selectedDoc = this.result.selected[0];     var copiedDoc = selectedDoc.copyDocument(false, true);      if(finder.setEditable()){         this.searchAction();         var addParam = copiedDoc;         var toAdd = finder.operations.add.invoke(addParam, this.invokeContext);         copiedDoc.docCopyScript();         this.addInit(toAdd);         this.result.updateList; //force update     }else{         this.detail = new copiedDoc.metadata.UserInterface(copiedDoc, this);         this.isSearchButtonVisible = false;         this.detail.parent = this;     } }  copiedDoc.save(); // Notify listeners </pre>	N/A	<p>This method creates a new instance of a Document via the <i>copyDocument</i> method on the currently selected document (method is provided under the JavaScript Extensions).</p> <p>The Finder's add operation is then called with the new instance and <i>addInit()</i> method is called if any script is provided. The search menu of the Finder is hidden and the new document is set as this Finder's detail variable. Finders with editable tables calls the <i>searchAction()</i> method in cases when this finder is currently not displaying any data. The table element is notified to open the new row in edit mode.</p>

		<pre> Notifier.notify(this.model, "add", copiedDoc); } </pre>		
<b>cwGetDragData</b>	This method is called on the source controller <i>before</i> cwGetDropData is called.	return object;	object	For finder user interfaces, the default implementation of the method returns the object passed in.
<b>cwGetDragObjectType</b>	This method is used to allow metadata to provide a different type for the object than what is used by the product.	return this.model.cwGetDragObjectType(dobj);	object	When this method is called on a finder, the finder returns the output document's type.
<b>cwGetDragTypes</b>	This method is called during the fetch of the data. When an object's type is not included in the list returned by this method, it is marked as canDrag=false.	return this.model.metadata.selectOutputDocument.toString();	None	The default implementation of this method for finder user interface returns a list of metadata type names that correspond to the output object's metadata type, and any objects in the metadata which extend this output type.
<b>cwGetDropData</b>	This method is called on the target controller near the beginning of cwOnDrop <i>after</i> cwGetDropData has been called.	return object;	object	The default implementation of this method returns the object passed in.
<b>cwGetDropTypes</b>	This method is called as part of the data fetch. The list returned from this method is sent to the browser for verification of object acceptance when a user attempts to drop an object.	return this.model.metadata.selectOutputDocument.toString();	None	This method returns an array of metadata type names.
<b>cwOnDragOut</b>	This method is called once cwOnDrop has successfully completed.	<pre> if(sourceDs != null){      if(dragAction == "copy"){          return;      }else{          if(draggedData .length &gt;= 1){              for(var i = 0; i &lt; draggedData.length; i++){                  sourceDs.remove(draggedData[i]);              }          }      }      sourceDs.updateList;  } </pre>	draggedData (Object); sourceName (String); sourceDs (Object); actualDataDropped (Object); dragAction (String)	This method performs certain actions once cwOnDrop has successfully completed.
<b>cwOnDrop</b>	This method is called when the user drags and releases the mouse button on a table or tree.	<pre> var actualDataDropped;  try{ if(sourceDs != null){ //dragged data from source perspective actualDataDropped =(getDragDataMethod != null &amp;&amp; Global.isMethodDefined(source, getDragDataMethod))? source[getDragDataMethod](dataDropped):dataDropped; if(Global.isMethodDefined(this,getDropDataMethod)){ //dropped data from target perspective actualDataDropped = this[getDropDataMethod](actualDataDropped, sourceName, sourceDs, dragAction); } var selectList = new Array();  for(var i = 0; i &lt; actualDataDropped.length; i++){ var toAdd = actualDataDropped[i].copy(dragAction=="copy"? false:true); targetDs.add(dropIndex+i,toAdd); selectList[selectList.length] = toAdd; }  targetDs.setSelected(selectList); targetDs.updateList; } }catch(ex){ Global.throwException("UU0320"); } </pre>	source, (Object); sourceName (String); sourceDs (Object); dataDropped (Object); dropIndex (Integer); dragAction (String); target (Object); targetName (String); targetDs (Object); getDropDataMethod (String); getDragDataMethod (String); dragMethod (String)	<p>This method is the default implementation for drag and drop for tables. For finder tables which accept dropped objects, the following actions take place:</p> <p>The "Get Drop Data" method specified on the source element is called on the source controller first. The objects being dropped serve as a parameter. The source controller decides what to return for the target to work with. The object returned from this method serves as the dropped object.</p> <p>1) If the drag action is a "copy" (achieved by</p>

	<pre> this.cwOnSaveDrop(targetDs); if(dragMethod != null &amp;&amp; Global.isMethodDefined(source, dragMethod)){ source[dragMethod](dataDropped, sourceName, sourceDs, actualDataDropped, dragAction); }  if(Global.isMethodDefined(source, "cwOnSaveDrop")){ source.cwOnSaveDrop(sourceDs); } </pre>	<p>holding down the CTRL-key until the mouse is released to drop the object), this method will iterate through the list of objects that have been dropped and call "copy" on it. The copied object is then added to the targetDs starting at dropIndex.</p> <p>2) If the drag action is a "move" (the default drag and drop action), this method iterates through the list of objects that have been dropped and adds to the targetDs starting at dropIndex.</p> <p>If either #1 or #2 is successful, this method calls the drag out method specified on the source element.</p>		
<b>cwOnVelocityRowstyle</b>	<p>This method exists under the Finder User Interface level. This method is called for every result document for that finder when finder data is fetched.</p> <p><b>Note:</b> The cwOnVelocityRowstyle method is called only if the Finder User Interface has overwritten this method.</p>	<p>If the result document contains a variable called isNew, one may write a function like the following:</p> <pre> if(document.isNew)     return "styleWithGreenFont"; else     return "styleWithBlackFont"; </pre>	document; type Document; the current document being processed	This method must return a string representing the style name, which can be found in the application's cascading stylesheet (.css) file. For convenience, the parameter passed in by the system represents the current record (data document) being processed.
<b>deleteAction</b>	Invoked when the user clicks on the delete menu.  	<pre> var finder = this.model.metadata; var i;  for (i = 0; i &lt; this.result.selected.length; i++) {     var toDelete = this.result.selected[i];     finder.operations.deleteOp.invoke(toDelete, this.invokeContext);     if (this.detail!= null) {         this.detail = null;     } }  this.result.updateList; //force update </pre>	N/A	Takes the currently selected document and calls the Finder's delete operation. If the detail form of this document is currently being displayed, set the detail variable to null. Once the operation is successful, update the Finder's data to reflect the change.
<b>doinitFinder</b>	Invoked from the onInit() method of the finder.	<pre> var finder = this.model.metadata; this.invokeContext = new com.conceptwave.system.InvokeContext(finder, this.model);  if (this.model.searchInput != null) {     this.search = new this.model.searchInput.UserInterface(null, this);     if (this.model.searchDocument!=null){         this.search.model = this.model.searchDocument;     } }  this.isSearchButtonVisible = true; this.chart = false; </pre>	N/A	This methods main functionality is to initialize values of essential variables such as the search and result variable.

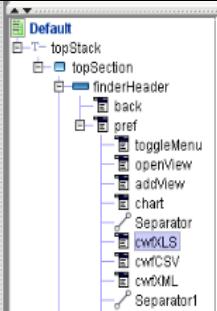
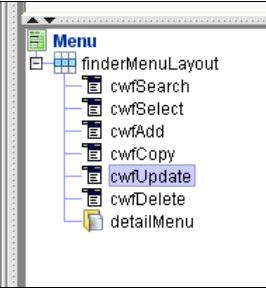
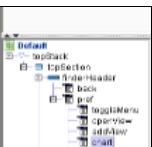
```

this.searchForm = this.model.getSearchForm();
this.resultForm = this.model.getResultForm();
this.detailForm = this.model.getDetailForm();
this.result = this.model.list;

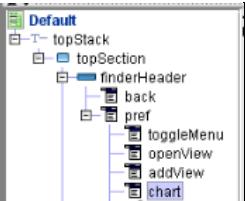
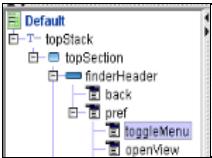
var viewsList = this.model.getViews();
if (viewsList != null) {
    var viewsDoc = new Array();
    for(var i = 0; i < viewsList.length; i++) {
        viewsDoc[i] = new com.conceptwave.system.FinderViewIteratedItem;
        viewsDoc[i].view = viewsList[i].toString();
        viewsDoc[i].finder = this;
    }
    this.views = viewsDoc;
    this.header = this.model.getVelView();
    this.changeFinderView(this.header);
} else {
    this.initSearch();
}

```

<b>downloadCsv</b>	Invoked when the cwfCSV menu is invoked.	<pre> this.resetModelStatus(); var download = new com.conceptwave.system.DownloadCsv(null, this); download.fileName = "finder.csv"; download.objectToPrint = this; download.mimeType = "text/csv"; download.resultUI = this.resultUI; download.resultForm = this.resultForm; Global.doDownload(download); </pre>	N/A	<p>Creates a new instance of DownloadCsv and instantiates the necessary values. Global.doDownload sets the download variable as the current application's cwDownload variable value. The generated method of the download is then called to process the Finder's data and present it to the user in comma-separated-value format as a browser download.</p> <p>Download.resultUI and download.resultForm are used to specify which user interface to use to look up the result table if result form of finder user interface does not have a table element.</p>
<b>downloadXml</b>	Invoked when the cwfXml menu is invoked.	<pre> this.resetModelStatus(); var download = new com.conceptwave.system.DownloadCsv(null, this); download.fileName = "finder.csv"; download.objectToPrint = this; download.mimeType = "text/csv"; download.resultUI = this.resultUI; download.resultForm = this.resultForm; Global.doDownload(download); </pre>	N/A	<p>Creates a new instance of DownloadXml and instantiates the necessary values. Global.doDownload sets the download variable as the current application's cwDownload variable value. The generate method of the download is then called to process the Finder's data and present it to the user in XML format as a browser download.</p> <p>Download.resultUI and download.resultForm</p>

				are used to specify which user interface to use to look up the result table if result form of finder does not have a table element.
downloadXsl	Invoked when the cwfXsl menu is invoked.  	<pre>this.resetModelStatus(); var download = new com.conceptwave.system.DownloadXls(null, this); download.fileName = "/cwf/finder.xls"; download.objectToPrint = this; download.mimeType = "application/vnd.ms-excel"; download.resultUI = this.resultUI; download.resultForm = this.resultForm; Global.doDownload(download);</pre>	N/A	Creates a new instance of DownloadXsl and instantiates the necessary values. Global.doDownload sets the download variable as the current application's cwDownload variable value. The generate method of the download is then called to process the finder's data and present it to the user in XSL format as a browser download.  <i>Download.resultUI</i> and <i>download.resultForm</i> are used to specify which user interface to use to look up the result table if result form of finder does not have a table element.
editVisible	Used as cwfUpdate menu's visible permission property.  	return this.searchVisible();	\$psCondition; type Boolean; Outcome of the privileges operation	Returns <i>true</i> if this Finder's detail variable value is set to null (the detail form is not currently shown in the browser).
hasChart	Used as the visible permission property of the chart menu.  	return this.model.isChartOverridden();	\$psCondition; type Boolean; Outcome of the privileges operation	Controls visibility of a chart when the set permissions are evaluated.
hideDetailAction	Invoked when the user clicks on the back menu (visible only when the detail form is currently being shown).	<pre>this.detail = null; this.isSearchButtonVisible = true; this.searchAction();</pre>	N/A	Sets the necessary variables used to determine the visibility of the <i>detailFormSection</i> .
initSearch	Invoked when the view for this finder is changed.	<pre>if (this.model.getAutoSearch()) {     this.searchAction(); } else {     if (this.result == null)         this.result = this.model.metadata.operations.select.emptyResult(); }</pre> <pre>this.isSearchFormVisible = this.model.isShowSearchForm()</pre>	N/A	Initializes the result data according to the current view selected. If the current view specifies to use auto search, this Finder displays the result data otherwise the result data is empty.
onCalculate	When you create a new onCalculate under the finder user interface, the product automatically adds a document parameter. The code examples shows how to use the onCalculate method to create a customized checkmark icon in a finder result table.	<ol style="list-style-type: none"> <li>Add an image element under the table element.</li> <li>Create an onCalculate method that returns a string and set this as the image element's variable property.</li> <li>In your script, perform actions such as the following:</li> </ol>	document; type Document	The doc parameter represents the current result document being evaluated at the time

		<pre> if(doc.boolean1){     return "images\customCheck.png"; }  }else if(doc.boolean2){     return "images\customUnCheck.png" } </pre>		of the call.
onEnter	This method can be override (if desired) and set on the table's onEnter property.  For an example of using the onEnter method, see <a href="#">Set onEnter Property on Form Frame to Display Search Results</a> .	this.searchAction();	N/A	Default method invoked when the user's focus is on the Finder table and the <b>Enter</b> key is pressed.
onInit	Called when the finder user interface is created either through script (ex var f = new cwf_oe.EventLogFinder.UserInterface() ) or when an instance of a Finder is returned from a User Action Method invoked by a user interface element such as a menu.	this.dolnitFinder();	N/A	Initializes essential variable values for the Finder view <i>dolnitFinder</i> .
OnSelChanged	Invoked when the user makes a selection on this Finder table	<pre> if(this.model.isSingleClickDetail()){     this.showDetailAction(); }  }else if(this.model.isViewEditable){      var doToggle = this.result.selectedColToggle; //determines whether to change value or not - available for checkbox under table ONLY.      var columnName = this.result.selectedColName; //the column being clicked on - this is the name of the variable assigned to the checkbox      if(doToggle == "true"){          if(this.result.selected[0]!=null){              var doc = this.result.selected[0];             eval("doc."+columnName+"=!doc."+columnName);              this.result.updateList; //update result to show doc change         }     } } </pre>	N/A	If the current view of this finder has <i>Single Click Detail</i> set to True, the <i>showDetailAction()</i> method is called. Otherwise, if the Finder's table is editable, this method checks if the current column that the user has clicked has specified the toggle functionality to be turned on. This <i>Toggle Edit</i> property can only be set on a checkbox within the table element. If this property is set to <i>true</i> , the value of this variable assigned to the checkbox is modified accordingly and the Finder result data is updated.
resultVisible	Used as the visible permission property of the result section.	return this.detail == null && !this.chart;	\$psCondition; type Boolean; Outcome of the privileges operation	Returns <i>true</i> if a Detail Form or a Chart Form is not currently being viewed.
searchAction	Invoked primarily when the user clicks on the cwfSearch menu.	<pre> var finder = this.model.metadata;  var searchDoc = this.search==null? (this.model.searchDocument==null? null:this.model.searchDocument):this.search.model;  this.result = finder.operations.select.invoke(searchDoc, this.invokeContext); </pre>	N/A	Performs the Finder's search operation according to the search criteria (if any) provided by the user.
searchVisible	Used as the visible permission property of the searchFormSection.	return this.detail == null;	\$psCondition; type Boolean; Outcome of the privileges operation	Returns <i>true</i> if the <i>detail</i> variable does not currently have a value. By default, the Default Form and the Search Form cannot be visible at the same time.
showChart	Invoked when the user clicks on the chart menu.	this.chart = true;	N/A	Modifies the chart and

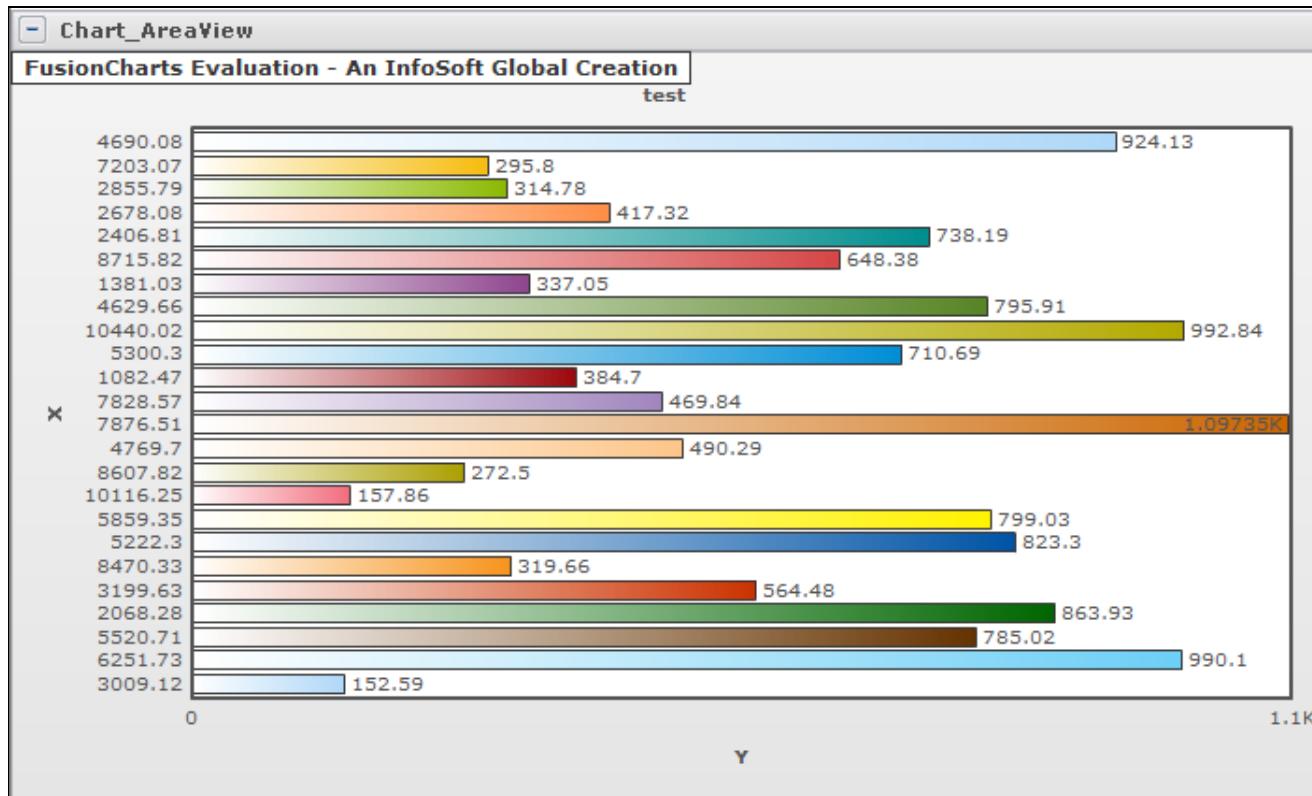
		this.isSearchButtonVisible = false;		isSearchButtonVisible variables accordingly. The chart variable is used to determine the chart form's visibility (see chartVisible). The isSearchButtonVisible variable is used in searchVisible() method to determine the visibility of the Search Form.	
showDetailAction	Invoked when the user clicks on the cwfUpdate menu. Also, if the finder's table has "Show Detail Column" set to true, this method is invoked when the detail column is clicked.	<pre>if(this.result.selected != null &amp;&amp; this.result.selected.length == 1) {     //if this finder is being displayed as a reference finder, return the currently selected     //document instead of displaying the detail document.      if(this.model.refFinderDocument!=null){         if(this.model.isRefEditable){              var leafName = this.model.parentLeafName;             eval("this.model.refFinderDocument."+leafName+" = this.result.selected[0].id");              Global.getCurrentObject().dialog = null;             //if this finder is being shown as an editfield for an editable table             if(this.model.getRootFinderParent!=null){                  this.model.getRootFinderParent.result.updateList;             }         }     }else if(this.detailForm!=null )         if(this.result.selected[0].metadata!=null){              this.detail = new             this.result.selected[0].metadata.UserInterface(this.result.selected[0],this);              this.detail.parent = this;             this.isSearchButtonVisible = false;         }     } else {         this.detail = null;     } }</pre>	N/A	In a regular Finder, this method takes the currently selected row and create the user interface for it. The newly created user interface instance is then set as the detail variable's value. If this Finder has specified a detail form to be used, the detailSection in this finder's Default form displays the detail document using the corresponding Detail Form.  If this Finder is currently being used as a Reference Finder (made visible by clicking a reference element icon), this method takes the currently selected document's ID and sets it as this Finder's parent document's variable value (depending to which variable the reference element is bound).	
showSearchForm	Used as the visible permission property of the searchFormSection.	return !this.backVisible() && (this.isSearchFormVisible == null    this.isSearchFormVisible);	\$psCondition; type Boolean; Outcome of the privileges operation	Returns true if the detail variable currently does not have value and isSearchFormVisible returns false.	
toggle SearchAction	Invoked when the user clicks on the toggleMenu.		this.isSearchFormVisible = !this.isSearchFormVisible;	N/A	Sets the isSearchFormVisible to its opposite value. This variable's value determines the visibility of the search form (see searchVisible).
toggleVisible	Used as the visible permission property of the toggleMenu.	return (this.detail==null && this.search!=null && this.searchForm!=null);	\$psCondition; type Boolean; Outcome of the privileges operation	Returns true if the Detail Form is not currently being viewed and this Finder has specified an input document and a search form to use.	
updateAction	Invoked when the cwfUpdate menu is clicked.	<pre>if (this.result.selected.length == 1) {      this.detail = new     this.result.selected[0].metadata.UserInterface(this.result.selected[0],</pre>	N/A	Takes the currently selected row data and displays the Detail Form and the	

```
    this);
    this.isSearchButtonVisible = false;
}
```

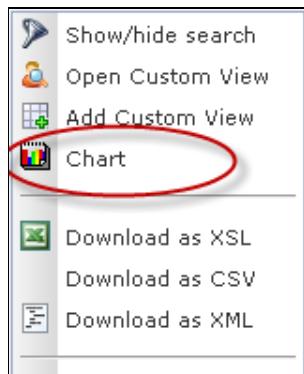
data populated for  
visible form elements.

## Finder Chart Form

The **Finder Chart Form** is, by default, one of the Forms that appears in Finder User Interface. The Chart Form is used to define different chart types (for example, bar, pie, line) that can be used for animated in 2D or 3D graphical presentation of the Finder Result. Data that is configured to display in the Result Form can also be configured to display in a chart.

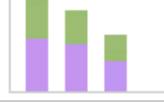


The Form elements, Chart and ChartField, can be configured to display the data displayed in the Result Form. The Chart option contained in the Preference menu allows users to display the Result data in a graphical format in a Web browser at runtime.



The following list the types of Charts and the different types of display that are available:

Chart Type	Example	Description
Area	A line graph titled "Monthly Sales Summary For the year 2006". The Y-axis is labeled "Sales" with increments of \$15K to \$40K. The X-axis is labeled "Months" with months from Jan to Dec. The line starts at approximately \$15K in Jan, rises to \$25K in Feb, dips to \$22K in Mar, rises to \$30K in Apr, dips to \$28K in May, rises to \$35K in Jun, dips to \$32K in Jul, rises to \$38K in Aug, dips to \$35K in Sep, rises to \$40K in Oct, dips to \$38K in Nov, and ends at \$42K in Dec. A light gray shaded area follows the line, representing the area under the curve.	Shows trends and performance using solid area over a period of time Appears in 2D (Area2D) Appears in (multi-series)

Bar		Horizontal bars to support longer data labels Appears in 2D (Bar2D) and 3D (Bar3D) Appears in multi-series2D and multi-series3D
Column		Compare data using columns Appears in 2D (Column2D) and (Column3D) 3D Appears in multi-series2D and multi-series3D
Line		Shows trends and performance using lines over a period of time Appears in 2D (Line2D) Appears in (multi-series)
Pie		Shows individual values with respect to total. Appears in 2D (Pie2D) and 3D (Pie3D)
Scroll Area		Area charts with scrollable x-axis Appears in 2D (ScatterArea2D)
Scroll Column	Similar to scroll area but displays with columns	Column charts with scrollable x-axis Appears in 2D (ScrollColumn2D)
Scroll Line	Similar to scroll area but displays with lines	Line charts with scrollable x-axis Appears in 2D (ScrollLine2D)
Scroll Stacked Column	Stacked area columns with scroll bar	Show relative quantity with respect to whole with scroll bar Appears in 2D (ScrollStackedCol2D)
Stacked Area		Show relative quantity with respect to whole Appears in 2D (StackedArea2D)
Stacked Bar	Similar to stacked area but displays with bars	Show relative quantity with respect to whole Appears in 2D (StackedBar2D) and 3D (StackedBar3D)
Stacked Column	Similar to stacked area but displays with columns	Show relative quantity with respect to whole Appears in 2D (StackedColumn2D) and 3D (StackedColumn3D)

**Note:** The chart displays as an empty Form when it is configured but there is not any data to display.

## Configuring the Chart

There are several steps involved to configure the Chart Form:

1. To display a chart, you must override the Chart Form by right-clicking it and selecting **Override** from the menu (see [Override a Form](#)).
2. In the Element Tree pane, right-click the Chart Form and from the menu click **Add**.
3. From the Add Element dialog box, select **Structures > Chart** and click **Finish**.
4. Continue by configuring the properties of the Chart Form.
5. Right-click the Chart Form element, and from the menu click **Add**.
6. From the Select Form Element Type dialog box, select **Misc Controls > Chart Field** and click **Finish**.
7. Continue by configuring the properties of the ChartField.

## Configuring Chart Properties

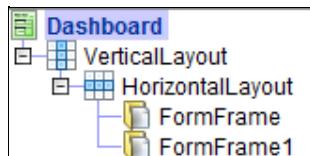
After adding the Chart Form element you can configure the chart properties. The Chart Form element contains properties that enable you to bind to a results form using the Variable property. To view the Chart at runtime, you need to configure the Variable property. The Variable points to an array of documents for the Finder. From the Variable property field, select a result form from the drop-down menu. For a list and description of chart properties, see [Chart Form Element](#)

The ChartField Form element configures the display of the chart at runtime. For example, you can configure the color and label of the Chart. In addition you can configure the variable (a numeric variable from the Chart Form element). For a list and description of chart properties, see [ChartField Form Element](#).

**Note:** By default, the Variable property field in the Chart Form element is configured to refer to the Result variable of the Finder User Interface. This variable is configured as an array of documents. When customizing this field, ensure that variable refers to an array of documents.

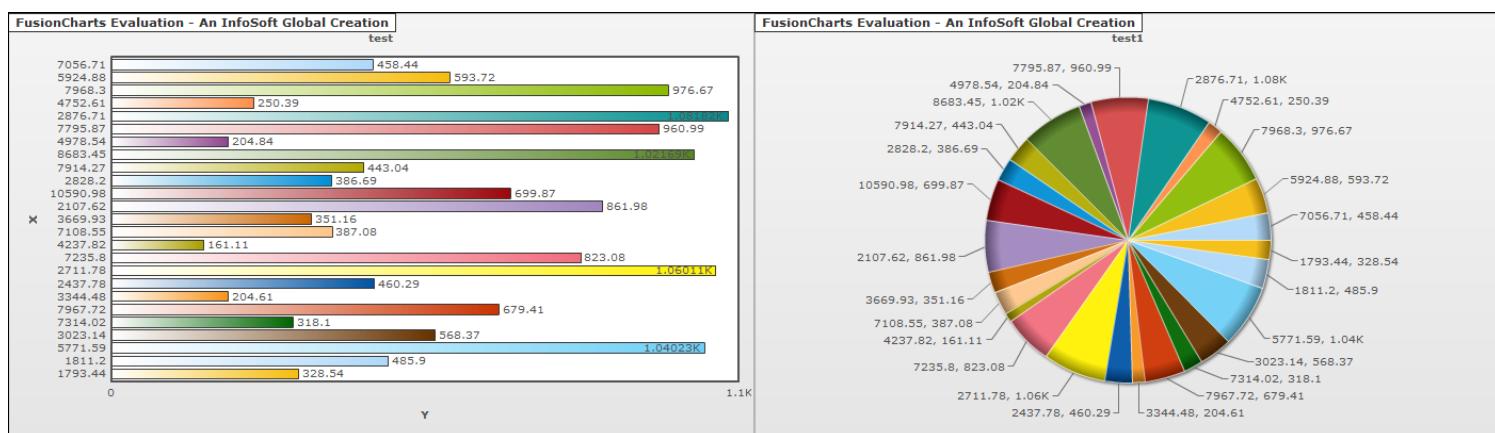
## Displaying Multiple Charts

You can configure multiple charts to display on the same form at runtime. This is a convenient way to compare data results in different formats on one page. Alternatively, you can display multiple charts using different data.



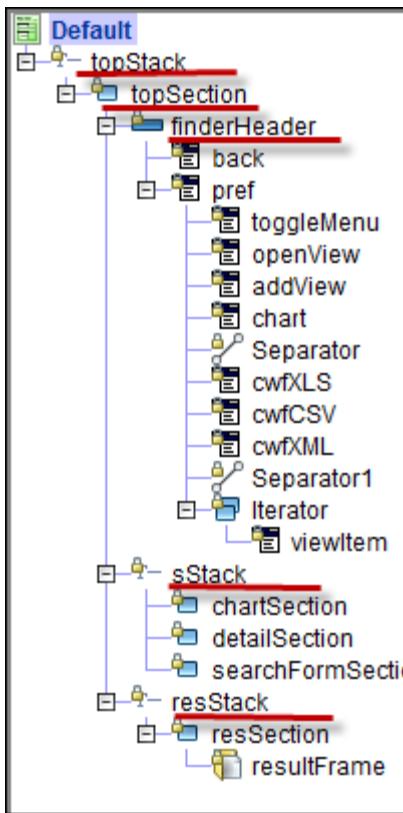
To configure multiple charts on a single page:

1. In the Finder User Interface, right-click the node and select **New Form**. Alternatively, you can use an existing Form.
2. In the **New Metadata Object** dialog, type the name of the new form.
3. In the **Form Element** pane, right-click the node and select **Add**.
4. From the Add Element dialog, select **Misc Controls > Form Frame**.
5. Continue by adding elements to configure the chart layout.
6. Configure the variable property for each Form Frame. This variable refers to a Chart Form element.
7. Configure the **Presentation User Interfaces** so that the newly created Form displays in the menu at runtime.



## Finder Default Form

The **Finder Default Form** is a base form that is pre-configured to display the Finder menus and information at runtime.



The Default Form has a number of sections, created by Form elements, which display specific information at runtime.

Menu Section	Description
(topStack) Section Stack	Section Stack is a container that allows for the grouping of Finder data or elements on a page in a logical order.
(topSection) Section	Creates a section within the Section Stack.
(finderHeader) Header	This Form element contains a series of menus for the Finder Preference drop-down list menu. <ul style="list-style-type: none"><li>The back menu element display the back button for the <a href="#">Preference Menu</a>.</li><li>The pref menu contains sub-menus that display the menu items for the Preference menu.</li></ul>
(sStack) Section Stack	This Section Stack element contains three subsections: <ul style="list-style-type: none"><li>chart section</li><li>details section</li><li>search search</li></ul> These sections are displayed in <a href="#">mutex</a> , which means that only one of these three sections is displayed at one time while the other two sections are hidden.  When the <b>Visible</b> property is set to <i>topSectionVisible</i> , the data for the section selected (chart, detail or search) is displayed. If there is no data to display, the detail section of the resStack (see the row in this table) takes 100% of the page.  See <a href="#">Additional Menus</a> for more information.
resStack	This Section Stack element contains a Section element that contains a Form Frame. The Form Frame displays

the results of the search using the output document of the Finder. The Form Frame references the string variable resultForm, which defaults to the Result Form selected in the Views tab of the Finder.

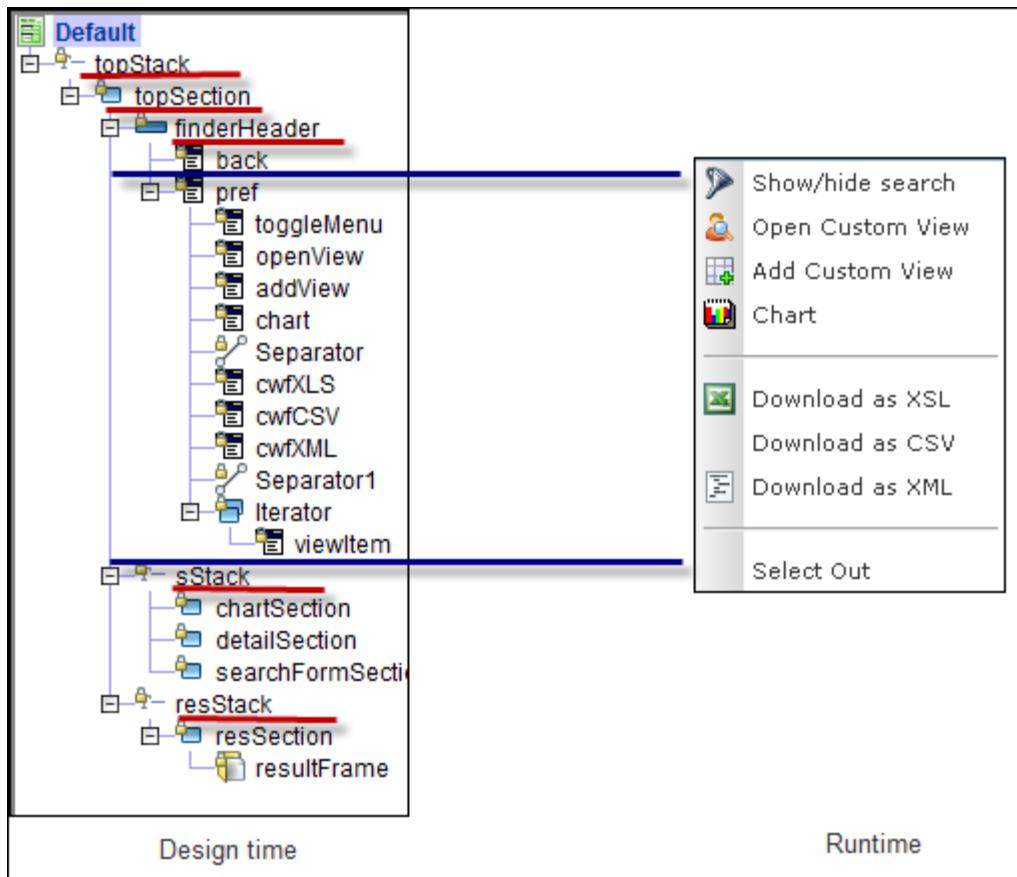
When the **Visible** property is set to resultVisible, content of the Form Frame is displayed.

See [Additional Menus](#) for more information.

You can customize the menu settings by overriding the Default Form and configuring the properties for each of the Form elements.

## Preference Menu

The Preference (pref) menu consists of several Form element menus, each representing an item in the Preferences drop-down menu displayed at runtime. For each menu element, the **Label** and **Tooltip** properties are editable. To customize all properties for each menu, the Default Form must be overridden.



The Form element menus map to the corresponding Preference menu item displayed at runtime. User Interface variables and methods are available to configure each menu element.

Design time Menu Name	Runtime Preference Menu Item	Description
toggleMenu	Show/hide Search	The toggleMenu menu displays (or hides) the Search Form. In the toggleMenu element, the <b>Click method</b> property is set to the <i>toggleSearchAction</i> method. In the toggleMenu element, the <b>Visible</b> property is set to the <i>toggleVisible</i> method which determines that the Search is visible at first display at runtime.
openView	Open Custom View	The openView menu finds a custom view for the Finder and creates the system Finder Favorite. It creates search criteria based on the current Finder information and performs a search based on search criteria (such as the Finder name) displaying the results in the <b>Favorite</b> dialog. It also sets the header of the Favorite Finder. In the openView menu element, the <b>Click Method</b> property is set to <i>findCustomView</i> . The following is an example of the script used in the <i>findCustomView</i> .

		<pre> var ui = new cwf_oe.favoriteFinder.UserInterface(null, this); var finder = ui.model; var search = finder.searchDocument; search.putValueByName("finderName", this.model.getName(), false); search.putValueByName("menuPath", null, false); ui.searchAction(); ui.header = 'Custom Views for ' + this.model.getLabel(); ui.parentFinder = this; return finder; } </pre>
addView	Add Custom View	The addView menu displays the Favorite document in a popup allowing you to add a favorite for this Finder.
chart	Chart	The chart menu displays the chart that was set in the Chart Form and renders it in the browser. In the chart menu element, the <b>Click Method</b> property is set to <i>showChart</i> . In the chart menu, the <b>Visible</b> property is set to the <i>hasChart</i> method. This method controls the visibility of the chart using permissions. If the default chart form is overridden then the <i>hasChart</i> returns <i>true</i> . The chart is displayed when the Chart menu item is clicked in the Preference menu.
Separator	menu separator (line)	Separates menu items.
cwfXSL	Download as XLS	The cwfXSL menu is used to download XSL data from the database to the file. The <b>Click Method</b> property is set to download to the specific download type.
cwfCSV	Download as CSV	The cwfCSV menu is used to download CSV data from the database to the file. The <b>Click Method</b> property set to download to the specific download type.
cwfXML	Download as XML	The cwfXML menu is used to download XML data from the database to the file. The <b>Click Method</b> property set to download to the specific download type.
Separator1	menu separator (line)	Separates menu items.
Iterator	Iterate	Iterates through the menu items defined in the <i>viewItem</i> menu.
viewItem	Represents the Finder's current view	The viewItem menu displays the list of views listed in the Preference menu. In the viewItem menu element, the <b>Click Method</b> property is set to <i>changeView</i> . For each menu item, the <i>changeView</i> method is called.

For more information about the Preference menu, see [Finders at runtime](#).

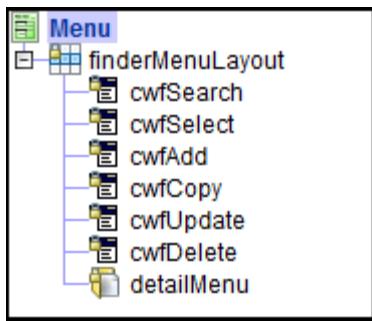
## Additional Menus

Form Element	Finder Section Item	Description
chartSection	Chart Section	This section displays the chart. By default, the chart menu is hidden in the user interface. In the chartSection section element, the <b>Visible</b> property is set to the <i>chartVisible</i> method setting the chart as visible.
detailSection	Detail Section	This section displays the Finder's Detail menu. In the detailSection section element, the <b>Variable</b> property is set to <i>detail</i> . The <i>detail</i> variable is of the User Interface type and displays the Detail menu. The <b>Visible</b> property is set to the <i>detailVisible</i> setting the section to visible. When the detail section is visible, it displays the back arrow at the top of the page, which is bound to the Finder User Interface <i>backvisible</i> method.
searchFormSection	Search Form	This section displays the Finder's Search menu. The <b>Variable</b> property of this menu is set to <i>search</i> . The <i>search</i> variable is of the User Interface type and displays the Search menu. The <b>Visible</b> property is set to the <i>ShowSearchForm</i> method setting the Search Form as visible. The

		Finder criteria are entered using the form that is specified in the Search Form field of the <a href="#">Finder View</a> tab.
resSection	Result Section	This section displays the Finder's Result menu. This menu contains a Form Frame Form element in which the result data is displayed.

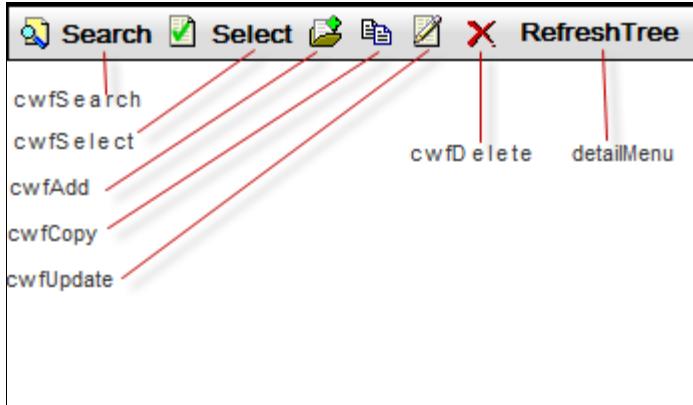
## Finder Menu Form

The **Finder Menu Form** contains menu actions, such as Search, Select and Add, Copy, Update and Delete buttons. By default this menu is pre-configured, but you can override the Form to customize its settings.



The *cwf* prefix for each menu indicates that they are framework objects, which are core system metadata objects.

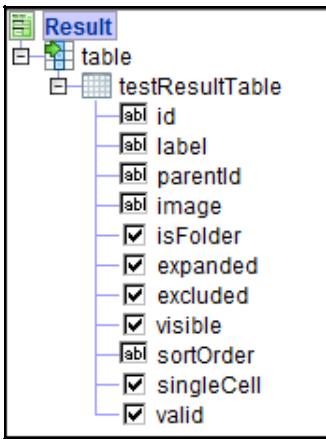
The *detailMenu* menu has its Form properties set to refer to the Finder detail variable of User Interface type. The detail variable refers to the Detail document of the Finder. When the user selects this Finder item the *showDetailAction* method is invoked.



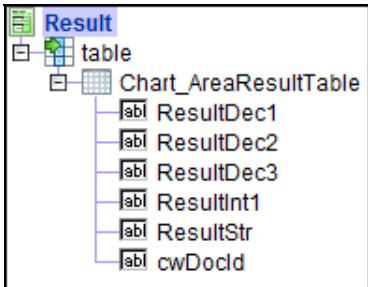
**Note:** Additional menu items can be added to reflect the desired functionality at runtime.

## Finder Result Form

The **Finder Result Form** specifies how the output Document is presented in the Finder User Interface. The Default Form element representation of the variables under the output document (that is, a boolean type variable is represented by a checkbox by default) are automatically added to the Result Form. The system populates the fields that are displayed under the table element using the variables defined in the User Interface. When the Finder output document is changed the system automatically changes the corresponding Result Form.



Each Form element in the Result Form can be configured using the Form element properties. For example, you can configure the Variable property to bind to a variable in the User Interface.



The variable references data in this Document and returns the value to the field which is displayed in the Result Form at runtime.

General	Variables	Methods	Mapping						
		Name	Type	Methods	Vis	Opt	Edit		
ResultDec1			Decimal, 10.2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
ResultDec2			Decimal, 10.2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
ResultDec3			Decimal, 10.2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
ResultInt1			Integer, 12	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
ResultStr			String, 20	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
cwDocId			String, 16	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

The result form is also used to download data related to the finder table. The selected download methods look in to the result form and attempt to find the [table element](#). But you cannot download the results when a finder's result form belongs to another user interface (no Table element related to the result form of the finder), and the [download](#) functionality of a finder user interface is overwritten. In this scenario, to specify which user interface to use to look up the result table, the download functionality can be used to [create download object](#) through application user interface script. The download object contains *resultUI* and *resultForm* variables that can be set in the script.

### Notes:

- For the Results data at runtime, the table column sort (ascending/descending) functionality is available at any time. You do not need to wait for all data for that particular Finder to have been retrieved from the server to perform a sort.
- If there are any validation errors in the user interface and the [table element](#) is used, the errors are shown as soon as finder is displayed at runtime.

## Edit In Table

The **Edit in table** option is available in the Table Form element. The **Editable** property defines if the user is allowed to edit the data of that Finder

through the Table element during runtime. The property provides a list of permission-based methods that set the permissions of the table. For each checkbox Form element in the Result Form, set the Toggle Edit property to true. If you have overridden the *OnSelChanged* method (from the Methods tab of the User Interface) ensure that the super implementation is called or the toggle edit functionality will not function properly.

## Select All Visible Rows in a Finder

You can select all visible rows of a finder to action. You can use the following two script methods:

- `selectAllRecords()`, which selects all fetched records in the table
- `deselectAllRecords()`, which deselects all records in the table

### Notes:

- The `selectAllRecords()` method respects any user-applied filters
- Any changes made to the finder result after selecting all records must occur in a separate user action

The following is an example on how to use the two methods:

1. Create a finder with a filter, which has some records in the finder result.
2. Create a user action containing `this.result.selectAllRecords` script. Proceed to bind this user action to a **Select All** menu.
3. Filter the finder result and then click the **Select All** menu. All filtered records are selected.
4. Click the **Delete** menu to delete the selected records. Only filtered records are deleted and other records remain in the table result.

## Browser-side Filtering

You can provide the results of browser-side filtering to the finder. The finder's current list continues to represent the full finder results. A new method, `filteredList()`, is provided to expose the filtered list.

The following is an example on how to use the `filteredList()` method:

1. Create one finder table bound to the `result` variable.
2. In the same form, create another table bound to a variable and call it **filtered**.
3. Create a button in the same form that has a click method with similar script functionality. The following is an example for reference:

```
var f = this.result.filteredList;

this.filtered = new DataObjectList();
if(this.result.filteredList!=null && this.result.filteredList.length > 0){
    var i = 0;
    for(i=0;i<f.length;i++){
        this.filtered.addLast(f[i]);
    }
}
this.filtered.updateList;
```

4. At runtime, click **Search**, add a filter value, then click the **Filter** icon.
5. Click the button that you created in step 3. The second table bound to the `filtered` variable contains the same records as the table from step 1.
6. Click **Search**, and then click the button from step 3. The table from step 2 is empty because there is no filter set.

For any item in the current list, the `isFiltered()` method is available that indicates whether the item is in the filtered list. The following is an example on how to use the `this.result.isFiltered()` method:

1. Create an integer variable under the Finder User Interface.
2. Using the form in the previous example, add a textfield bound to the integer variable from step 1.
3. Add a button with a click method that has similar script functionality. The following is an example for reference:

```
Global.showUserMessage(this.result.isFiltered(this.result[this.Integer]));
```

4. At runtime, click **Search**, add a filter value, then click the **Filter** icon.
5. Enter a row number, and then click the button from step 3. This button displays a popup showing either true or false, depending on your filter settings.

## Return the Result Form Dynamically

The `ui_common.baseFinder` base UI for finders allows for returning the result form dynamically. To use this functionality, ensure that you do the following:

- Your metadata overrides the `getResultSet` method of `ui_common.baseFinder`.

- The getResultForm method returns the top-level UI (that is, name) that extends ui\_common.baseResult. As an example, for cwa\_admin.findEventLog, the getResultForm method publishes an event to get the result form UI (top-level UI), so that your metadata can provide its own handler and return the extended UI.
- The getResultForm method can have any custom logic. As an example, it can be used to dynamically choose what result form to display based on the first row from the result list.

**Note:** The getResultForm method is not invoked every time when a search occurs. It is usually called once when changing finder views.

## Finder Search Form

The Finder Search Form is, by default, one of the Forms that appears in the Finder User Interface. The Search Form contains one Form Frame Form element. By configuring the Form Frame properties you can define the runtime behaviour of the Search functionality. By default, the Variable property of the Form Frame is configured to bind to the system-generated *search* variable of User Interface type. The *search* variable returns the content of the Search document. Use this Search Form when you want to search for data, but do not need to display the results in the Result Form (for this behavior use the Default Form). For example, if you want to perform a search to display the child objects of a Finder, you can setup the Search Form to return this information. In addition, this Search Form is useful when performing a search on migrated data.



### Set onEnter Property on Form Frame to Display Search Results

The following example describes how to enable the **Enter** key to display the search results after you enter your search criteria in a [navigation tree](#) finder.

1. Under the finder's **UserInterface**, locate **Search**.
2. Right-click **Search** and select **Override** from the menu.
3. In this Search form, right-click the **Form Frame** element and select **Copy & Replace...** from the menu.
4. Set the **onEnter** property to **searchAction()**, to allow the finder to refresh when pressing the **Enter** key when using the navigation tree.

## Finder Pagination

---

Finder pagination allows you to display the amount of data you want on your page, such as setting the page size in the application.

### Finder User Interface Methods

The following are finder user interface methods that you can use to control the amount of data on a page:

Method	Description
fetchNextPage	If the finder user interface's result variable currently has a value and there is more data to be displayed, use this method to increase the page count.
fetchPreviousPage	If the finder user interface's result variable currently has a value and there is previous data to be displayed, use this method to decrease the page count.
getCurrentEndRow	This method gets an index value of its last row on the current page.
getCurrentStartRow	The method gets an index value of its first row on the current page.
getDataPageSize	This method gets the size of the current page.  <b>Note:</b> To set the page size, call the following finder user interface method (not result):  <code>finderUI.setDataPageSize(number)</code>
hasNextPage	If this.cwDataPageSize is set, this permission can be used determine whether to disable or enable the <b>Next</b> button. When a next page exists, the <b>Next</b> button's disabled property is set to false. If a next page does not exist, the <b>Next</b> button's disabled property is set to true.  <b>Note:</b> To determine the true value of hasNextPage(), use this.result.hasNextPage() in a separate script.
hasPreviousPage	If this.cwDataPageSize is set, this permission can be used determine whether to disable or enable the <b>Previous</b> button. When a previous page exists, the <b>Previous</b> button's disabled property is set to false. If a previous page does not exist, the <b>Previous</b> button's disabled property is set to true.  <b>Note:</b> To determine the true value of hasPreviousPage(), use this.result.hasPreviousPage() in a separate script.
next	This method advances to the next page.
previous	This method goes to the previous page.
setDataPageSize(int size)	This method sets the amount of data to be displayed in the user interface at any given time. When this method is called, the finder table data list refreshes to show the new data size.

### DataObjectList Methods

The following are finder DataObjectList methods for finder pagination:

Method	Description
fetchNext	The fetchNextPage method usually calls this method to increase the current page number to be displayed in the user interface.
fetchPrevious	The fetchPreviousPage method usually calls this method to decrease the current page number.
firstPage	This method goes to the first page.
getCurrentPage	This method returns the current page number.

getTotalPages	This method returns the total number of pages.
hasNext	This method checks whether there is enough data for a next page.
hasPrevious	This method checks whether there is previous data to be displayed.
lastPage	This method advances to the last page.
setCurrentPage(number)	This method sets the current page number. If the number is greater than the total number of pages, this method sets total - 1.  <b>Note:</b> The current page number starts at 0.

#### Notes:

- The fetchNext and fetchPrevious methods only work if cwPageSize has been set. Otherwise, the finder works with the product configuration (that is, normal paging).
- Once cwPageSize is set, clicking the **Search** button of a finder uses its value.
- At all times, if the setDataPageSize method is called, the result on the user interface updates to show the new maximum data count. Setting cwPageSize does not refresh the table.
- Calling the fetchNext method automatically calls searchAction if the result variable does not already have a value.
- Calling the fetchNext and fetchPrevious methods when there is no data to show keeps the current page viewed on the user interface (for example, the table does not display as being empty when there is no data left to show).
- For finders with paging capabilities, the ability to sort the entire set of results, and not just the results visible on the page, is available.

## Application of Methods

This section shows how to use both finder user interface and DataObjectList methods for finder pagination.

### Example 1: Create two buttons and set the dataPageSize

- In Velocity Studio, create a finder with two buttons:
  - Previous
  - Next
- In the finder's **UserInterface**, locate the onInit method and set the dataPageSize (for example, 10).
- Verify that the number of rows returned is correct.

### Example 2: Create menus that change the amount of data displayed

- In Velocity Studio, create a finder with two buttons:
  - Previous**
  - Next**
- In the finder's **UserInterface**, create three menus with the following labels:
  - <u>10</u>
  - <u>20</u>
  - <u>30</u>
- Create click methods for each of these menus. As an example, the menu with the <u>10</u> label has a click method of this.setDataPageSize(10).
- Check that each menu changes the amount of data shown on the screen.

### Example 3: Set the **Disabled** property to the hasPrevious and hasNext methods for the **Previous** and **Next** buttons, respectively

- In Velocity Studio, for the **Previous** and **Next** buttons, set the **Disabled** property for each button to **hasPrevious** and **hasNext**, respectively.
- Verify that when there is no data to show after the current data has displayed, that the **Next** button is disabled.
- Check that on the user interface's first load, that the **Previous** button is disabled.

## Finders at Runtime

Finders are used to search, access, add, update, and delete data from different data sources and display it at runtime. There are several types of Finders available including Document, Order, Rule, Script, and SQL. Each finder displays different type of information at runtime. Using the available Finder User Interface features (configured at design time) end users can customize the display of the data.

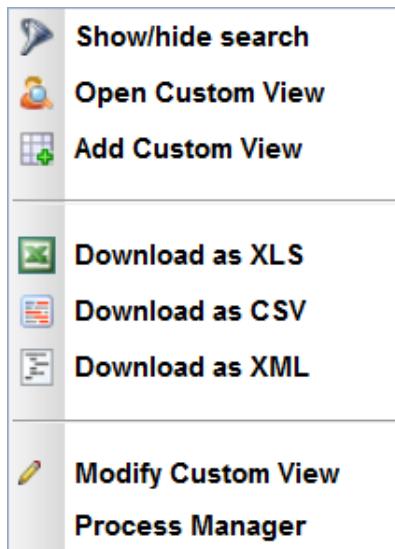
In most cases, when displaying a finder user interface, properties such as the search form to display, the initial search data, the result form to use, and so on are taken from the first available finder view as defined in Velocity Studio. A finder's custom view changes the way the finder user interface looks on top of that static view. If a custom view is set as the default view, it is used to render the user interface as defined in that custom view and run the finder automatically if the corresponding option is on.

The runtime Finder data can be customized using the following menus and features:

- [Preference Menu](#)
- [Table Features](#)
- [Editing Table Data](#)
- [Displaying a Hover Form](#)

### Preference Menu

In Velocity Studio, the Default Form of the Finder User Interface contains Form elements that are, by default, configured to display a Preference menu at runtime. At runtime, the Finder contains a Preference drop-down menu contains a number of options that enable the user to drill down on data or change how it is displayed in the Finder User Interface. The following section discusses the options and how they works, including some important design time configuration information.



#### Show/hide Search

This menu option controls if the detailed information in the Finder is displayed or hidden. In the Finder Default Form, the `toggleMenu` menu element contains a `Visible` property that sets the menu as `toggleVisible` or `toggleHidden`.

#### Finder Name

The name of the current Finder displays in two places, in the header of the Finder and on the lower region of the **Preference** menu. When the user selects and opens a specific View, the name of the selected view appears on the top of the Finder dialog and on the Preferences dialog as a label. The name of the view is stored as a variable called `header` in the Variables tab in the Finder User Interface. For more information see [Finder User Interface Variables](#).

The screenshot shows a Velocity Studio interface for a 'Case5FinderView'. At the top left is the view title 'Case5FinderView' with a red circle around it. Below it is a search bar with a 'Customer Name' input field and a 'Search' button. To the right is a context menu with options: 'Show/hide search', 'Open Custom View', 'Add Custom View', 'Download as XSL', 'Download as CSV', and 'Download as XML'. The bottom right of the menu also has a red circle around the text 'Case5FinderView'.

When the Finder User Interface menu is initialized, the list of Finders is generated. In Velocity Studio, the Finder Default form contains an Iterator element that iterates through the objects specified in the Menu element, which is located beneath it. The Menu variable property is set to `view--` which refers to the name of the Finder view that displays in the Preference menu at runtime.

### Custom View

The **Open Custom View** menu option enables end users to open a custom view for the current Finder. The custom view enables users to search, select and save particular records of a Finder. In the openView menu, the *Click method* property controls the display of the view. To change a custom view, you would have to open a custom view and then click the Edit menu option.

Once a custom view is created, it can be changed by using the **Modify Custom View** menu option, which allows you to edit the custom view. The `customViewFinder` script is used for this menu's operation, as it contains SQL to retrieve information from two separate database tables.

### Favorites

The **Add Custom View** menu options enables end users to add and customize a Favorites view. The Favorite dialog enables the user to save Finder settings as a favorite record (that displays in the Favorite menu). To access the Favorites view, the user must view Finder results.

The screenshot shows the 'Favorite' dialog box. It has a title bar 'Favorite' and a main area labeled 'Favorite'. Inside, there is a 'Label \*' input field containing '1', a 'Max Rows' input field containing '1,000', a 'Shared' checkbox, and two checked checkboxes for 'Show search form at start' and 'Run query at start'. At the bottom are 'Save' and 'Cancel' buttons.

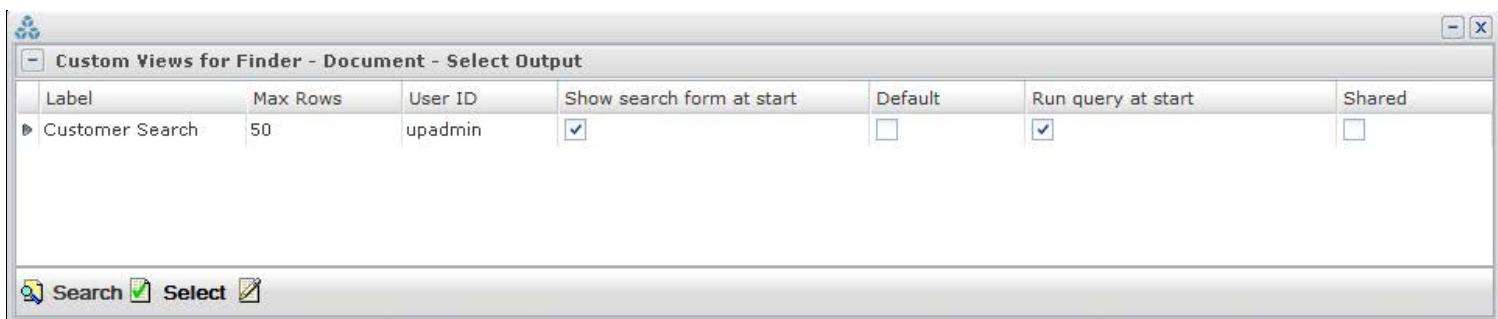
The **Favorite** menu allows the user to store, change, and select a set of parameters for a given Finder.

This set may include some of the Finder search criteria, the resulting columns, the sort order and the maximum number of rows. Each set has a unique name and is called *Favorite* or *Favorite record*.

The menu contains the following menu items:

- **Add Custom View** - allows you to change the favorite parameters for the current favorite record.
- **Open Custom View** - enables the user to search, select and save particular records of a Finder. This data can be viewed in the Favorites menu.
- **Modify Custom View** - allows you to change the current favorite view's parameters.

If the Finder is not opened using the **Open Custom View** dialog or as a default Favorite, or if the user that has created the favorite record is not the current user, the operation is not performed. If the Finder is opened using the **Open Custom View** menu, the **Favorite** dialog (refer to the [Add Favorite](#) section) opens and allows the user to change the the favorite parameters for the current favorite record.



The new favorite record is stored with the current Finder settings and replaces the existing record.

Favorite Menu Item	Function
<b>Label</b>	The name of the <b>Label</b> . Note, that the favorite <b>Label</b> parameter cannot be changed once it is entered.
<b>Max Rows</b>	Saves the current Finder result to the database with the <b>Label</b> field used as the identifier. Maximum number of rows to retrieve. The default is the same number of rows as specified for <b>Page Size</b> . The maximum rows allowed is 999. Setting the maximum number of rows avoids caching memory.
<b>Default</b>	Open the corresponding Finder user interface using the properties set in this custom view by default.
<b>Show search form at start</b>	Opens the selected Favorite and displays the report on screen.
<b>Run query at start</b>	Searches the database for existing Favorites.
<b>Save Search Criteria</b>	By default, this field is selected, meaning that the finder's search criteria are saved for custom views and favorites. Otherwise, when this field is not selected, the search criteria are cleared when you leave your finder results.
<b>Save Table State</b>	By default, this field is selected, indicating that you want to save the table layout for your custom view. This field is always visible.

## Chart

The **Chart** options of the Preference menu displays information that is configured in the Finder Chart Form. The Finder Chart Form consists of a Chart Form Element and a ChartField. When these elements are configured, they refer to a result Form, and display the content of the form in a chart display. For more information see, [Finder Chart Form](#).

## Download Files

In the Preference menu there are three options for downloading files:

Downloading Option	Function
Download as XSL	To import data from a XSL file click the <b>Download as XSL</b> button. The Download as XSL dialog opens where the file is selected. The download only contains the current page information.
Download as CSV	To import data from a CSV file click the <b>Download as CSV</b> button. The Download as CSV dialog opens where the file is selected.
Download as XML	Open a previously-saved favorite record that opens the corresponding Finder in the screen

## Move Table Columns

At runtime, it is possible to move an entire results column. By right-clicking the column header and holding down and moving the mouse, the user is able to reposition the entire column to either left or right.

## Display Number of Retrieved Rows After Conducting a Search

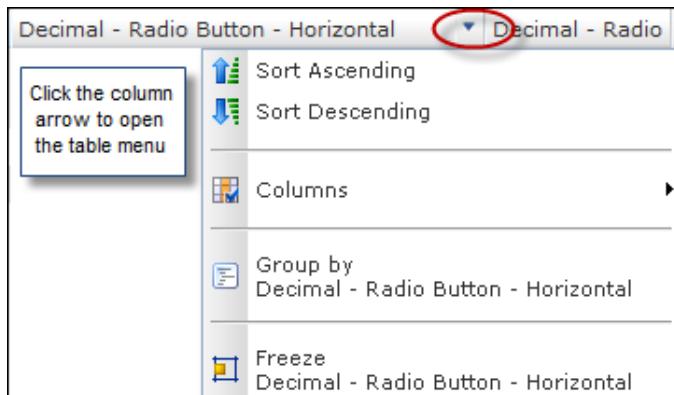
The number of retrieved rows after conducting a finder search appears in the status bar of your Web browser.

## Table Features

At runtime, data from the Finder User Interface Result Form is displayed in a table. The user can customize the table using several of the table options.

### Table Menu

Each column in the table displays a menu when the column arrow is selected.

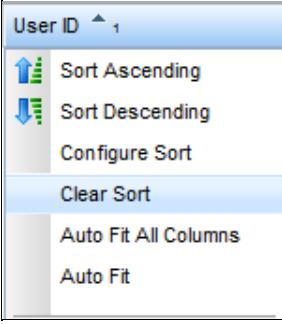
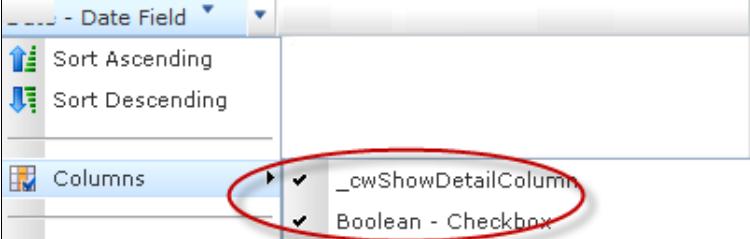
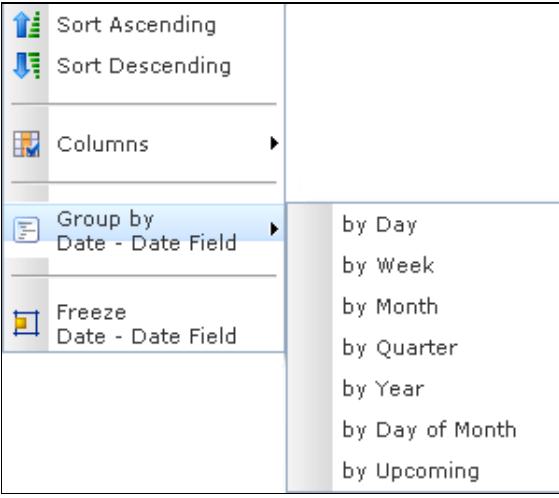
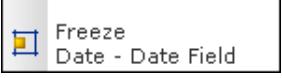
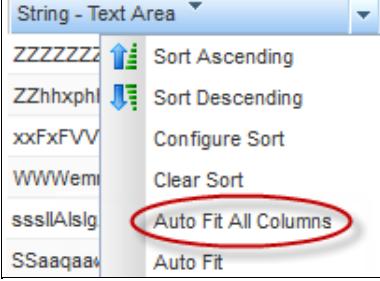


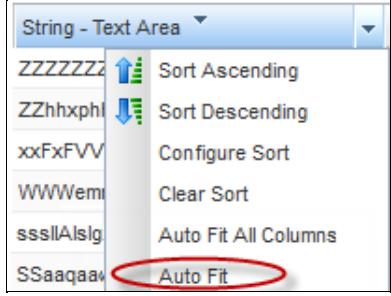
**Note:** In Velocity Studio, the display of the table is bound to a Finder User Interface variable called *tableState*. The *tableState* variable is set to the current view state of the table.

The following options are available for each column in the table.

Option	Function	Example
<b>Sort Ascending</b>	This menu item arranges column data into a sequence from the lowest to the highest values.	<p>A screenshot of a date field sorted in ascending order. The dropdown menu shows "Date - Date Field". The list contains four dates: 1/15/2030, 10/27/2029, 8/8/2029, and 3/2/2029.</p>
<b>Sort Descending</b>	This menu item arranges column data into a sequence from the highest to lowest values.	<p>A screenshot of a date field sorted in descending order. The dropdown menu shows "Date - Date Field". The list contains four dates: 1/15/2030, 10/27/2029, 8/8/2029, and 3/2/2029.</p>
<b>Configure Sort</b>	Selecting this option launches the Sort dialog, allowing you to specify the order in which your table columns are sorted.  You can perform the following actions: <ul style="list-style-type: none"><li>• Add a sorting level by clicking the <b>Add Level</b> button. Click the <b>Column</b> drop-down menu and select the</li></ul>	<p>A screenshot of the Sort dialog box. It has tabs for "Add Level", "Delete Level", "Copy Level", and sorting arrows. The main area shows a table with columns "Sort by" and "Order". Under "Sort by", "Customer Name" is selected with "Ascending" order. Under "Then by", "Account Category" is selected with "Ascending" order. Other options like "Contact Name", "Account Number", "Telephone", and "Mailing Address" are listed in the dropdown. There are "Apply" and "Cancel" buttons at the bottom.</p>

	<p>column name, and then click the <b>Order</b> drop-down menu to specify whether to sort the data in either ascending or descending order.</p> <p>Numbers appears in the column headings, indicating the order in which the columns are sorted.</p> <ul style="list-style-type: none"> <li>• Remove a sorting level by selecting a level from the table and then clicking the <b>Delete Level</b> button.</li> <li>• Copy a sorting level by selecting a level from the table and then clicking the <b>Copy Level</b> button.</li> <li>• Move a sorting level up or down in the sorting configuration by clicking the <b>Up</b> and <b>Down</b> arrow buttons, respectively.</li> </ul> <p>When you have finished configuring your sort, click the <b>Apply</b> button to save your changes.</p>
<b>Clear Sort</b>	This menu item allows you to clear your previously configured sort.

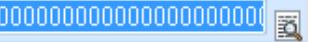
		
<b>Column</b>	This column item hides or shows column data.	
<b>Group By Name</b>	<p>This menu item groups results based on the value of the column. For example, if the column is set by a datatype, then the data is grouped by calendar specifics.</p> <p><b>Note:</b> For Translation Fields (Translation Form element), the <i>Group by In Table</i> property field must be set.</p>	
<b>Freeze Frame</b>	The menu item freezes the indicated table so that it remains in place and visible when horizontal scrolling occurs. The column that is frozen is moved into the first column position (far left of the table). After the column is frozen, you can select the unfreeze option to return the column to its previous state.	
<b>Auto Fit All Columns</b>	This menu item ensures all data automatically fits within all table columns.	

<b>Auto Fit</b>	This menu item ensures all data automatically fits within the specified table column.	
-----------------	---	---

## Editing Table Data

The user can edit data in the table when the Table element property called *Editable* is set to *True*. Setting the *Editable* property to true enables the user to edit the fields of the table at runtime. There are two ways in which users can edit table data; click the row that contains the data that the user want to edit or click the  arrow icon of the row that the user want to edit. Clicking the icon makes the row editable and is the same action as a double-click.

When a field of a table is editable, do the following:

Form Element Type	Element in Table	Edit Action
Text Field	<input type="text" value="55.33"/>	Type text in text field.
Text Area	see Text Field	Type text in text field.
Large Text		Click the Large Text icon. A Finder opens for this data. In the Finder, select a row of data. The data that the user selects replaces the data that currently appears in the field. Scroll to the bottom of the Finder and click <b>Select</b> . The current data is replaced by the selected data in the Finder. Note, the <i>Editable</i> property of the Large Text Form element must be set to <i>True</i> .
Large Text Area	see Text Field	Type text in text field.
Select	<input type="button" value="22"/>	Select text from the drop-down menu list.
Date Time	<input type="button" value="6/7/2005"/>	Click the calendar icon. A calendar selector displays from which the user can select the new data and time.
Date	<input type="button" value="6/7/2005"/>	Click the calendar icon. A calendar selector displays from which the user can select the new data.
Reference	<input type="button" value="436"/>	Click the Reference icon. A Finder opens for this data. In the Finder, select a row of data. The data that the user selects replaces the data that currently appears in the field. Scroll to the bottom of the Finder and click <b>Select</b> . The current data is replaced by the selected data in the Finder. Note, the <i>Editable</i> property of the Reference element must be set to <i>True</i> .
Translation	<input type="button" value="This is translated text."/>	Click the Translation icon. A Finder opens for this data. In the Finder, select a row of data. The data that the user selects replaces the data that currently appears in the field. Scroll to the bottom of the Finder and click <b>Select</b> . The current data is replaced by the selected data in the Finder. Note, the <i>Editable</i> property of the Translation element must be set to <i>True</i> .
Hyperlink	see Text Field	Add a URL.
Upload File	see Text Field	Type the path and name of the file to be uploaded.
Variable	see Text Field	Type the name of the variable.
Image	see Text Field	Type the path and name of the image.
Label	see Text Field	Type the name of the label.
Checkbox	<input checked="" type="checkbox"/>	Check or uncheck the checkbox. Note, the <i>toggle edit</i> property of the Checkbox Form element must be set to <i>True</i> .

## Editing a Table Row

Table rows can be added, edited, deleted and copied using the table edit control located at the bottom of the table.

Table Edit Icon	Table Edit Action
	Add
	Copy
	Edit
	Delete

## Displaying a Hover Form

In Velocity Studio, when the hover properties of the Table element are configured, the user can display a document as a hover form in the Finder at runtime. At runtime, when the user hovers on a row, the form displays beside the cursor.

**Note:** There is a short delay (2 seconds) between the time the cursor is set on a row and the display of the form. This value may be overridden by adding a variable called [Hover Delay](#) into your design time configuration. Acceptable values for this variable are between 100 and 5000 milliseconds. Any numbers above or below this range are ignored and the default 2000 (2 seconds) is used.

There are several ways in which to display a hover:

- [Hover form with local form](#)
- [Hover form with variable](#)

For more information see [Table element](#).

## Finder Custom View and Export

The finder user interface allows you to add custom views and export favorites for the finder. When creating a custom view, it stores the current search and result forms, the table state (for example, width, visibility, and order of columns), search criteria, and other parameters to the favorite document.

You can then open the custom view through the favorites finder, and change or delete it. If the view is marked as the default, it is loaded automatically when creating and displaying the finder user interface.

The favorites finder can also be displayed as a global finder. In this case, it shows all custom views of all finders, so you can select and open any finder for any page content. See [export](#) and [custom view](#) options on how this functionality can be used in your application.

### Metadata Objects

The following table describes metadata objects pertaining to custom views and favorites:

Metadata Object	Description
ui_common.favoriteFinder	This finder displays favorites and custom views. It also displays the old cwf_oe.favoriteFinder.
ui_common.favorite	This object represents the favorite document. It does not have child UI. Instead, use the top-level ui_common.favoriteUI.
ui_common.favoriteSearch	This object is the search document for the finder.
ui_common.customViewFinder	This object is an SQL finder that searches for custom views, to replace cwf_oe.customViewFinder. It does not have a child UI and is only used to get data.
ui_common.favoriteResult	This object denotes the top-level UI for the favorite finder's result table.

### Scripts

The following table describes scripts pertaining to custom views and favorites:

Script	Description
ui_common.addFavorite	This script adds a new favorite record based on the specified finder UI state.
ui_common.addToFavorites	This script gets the current finder (that is, application page content) and adds the favorite record for it.
ui_common.checkFavoriteLabel	This script returns true if the label is already used by other favorite, except for the favDoc specified.
ui_common.deleteFavorite	This script deletes the favorite document with the specified ID.
ui_common.getDefaultFavorite	This script gets the default favorite view for the current user, application, and finder specified
ui_common.getFavorites	This script gets custom views for the current user and application. If the finder name is specified, it returns views for the finder only. If the finder name is null, it returns all found favorites.
ui_common.loadDefaultFavorite	This script loads the default favorite for the specified finder UI.
ui_common.loadFavoriteRecord	This script loads the specified favorite using both the specified finder and finder UI. If the finder UI is null, this script creates and returns a new finder UI.
ui_common.openFavorite	This script opens the specified favorite record to the specified finder UI. If the finder UI is null, the script creates a new UI and sets it as the application page content.
ui_common.processDefaultFlag, resetDefaultFavoriteID, setDefaultFavoriteID	These scripts manage the <b>Default</b> flag for the favorite record.

### Events

The following table describes events pertaining to custom views and favorites. Parameters that are bolded signify that they are mandatory:

Event	Parameters	Description
UI_ADD_FINDER_VIEW	<b>finderUI</b>	This event creates a new favorite view based on the current finder UI state. It returns the ui_common.favorite document.
UI_GET_FINDER_VIEWS	<b>finderName</b>	This event returns the list of custom views for the specified finder.
UI_LOAD_DEFAULT_FINDER_VIEW	<b>finderUI</b>	This finder returns true if the default view has been found and has been loaded to the specified finder UI.
UI_OPEN_FAVORITE	<b>• favDoc</b> <b>• parentFinderUI</b> (that is, the finder in which this favorite should be loaded; it can be null, meaning that a new finder UI is created)	This finder loads the favorite record into the specified parentFinderUI. If parentFinderUI is null, the event creates a new UI and sets it as the application's page content.

## Scripts

---

A Script is a metadata object that contains a JavaScript, which can be invoked by the metadata object's name. Parameters and Return Type of the JavaScript function can be defined in the Script as well.

The Script can be called from any other script anywhere in the application metadata. When calling a JavaScript script the format is as follows: <namespace\_name>:<script\_name> (the name of the namespace followed by a colon (:) and then the JavaScript script name).

To differentiate scripts that are created within **Methods** of metadata objects, these Scripts are commonly referred to as *top-level scripts* or *global scripts*. The Script can also be used in the System Administration application to create a script recurrence in the [Calendar library](#). To create a Script, do the following:

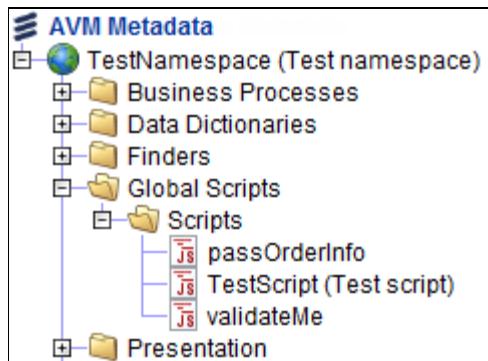
In the **Metadata** tab of **Navigation** pane, either:

1. Right-click a Namespace. Select **New ...**
2. **New Metadata Object** wizard appears. Expand the **Global Scripts** node, select **Script**, and then click the **Next** button.
3. Select one from the following types of scripts, and click the **Next** button.
  - o **General**
  - o **Action Auditor**
4. Enter the name of Script (must conform to JavaScript naming conventions), and other parameters (see [Script general properties](#) for description). Then, click the **Finish** button.

OR

1. Right-click the **Script** folder or the **Global Scripts** folder in a Namespace, if it is present. Select **New Script**.
2. Follow step 2 mentioned previously.

After the Script is created, the newly created Script appears under the **Scripts** folder.



Right-clicking a script shows a number of actions that you can perform on your script, including [Break on first line](#), [Show usage](#), and [Find in Scripts](#) for interface operations. See [Right-click Popup Menu](#) for a list of common actions that you can perform.

## Script General Tab

The general Script properties are defined on this tab.

The screenshot shows the 'General' tab of a script configuration window. The fields include:

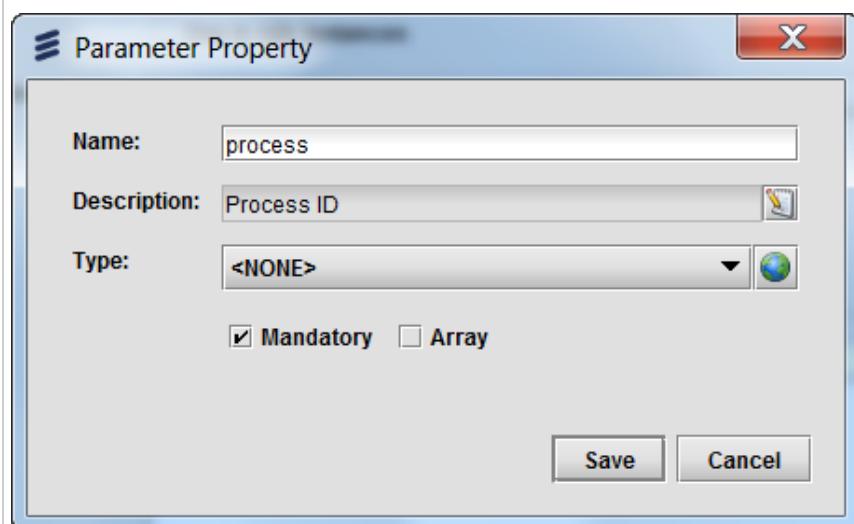
- Name:** createAccounts
- Label:** createAccounts
- Description:** (empty)
- Overrides:** <NONE>
- Parameters:** A table with columns Name, Type, and Description. It contains no rows.
- Script:** A code editor containing the following JavaScript function:

```
function createAccounts(){
    var ord=new worklistNS.custOrder();
    //var ord = new Order ("worklistNS:custOrder");
}
```
- Return:** com.conceptwave.system.Void
- Return Description:** (empty)
- Feature Restriction:** (checkbox checked)

The following table describes the fields for the **General** tab of the Script:

Field	Description
<b>Name</b>	Name of the metadata object (used in scripts; must conform to JavaScript naming conventions). Read-only after creation. To change it, use the <b>Rename</b> command in pop-up menu by right-clicking the metadata object.
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).  <b>Note:</b> Copying a restricted script from a library is not allowed. Additionally, copying a script from a library namespace that is restricted is also not permitted.
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Virtual</b>	If checked, it provides the ability to create methods in template metadata that does not have implementation, but can be used in template script and be bound to UI elements. The implementation is done in user metadata (see <a href="#">Virtual Field</a> for more information).
<b>Label</b>	Metadata object display label.
<b>Description</b>	Description of the metadata object for documentation.
<b>Overrides</b>	Metadata object to override. Typically used to customize system or library metadata objects; for example, scripts defined in the library metadata <i>Process Management Product</i> can be overridden to extend functionalities required in your application. At runtime, the Script specified in <b>Overrides</b> field is completely replaced by this Script. A search for the overridden Script will return this Script as the result.
<b>Use in rule instances</b>	If checked, the script is available for use in instance expressions of Rules (that is, inserting Functions).
<b>Parameters</b>	Table to define script parameters. Click the <b>Add</b> or <b>Remove</b> button to add or remove parameters, respectively. The

order of the parameters shown in this table is the order of parameters upon script invocation. Click the **Edit** button with a selected parameter to edit a parameter.

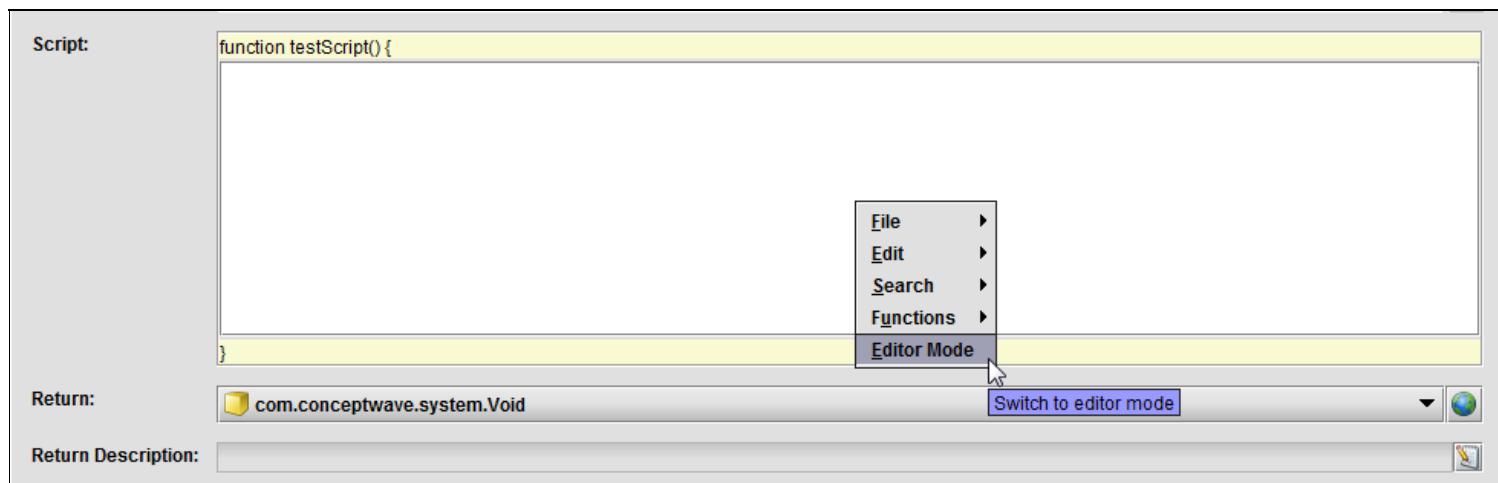


Field	Description
<b>Name</b>	Mandatory. Name of script parameter.
<b>Description</b>	Click the <b>Pencil</b> icon and describe your parameter in the field. Click the <b>OK</b> button to save your changes.
<b>Type</b>	Data Type of the parameter.
<b>Mandatory</b>	If checked, the parameter is mandatory and the expression editor will not allow the parameter value to be null.
<b>Array</b>	If checked, the parameter is an array of the specified Data Type.

<b>Script</b>	Field containing the JavaScript script to be executed.
<b>Return Type</b>	The Object type to be returned by the script. Choose <i>com.conceptwave.system.Void</i> if no value shall be returned.
<b>Return Description</b>	This field allows you to describe the Script Object's return type.
<b>Feature Restriction</b>	This field allows you to specify <u>features that you want to restrict</u> . Click the field's <b>Checkbox</b> button ( <input checked="" type="checkbox"/> ) to launch the Select Feature dialog. You can select all features that you want to restrict and then click the <b>Save</b> button.

Notice that, in the **Script** field, Velocity Studio automatically generates the function statement, as well as the opening and the closing brackets of the function block. The name of the function is the **Name** of the script, and the function parameters, if any, are added corresponding to **Parameters** of the script. You only need to insert segments of JavaScript code within the function.

In the **Script** text input area, you may right-click to pop-up a context menu:

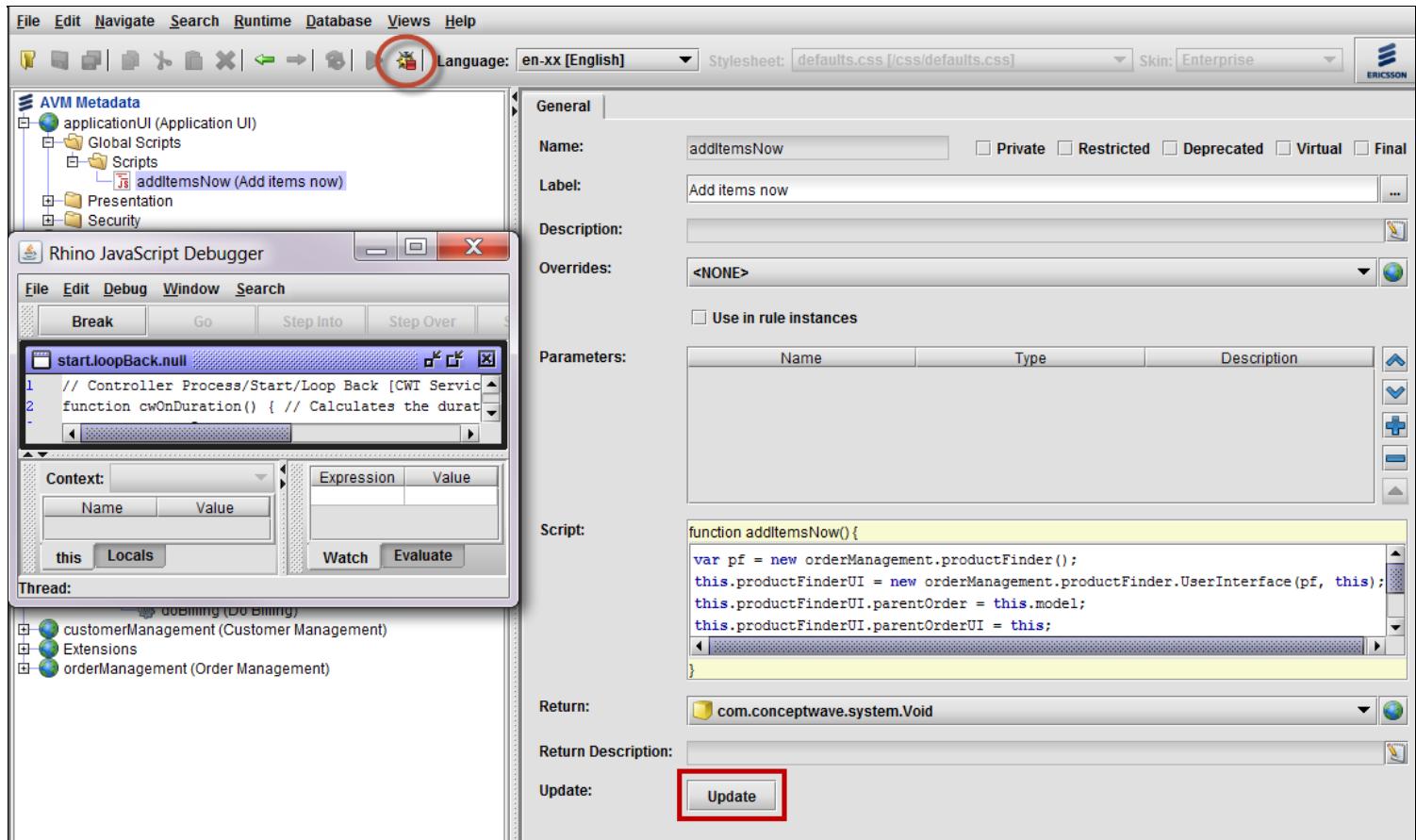


Select the **Editor Mode** option from the pop-up menu to open the **JavaScript Editor** for the script. You can also double-click the **Script** field to launch the **JavaScript Editor**.

**Note:** The other items in the pop-up menu are used for editing script contents directly in the field, rather than using the **JavaScript Editor**.

### Save Script Changes in Runtime or Debug Mode

When you are in either runtime or debug mode, you can make changes to your script and then click the **Update** button, which updates the runtime version of the script and saves your changes.

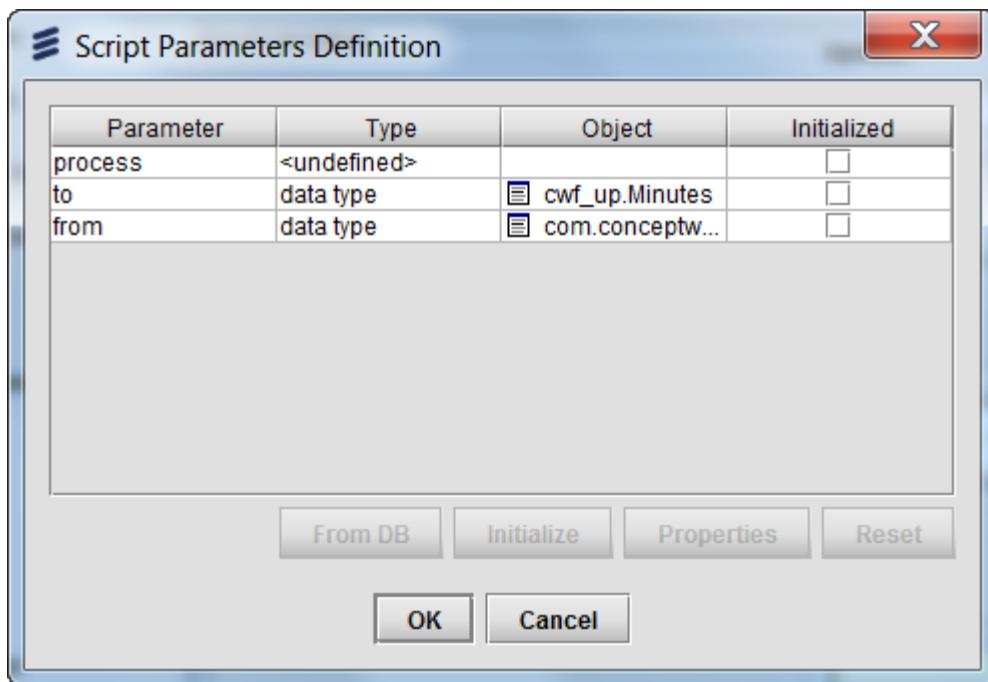


**Note:** The **Update** button is only visible when runtime is active.

## Script Operations

An individual Script has a **Debug Script** item in its right-click pop-up menu in addition to the common right-click menu items. When selected, this menu item opens the **Script Parameters Definition** dialog (see figure below) followed by the **Rhino JavaScript Debugger** dialog (see below). These dialogs allow invocation of the script with specified parameters for debugging purposes.

**Note:** The **Debug Script** pop-up menu command is only enabled when there is no validation errors in metadata. If the command is disabled (that is, greyed-out), fix all validation errors to enable the menu command.

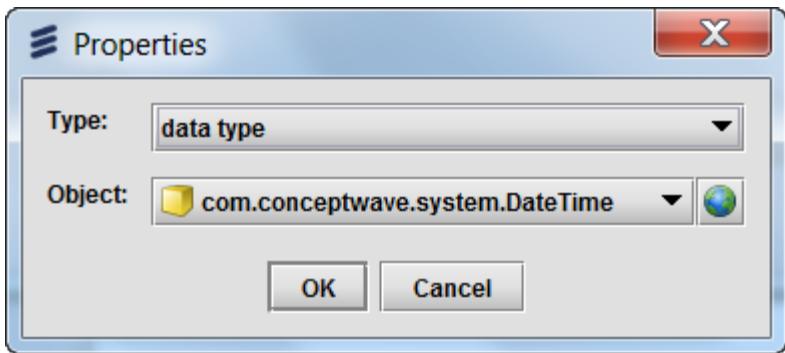


The **Script Parameters Definition** dialog contains a table with a list of the parameters defined in the script. Each parameter must have the type defined and be initialized before the **Rhino JavaScriptDebugger** dialog can be used. Velocity Studio will keep the initialization data for the last debugged script. Therefore the script can be modified and debugged multiple times without having to re-define the parameters with the same parameter name. The table below contains a description of the columns.

Description of columns in the Script Parameters Definition dialog.

Column	Description
Name	The parameter name as defined in the <b>Script</b> field on the <b>General</b> tab. Read-only.
Type	Mandatory. The type of parameter. Either double-click the table row or click the <b>Properties</b> button to open the <b>Properties</b> dialog (see figure below) where the type can be selected. Must be defined to evaluate and debug the script.
Object	Mandatory. The metadata object that represents the parameter type. Either double-click the table row or click the <b>Properties</b> button to open the <b>Properties</b> dialog (see figure below) where the type can be selected. Must be defined to evaluate and debug the script.
Initialized	If checked, the parameter has been initialized.

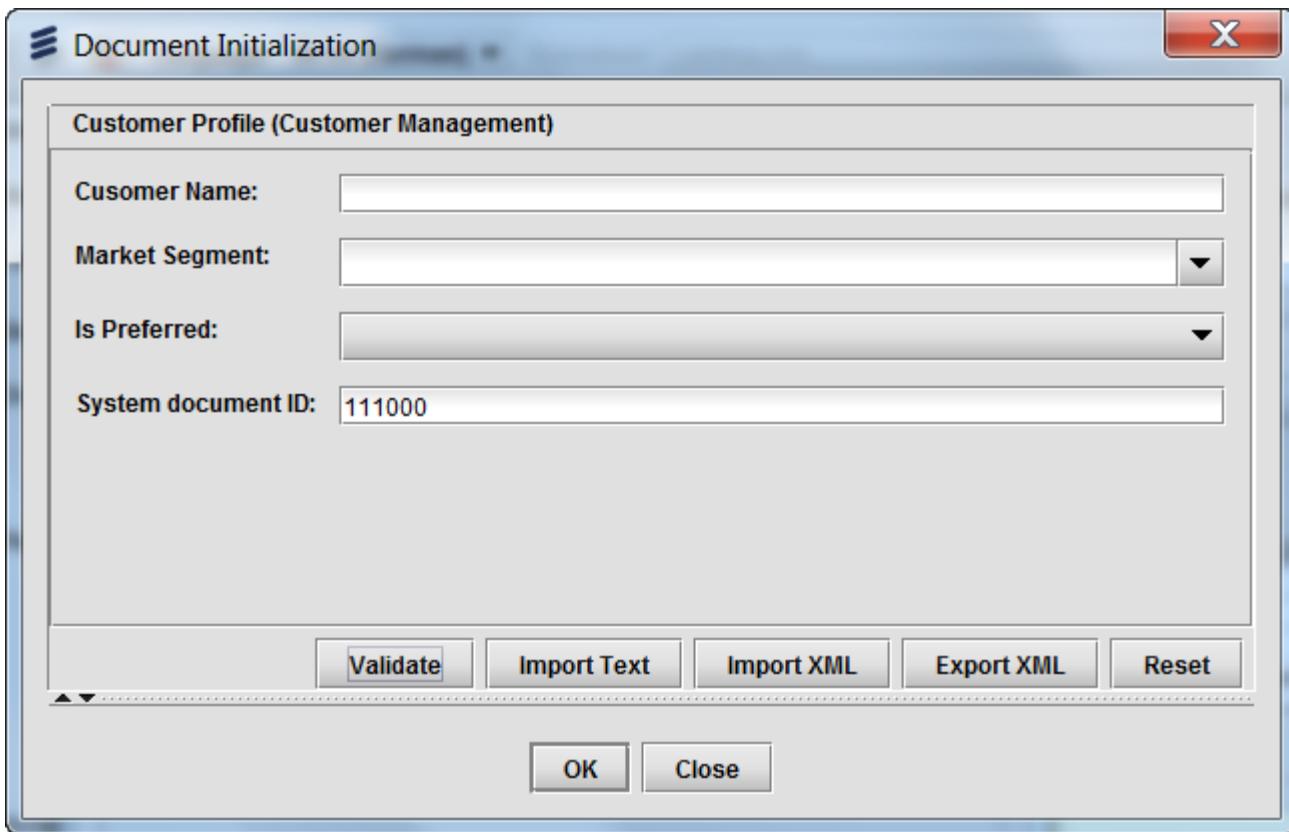
Click the **Properties** button to open the **Properties** dialog where the **Type**, type of parameter, and **Object**, metadata object that represents the type, are defined. The *null* and *process* types do not require an object to be specified.



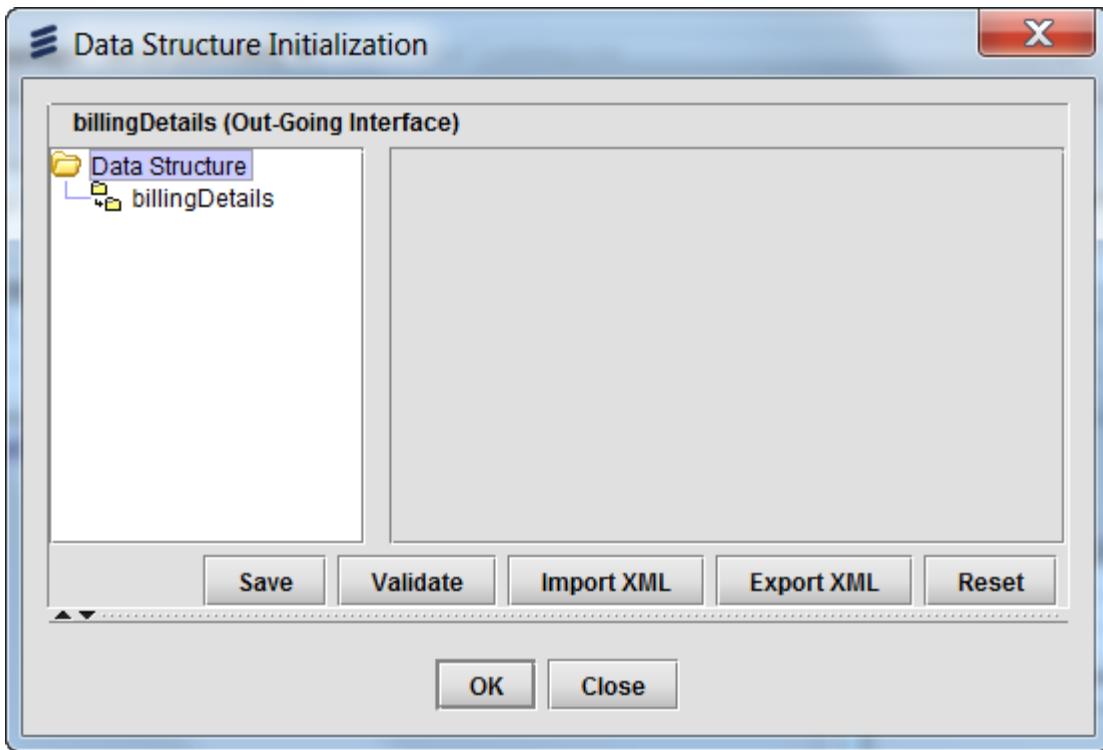
After the **Type** and **Object** have been defined, the parameter value can be set using either the **Initialize** or **From DB** buttons. If the button is not applicable for the chosen object type, the button will be disabled. The *null* type does not require initialization.

When the **Initialize** button is clicked, the dialog that opens depends on the parameter type specified in the **Type** column. This button can be used to initialize the *data type*, *document* and *data structure* types.

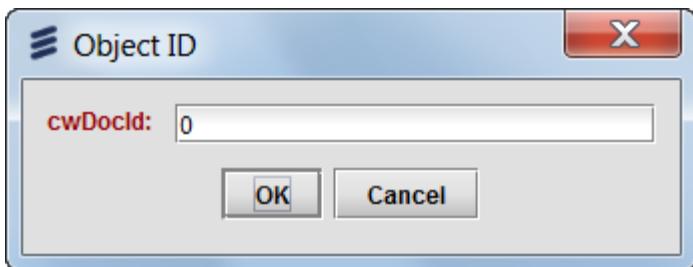
The **Document Initialization** dialog (see figure below) will be opened for the *document* object type. The Document can be initialized by entering data manually or by loading existing data from a text or XML file. To validate the data entered, click the **Validate** button. Click the **OK** button to initialize the parameter.



The **Data Structure Initialization** dialog (see figure below) will be opened for the *data structure* object type. The Data Structure can be initialized by entering data manually or by loading existing data from an XML file (refer to section on Data Structure Operations). To validate the data entered, click the **Validate** button. Click the **OK** button to initialize the parameter.



When the **From DB** button is clicked the **Object ID** dialog (see figure below) will open, prompting the user for the ID of the instance of the specified object. The ID must exist in the database. This button can be used to initialize the *document*, *order* and *process* types.



When all the parameters are initialized, click the **OK** button to proceed to debug the script. The [Rhino JavaScript Debugger](#) dialog will open in front of the Velocity Studio window. The **Rhino JavaScript Debugger** is a third-party product; documentation on how to use this product can be found at <http://www.mozilla.org/rhino/debugger.html>.

To cancel debugging the script, click the **Cancel** button in the **Script Parameters Definition** dialog. The **Rhino JavaScript Debugger** dialog will still open and can be closed at anytime. Velocity Studio will keep the initialization data for the last debugged script. Therefore the script can be modified and debugged multiple times without having to re-define the parameters with the same parameter name.

## Support for MS Excel Spreadsheet Format

Velocity Studio allows you to natively read and write MS Excel 97-2003 and 2010 spreadsheet formats. Support exists for .xls and .xlsx (XML-based) files. There is also support for APIs with read and write cell access.

**Note:** This Excel support is not intended to create on-the-fly spreadsheets with complex formatting. Excel spreadsheets with proper layout and styling must be pre-created to allow for cell values to be manipulated.

### Object Model

The following script objects are available:

- ExcelWorkbook
- ExcelSheet
- ExcelRow
- ExcelCell

These script objects are direct wrappers on top of the Java APIs. Additionally, Global contains an API to load from an attachment or resource, and an API to save it as an attachment.

### CwfScriptGlobalFunctions API

```
CwfScriptGlobalFunctions {
    public ExcelWorkbook loadExcelDocumentFromResource(String path, boolean xlsx, CwfContext uc);
    public ExcelWorkbook loadExcelDocumentFromAttachment(String id, boolean xlsx, AvmContext uc);
    public void saveExcelDocumentToAttachment(ExcelWorkbook book, String id, AvmContext uc);
}
```

### ExcelWorkbook API

```
ExcelWorkbook {
    public ExcelSheet[] getSheets();
}
```

### ExcelSheet API

```
ExcelSheet {
    public ExcelRow[] getRows();
}
```

### ExcelRow API

```
ExcelRow {
    public ExcelCell[] getCells();
}
```

### ExcelCell API

```
ExcelCell {
    public String getValue();
    public void setValue(String value);
    public void setDateValue(Date value);
    public void setBooleanValue(Boolean value);
    public void setDoubleValue(double value);
    public void setFormula(String formula);
}
```

The following is a sample script that uses all APIs:

```

//read cell from xlsx document
var wb = Global.loadExcelDocumentFromResource("Book1.xlsx", true);
var sheet = wb.getSheets()[1];
var row = sheet.getRows()[0];
var cell = row.getCells()[0];
Global.logDebug(cell.getValue());

//read cell from xls document
var wb = Global.loadExcelDocumentFromResource("Book1.xls", false);
var sheet = wb.getSheets()[1];
var row = sheet.getRows()[0];
var cell = row.getCells()[0];
Global.logDebug(cell.getValue());

//change value, save as doc attachment
var docId = "1";
cell.setValue("YYY");
Global.saveExcelDocumentToAttachment(wb, docId);

//read cell from xls document (
var wb = Global.loadExcelDocumentFromAttachment(docId, false);
var sheet = wb.getSheets()[1];
var row = sheet.getRows()[0];
var cell = row.getCells()[0];
Global.logDebug(cell.getValue());

//create a dynamic document from all name-value pairs in bulk upload spreadsheet
var dyn = new DynamicDocument("ns:test", null, null, false);
wb = Global.loadExcelDocumentFromResource("BulkUpload.xlsx", true);
var sheet = wb.getSheets()[0];
var rowNames = sheet.getRows()[0].getCells();
var rowValues = sheet.getRows()[1].getCells();
for (var i=0; i<rowNames.length; i++) {
    var name = rowNames[i].getValue();

    //Note: Names are duplicate in this doc, so do not create leaf again if it exists
    if (!dyn.isDynamicLeaf(name))
        dyn.createLeaf(name, name, 3, 0, 4, 0, null);
    dyn.putValueByName(name, rowValues[i].getValue(), false);
}
dyn.save();

```

## Events

The event is a user-defined string, which is unique for the application. A lightweight, [publish-subscribe](#) mechanism is available that allows you to publish an event to which many different global scripts may subscribe. The event metadata object serves as the signature definition for any event handler metadata object that will handle such an event. To create an event, do the following:

In the **Metadata** tab of **Navigation** pane, either:

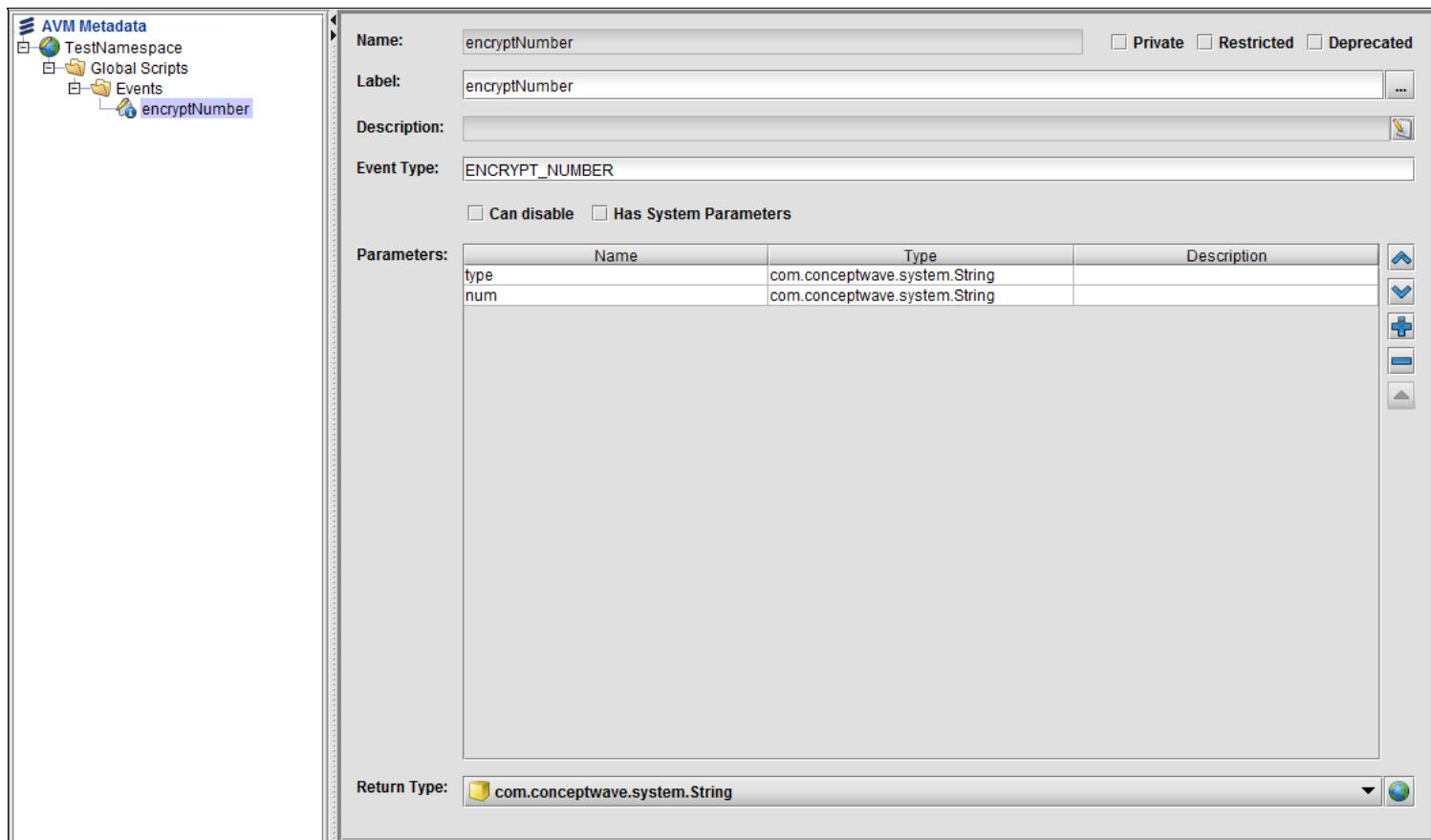
1. Right-click a Namespace. Select **New ...**
2. **New Metadata Object** wizard appears. Expand the **Global Scripts** node, select **Event**, and then click the **Next** button.
3. Enter the name of event in the **Name** field (must conform to JavaScript naming conventions), which is the default name for event handler. You can also specify other parameters.
4. Click the **Finish** button.

OR

1. Right-click the **Events** folder or the **Global Scripts** folder in a Namespace, if it is present. Select **New Event**.
2. Follow step 2 mentioned previously.

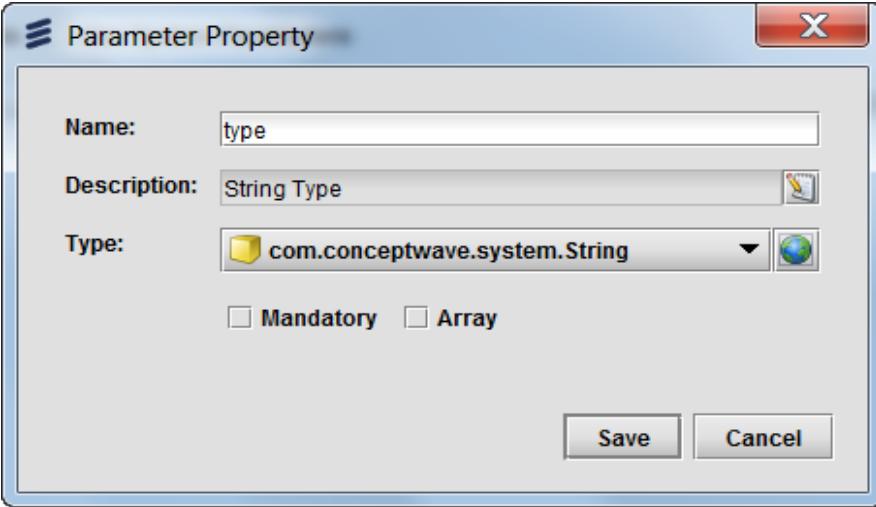
Your newly created event appears under the **Events** folder.

**Note:** Creating two events that have the same **Event Type** is not permitted.



An event contains the following property fields:

Field	Description
<b>Name</b>	The name of the event metadata object (must conform to JavaScript naming conventions). The field is read-only after creation. To change it, use the <b>Rename</b> command in pop-up menu by right-clicking the metadata object.
<b>Label</b>	This field represents the display label of your metadata object display label.
<b>Description</b>	A description of the event metadata object for documentation.
<b>Event Type</b>	Specify a unique name that the application will publish.
<b>Can</b>	Select this property's checkbox to specify that the event can be disabled. The default value is false for this property.

<b>disable</b>													
<b>Has System Parameters</b>	Select this property's checkbox to indicate that the event has system parameters. The default value is false for this property.												
<b>Parameters</b>	Table to define a list of parameters that event handlers are expected to receive. Click the <b>Add</b> or <b>Remove</b> button to add or remove parameters, respectively. The order of the parameters shown in this table is the order of parameters upon script invocation. Click the <b>Edit</b> button with a selected parameter to edit a parameter.												
	 <p>The dialog box is titled "Parameter Property". It contains the following fields:</p> <ul style="list-style-type: none"> <li><b>Name:</b> type</li> <li><b>Description:</b> String Type</li> <li><b>Type:</b> com.conceptwave.system.String</li> <li><b>Mandatory:</b> <input checked="" type="checkbox"/></li> <li><b>Array:</b> <input type="checkbox"/></li> </ul> <p>Buttons at the bottom include "Save" and "Cancel".</p>												
	<table border="1"> <thead> <tr> <th>Field</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><b>Name</b></td> <td>Mandatory. Name of script parameter.</td> </tr> <tr> <td><b>Description</b></td> <td>Click the <b>Pencil</b> icon and describe your parameter in the field. Click the <b>OK</b> button to save your changes.</td> </tr> <tr> <td><b>Type</b></td> <td>Data Type of the parameter.</td> </tr> <tr> <td><b>Mandatory</b></td> <td>If checked, the parameter is mandatory and the expression editor will not allow the parameter value to be null.</td> </tr> <tr> <td><b>Array</b></td> <td>If checked, the parameter is an array of the specified Data Type.</td> </tr> </tbody> </table>	Field	Description	<b>Name</b>	Mandatory. Name of script parameter.	<b>Description</b>	Click the <b>Pencil</b> icon and describe your parameter in the field. Click the <b>OK</b> button to save your changes.	<b>Type</b>	Data Type of the parameter.	<b>Mandatory</b>	If checked, the parameter is mandatory and the expression editor will not allow the parameter value to be null.	<b>Array</b>	If checked, the parameter is an array of the specified Data Type.
Field	Description												
<b>Name</b>	Mandatory. Name of script parameter.												
<b>Description</b>	Click the <b>Pencil</b> icon and describe your parameter in the field. Click the <b>OK</b> button to save your changes.												
<b>Type</b>	Data Type of the parameter.												
<b>Mandatory</b>	If checked, the parameter is mandatory and the expression editor will not allow the parameter value to be null.												
<b>Array</b>	If checked, the parameter is an array of the specified Data Type.												
<b>Return Type</b>	Click the drop-down menu and select the metadata object type for which event handlers are expected to return. Select <code>com.conceptwave.system.Void</code> if no value shall be returned.												

## Event Handlers

Velocity Studio allows defining new types of metadata elements called **event handlers**, which are global scripts that automatically subscribe for the [event](#) that they specify and do not require calling the `subscribeForEvent` API. The event handler metadata object captures the logic that is required to perform the expected functionality set out by the event.

A lightweight, [publish-subscribe](#) mechanism is available that allows you to publish an event to which many different global scripts may subscribe. The event metadata object serves as the signature definition for any event handler metadata object that will handle such an event. To create an event handler, do the following:

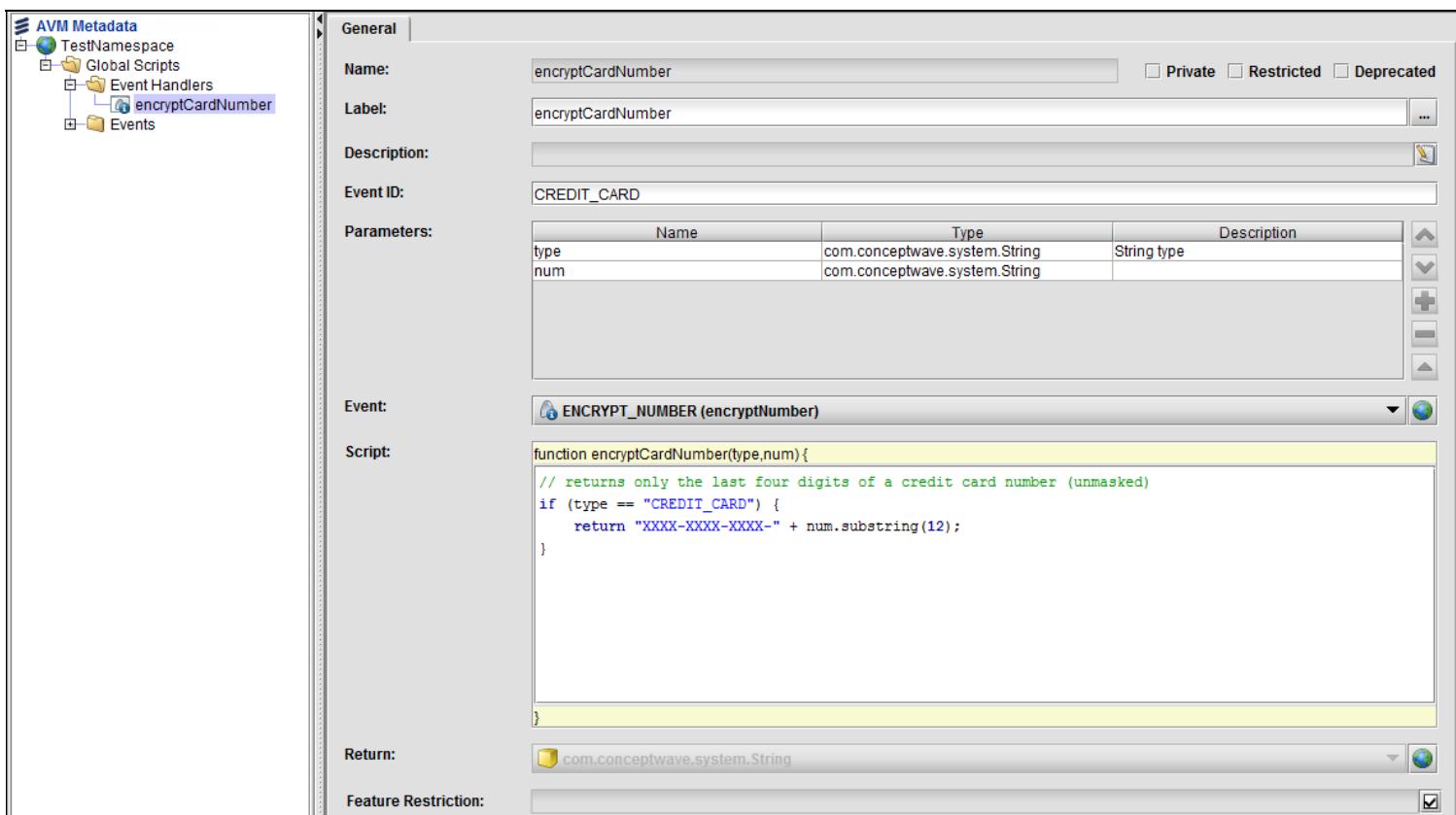
In the **Metadata** tab of **Navigation** pane, either:

1. Right-click a Namespace. Select **New ...**
2. **New Metadata Object** wizard appears. Expand the **Global Scripts** node, select **Event Handler**, and then click the **Next** button.
3. Enter the name of event handler in the **Name** field (must conform to JavaScript naming conventions), which is the default name for event handler. You can also specify other parameters.
4. Click the **Finish** button.

OR

1. Right-click the **Event Handlers** folder or the **Global Scripts** folder in a Namespace, if it is present. Select **New Event Handler**.
2. Follow step 2 mentioned previously.

Your newly created event handler appears under the **Event Handlers** folder.



An event handler contains the following property fields:

Field	Description
<b>Name</b>	The name of the event metadata object (must conform to JavaScript naming conventions). The field is read-only after creation. To change it, use the <b>Rename</b> command in pop-up menu by right-clicking the metadata object.
<b>Label</b>	This field represents the display label of your metadata object display label.
<b>Description</b>	A description of the event metadata object for documentation.
<b>Event ID</b>	The unique identifier that denotes the event handler.
<b>Parameters</b>	Table to define a list of parameters that event handlers are expected to receive.

<b>Event</b>	The event type for which this event handler handles.
<b>Script</b>	The logic that performs the expected functionality.
<b>Return</b>	The metadata object type for which event handlers are expected to return.
<b>Feature Restriction</b>	This field allows you to specify <a href="#">features that you want to restrict</a> . Click the field's <b>Checkbox</b> button ( <input checked="" type="checkbox"/> ) to launch the Select Feature dialog. You can select all features that you want to restrict and then click the <b>Save</b> button.

From the application, an event is published as follows:

```
publishEvent(event, mode, thisObject, parameters, ...)
```

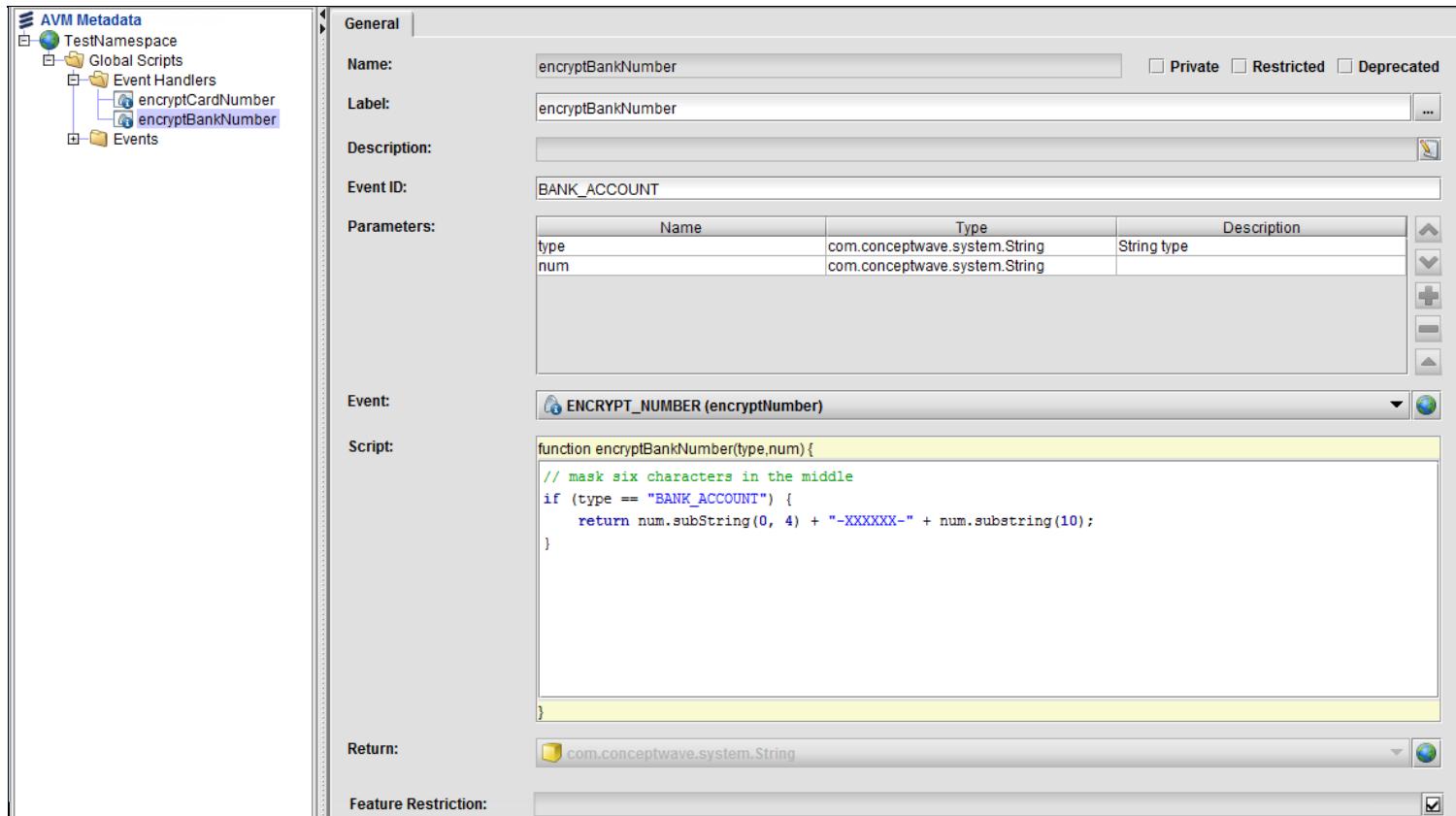
Where:

- event corresponds to the event type as specified in the event. Event handlers that handle this event type are processed in some sequence by the product.
- mode allows you to specify one of the following modes, with its expected behaviour:
  - **FIRST\_ONE** - processes each event handler until the first event handler returns a non-null value.
  - **SYNCH** - processes each event handler synchronously.
  - **ASYNCH** - processes each event handler asynchronously.
  - **Event ID** - processes a specific event handler for given Event ID synchronously.
  - **PROPAGATE** - passes the result of the previous event handler as the first parameter for the next event handler.  
Because event handlers do not run in the same order, this mode accumulates data without relying on the order of the data. You must define at least one parameter in the event script. If publishEvent() API passes a value for the *parameters* parameter, that value is used for the first event handler. Then the subsequent event handlers receive the result of the previous event handler as their first parameter.  
See this [example](#) for more information.
- thisObject is an object that can be accessed with **this** in the event handler.
- parameters is a sequential list of parameters corresponding to the list defined in the event.

For example, to publish the event as defined previously from the application, use the following:

```
var cardNumber = "4520124122532652";
publishEvent("ENCRYPT_NUMBER", FIRST_ONE, null, "CREDIT_CARD", cardNumber)
```

With the event and event handler mechanism, applications can have two event handlers for one event, as shown in this example:



In this example, the output from the ENCRYPT\_NUMBER event varies, depending on the parameter type.

## Get Information about Subscribed Event Handlers

The following global script functions are available for getting information about subscribed event handlers:

- getEventHandlers
- getEventIds
- hasEvent
- hasEventId

See the JavaScript documentation for details.

To invoke methods, you do not need to use any prefix, as shown in the following example:

```
var events = getEventIds(eventName);
```

## PROPAGATE Mode Example

This example illustrate how the PROPAGATE mode of the publishEvent() API works. If you have two event handlers as follows:

First event handler:

```
function EH1(data) {
  if(data == null)
    data = "";
  return data + " World";
}
```

Second event handler:

```
function EH2(data) {
  if(data == null)
    data = "";
  return data + " Hello";
}
```

And the actual script is as follows:

```
var result = publishEvent("SOME_EVENT", PROPAGATE, null, {"Example"});
```

The result is either *Example Hello World* or *Example World Hello*, depending on which event handler, EH1 or EH2, is invoked first.

## Publishing and Subscribing to Events

The AVM implements a lightweight, publish-subscribe mechanism that allows you to publish an event to which many different global scripts may subscribe. The event is a user-defined string, which is unique for the application.

The following API controls the publish-subscribe script mechanism:

```
subscribeForEvent(script, event, id)
```

Where:

- `script` is a fully qualified metadata global script element name
- `event` is a user-defined, application-wide, unique name
- `id` is an optional name, which is unique for the given event

**Note:** The AVM allows you to specify full metadata element names using a period (.) as a delimiter between the namespace and element name (for example, `ns:name` is equivalent to `ns.name`). It is recommended that you use the period notation instead of the semicolon notation.

The API registers a global script as a subscriber for the specified event. If the `id` parameter is not specified or null, the script is added as a subscriber for the event, unless it is already subscribed for the event. In the latter case, the subscribe operation is silently ignored. If the `id` parameter is not null, the script is added as a subscriber for the event (unless it is already a subscriber) and replaces the subscriber with the same `id`, if it exists. The subscriber, which is defined later, can override the subscriber that was defined earlier.

Velocity Studio allows defining new types of metadata elements called **event handlers**, which are global scripts that automatically subscribe for the event that they specify and do not require calling the `subscribeForEvent` API. Velocity Studio comes with a predefined list of AVM events and all modules define their own event lists. These events can be used in the application metadata. The application can also add new events and use them as needed.

**Note:** The publish-subscribe script mechanism does not need event handlers to work. However, using them simplifies creating and maintaining the applications.

```
publishEvent(event, mode, this object, parameter, ...)
```

Where

- `mode` is either the ID of a subscriber script or one of the following predefined constants:
  - `SYNCH`
  - `ASYNCH`
  - `FIRST_ONE`
- `this` `parameter` is an object that is used as `this` object in the script and can be null
- `parameter` specifies zero, one, or more parameters. The same parameters are used for every subscriber script for the event.

The `publishEvent` API processes the subscribers for the event, depending on the mode, as follows:

- **Synchronous mode (SYNCH)**

All subscribed scripts are processed synchronously one after another in the order that they subscribe to the event. The API does not return any value. The values returned by the individual scripts are ignored. If one script generates an exception, the remaining scripts are not processed. The individual scripts may change some environment values, but should be done with care as the processing order of these scripts is unknown.

- **Asynchronous mode (ASYNCH)**

All subscribed scripts are processed in parallel in separate threads. The order in which the threads start is the order that they have subscribed for the event. The order in which they finish is random. The API does not return any value. The values returned by the individual scripts are ignored. If a particular script generates an exception, its processing is terminated; however, the rest of the scripts continue to run. The individual scripts should not change the environment values.

- **First one mode (FIRST\_ONE)**

The subscribed scripts are processed synchronously one after another until one of them returns a not null value or all of them are processed. The API returns the not null value returned from the script or null if all scripts return null. If a particular script

generates an exception, the remaining scripts are not processed. The individual scripts may change the parameters values; however, in this mode, it likely will not

- **Subscriber ID**

The script with a given ID, if any, is performed in synchronous mode. The script's result is returned and no other subscribed script is run.

If there are no subscribers for a given event, the API completes immediately and returns null. For synchronous and asynchronous modes, the API returns the Boolean value of true if there is at least one script to run.

The subscriber scripts are global metadata scripts that must have the same parameters as the parameters at the end of the publishEvent() API call.

The publish-subscribe mechanism is especially useful when customizing templates. An example is the customer template that must read and store the customer data distributed in one or more external systems. One of these operations is the store operation.

The traditional solution for this problem is to include an empty global script (for example, storeExtCustomer) in the customer template. This empty global script is called at a particular place in the store customer logic of the template, which works with one data store only. If the customer data is distributed in several external stores, the application overrides the storeExtCustomer script, introducing the proper logic to store customer in the additional data stores. While this solution is correct, it has a small drawback concerning the introduction of an empty script that will not be used in the majority of cases. There is also a big limitation, as the application will not be able to include two different templates that override the storeExtCustomer script.

The publish-subscribe mechanism, however, solves the multiple overriding problem well. Each template subscribes to the StoreCustomer event and the customer template logic publishes the event when there is a need to store customer data. The solution is outlined as follows:

- In the customer template logic for storing the customer:

```
publishEvent("StoreCustomer", SYNC, theCustomerObject);
```

- In the corresponding template initialization scripts:

```
subscribeForEvent("tns.tscript", "StoreCustomer");
```

The corresponding tns.tscript method with theCustomerObject as the this object will store its portions of the customer object in the corresponding data stores.

It is recommended that you avoid script-overriding. The publish-subscribe mechanism is available everywhere in the metadata products (templates), which eliminates the need to override scripts. However, overriding data structures will still be needed.

## Define a Constant

The DEFINE\_CONSTANT API allows you to define a constant that can be used only in system or template initialization, and in startup scripts. The API takes the following form:

```
DEFINE_CONSTANT (constant name, constant value, constant name, constant value...)
```

Constant names are strings that are valid JavaScript identifiers. Constant values can be any object.

**Note:** It is strongly recommended that constant names be in uppercase lettering with an underscore as a word separator.

Using the DEFINE\_CONSTANT API after initialization is complete will generate an exception. The same outcome occurs if the DEFINE\_CONSTANT API specifies a constant name that has already been used as a constant, script type name (such as a Document or Order), or any other AVM scope identifier. Once defined, the constant can be used as shown in the following example:

- In the system startup script:

```
DEFINE_CONSTANT ("STR_CONST", "a", "NUMERIC_CONST", 5);
```

- In the application:

```
var res = STR_CONST + NUMERIC_CONSTANT;
```

In this case, `res` has the value `a5`.

Constants can only be defined in AVM init. To create an initialization script, do the following:

1. Create a namespace called `cwlInit`.
2. Create a global script called `cwAVMInit`.

## Rule Sets

Rule Sets are used for data initialization, validation, and permissions. A Rule Set defines a set of rules that can be evaluated against a source data object (Document, Data Structure, or Top- Level User Interfaces) and its extensions. Individual rules are defined per data object node. Depending on type, rules can be defined on the top node or its immediate data type child nodes.

### Rule Sets Types

There are three types of the rule set.

- [Initialization Rule Set](#)
- [Validation Rule Set](#)
- [Permission Rule Set](#)

### Create a Rule Set

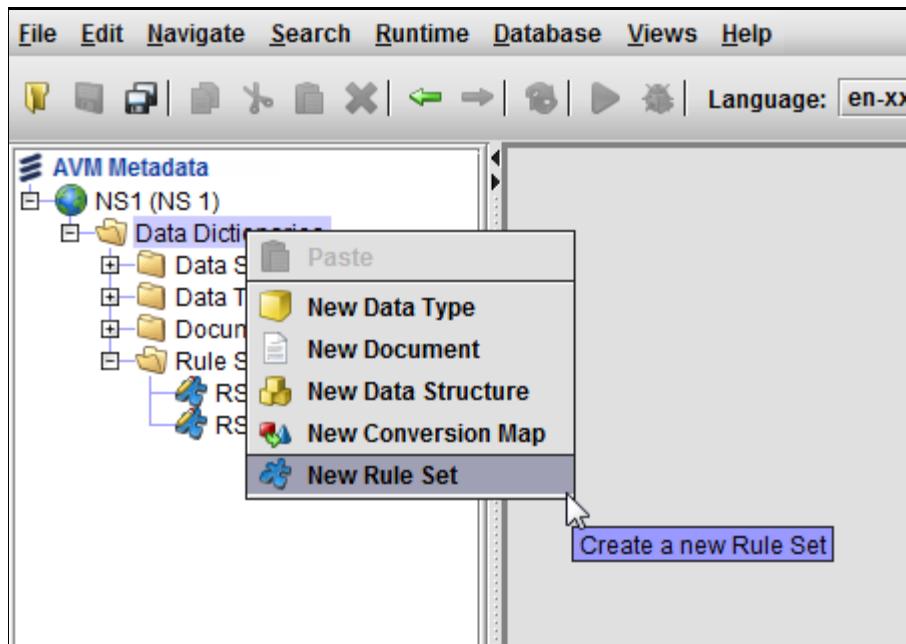
To create a new Rule Set, do one of the following options:

In the **Metadata** tab of **Navigation pane**, either:

1. Right-click a Namespace and select **New**.
2. **New Metadata Object** wizard appears as a popup. Expand **Data Dictionaries**, select **Rule Set**, and then click the **Next** button.
3. Step two of the wizard appears. Enter the **Name** of the Rule Set. Enter the **Label** for the Rule Set. Select **Type** and **Source** for the Rule Set, which are mandatory fields. Then, click the **Finish** button.

OR

1. Right-click the **Data Dictionaries** folder in a Namespace, if it is present. Select **New Rule Set**.
2. Follow step 3 from the previous procedure.



### Order of Application of Rule Sets to a Data Object

When multiple rule sets are applied to an object, applied order of rule sets is not defined or guaranteed to be consistent for those rule sets defined specifically for the source object type. In the case of object extension, rule sets defined for base types are applied before

the rules defined on extended types. However, this is on a variable-by-variable basis. In other words, all application rule sets are applied to the first variable, then the next variable, and so.

## Data Object Traversal

Traversal order of data object nodes and sub-nodes is not guaranteed or defined.

## Definition and Application of Rule Sets to Child Nodes

Rules can not be defined and applied depending on the rule set type and child node type. See the following chart:

Node Type	Initialization		Validation		Permission	
	Can Define Rule	Node's Base Type Rule Set Applied	Can Define Rule	Node's Base Type Rule Set Applied	Can Define Rule	Node's Base Type Rule Set Applied
<b>Top Node</b>	No	Yes	Yes	Yes	Yes	Yes
<b>Data Type Node</b>	Yes	N/A	Yes	N/A	Yes	N/A
<b>Container Node</b>	NO <sup>1</sup>	NO <sup>1,2</sup>	NO <sup>1</sup>	NO <sup>1</sup>	NO <sup>1</sup>	NO <sup>1</sup>
<b>Element Node<sup>3</sup></b>	NO	NO <sup>4</sup> YES <sup>5</sup>	Yes	Yes	NO	YES
<b>Array Node</b>	NO	Same as non-array node type	NO	Same as non-array node type	NO	Same as non-array node type

1. In the container node, definition of rules or application of rule sets is not allowed. It is a good practice to define top-level data object for all sub-structures. It results in improving the extensibility and reuse since anonymous container types cannot be extended, and they can be used meaningfully out of context of the parent.structure.
2. Even if a container node extends a top-level data structure, when container node are created, the framework currently does not run the top-level object initialization routine. It is not a good practice to apply rule sets from the top-level type either.
3. Element nodes are data structure child nodes that are defined with the element field referencing a top-level data object. These top-level objects do *not* have to be marked as *elements*.
4. When the parent node is created, container or element nodeds are not created automatically, so there is no node to apply the rule set to.
5. When referenced (for example, TopNode.ElementNode.Value1 = ...), element nodes are automatically instantiated and their initialization routines are executed.

## Initialization Rule Set

---

An initialization rule defines how to initialize data for specific nodes of a data object. Initialization rules consist of scripts whose return value is applied to a given node. Each node can have only one initialization rule. Initialization rule sets are applied automatically to an object when the object is constructed. During construction, the order of initialization actions is:

1. Data type default values are applied to nodes.
2. Leaf initialization scripts are applied.
3. Rule sets are applied.
4. Data object initialization scripts are applied.

An initialization rule can only be applied to the immediate child data-type nodes of an object. When applied to data type nodes, the return type is expected to match the type of the node, but it is not enforced. Applying an initialization rule to a node is equivalent to running the following in script:

```
dataObject.nodeName = dataObject.ruleScript();
```

where *ruleScript* is a script method with the same content of the rule's script.

During normal object initialization, element nodes are not automatically created. Therefore, rule sets for element nodes do not run automatically. Rule sets for element nodes are triggered when these nodes are manually instantiated (for example with a constructor), or automatically instantiated when referenced. When rule sets are applied to a data structure element node, the result is expected to be a new data structure object.

## Validation Rule Set

---

A validation rule defines conditions. If these conditions are met, rule set triggers validation errors. To validate the contents of element nodes, rule sets are defined on the referenced element.

Each node can have more than one validation rule defined. Validation rules are automatically applied when the **validate** script method of a data object is run or the data object user interface form is validated. If a node has multiple validation rules defined, they are evaluated in the order in which they are defined. The evaluation of rules for a node stops after the first rule is triggered.

### Validation Rule Types

- **Mandatory:** It is triggered, if a value for a node has not been set.
- **Script:** It is triggered, if rule set's associated script returns false.

## Permission Rule Set

---

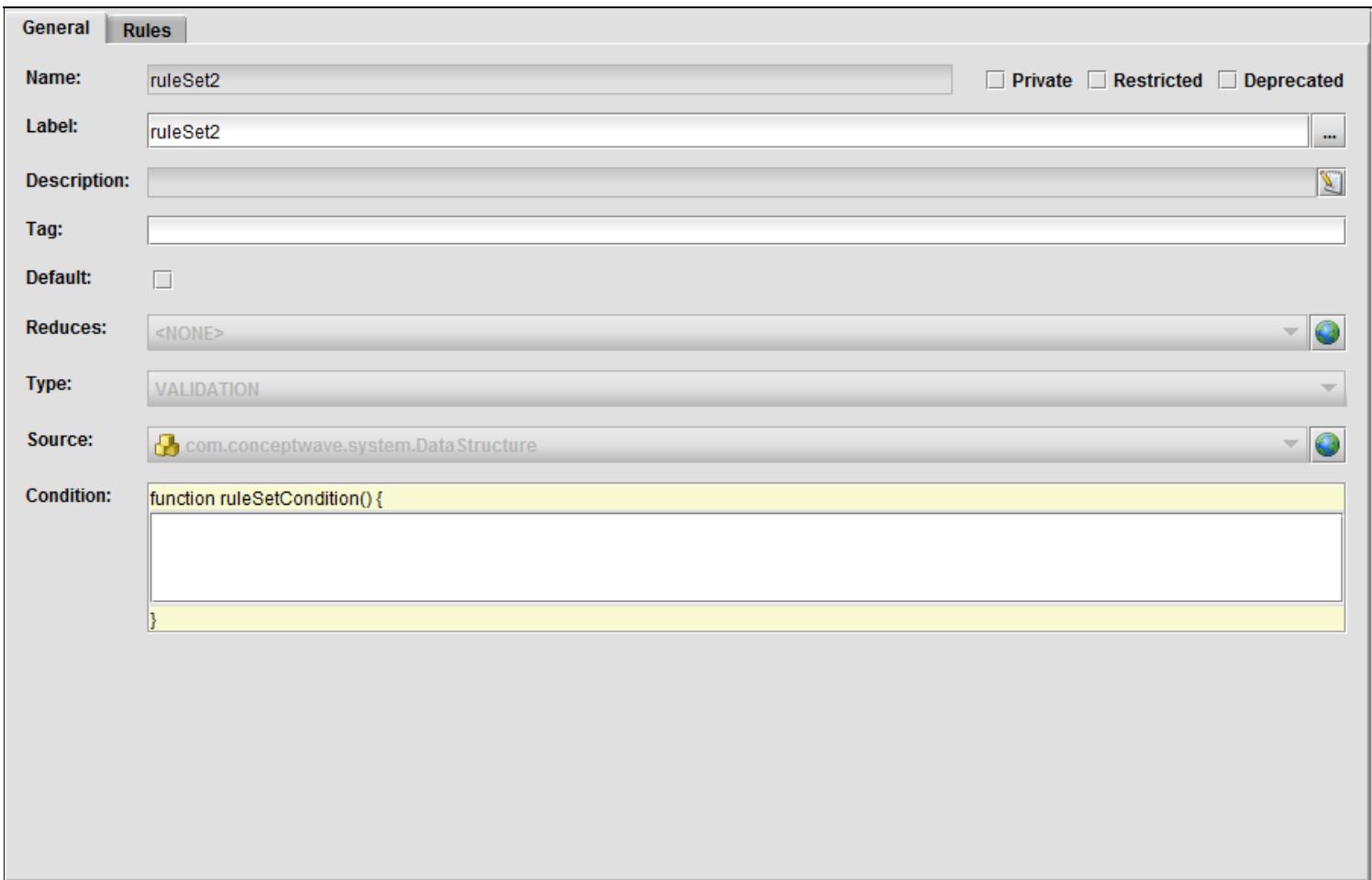
A permission rule defines accessibility of data object nodes in user interface (UI). Permission of element nodes or containers is determined by rule sets defined on the referenced element. Each node in a data object can have exactly one rule defined of each type of application. Permission rules are automatically evaluated by the UI when rendering the data object's UI. The order of permission rules is not relevant.

### Permission Rule Types

Type	Description
<b>Visible</b>	Indicates a given node can be visible in the UI. This rule type is applicable to all nodes.
<b>Editable</b>	Indicates a given node can be edited in the UI. This rule type is applicable to all nodes.
<b>Optional</b>	Indicates a value for a given node does <i>not</i> need to be defined. This rule type is applicable to data type.
<b>Add</b>	Indicates a given node can be added to its parent. This rule type is applicable to container nodes only.
<b>Delete</b>	Indicates a given node can be removed from its parent. This rule type is applicable to container nodes only.

## Rule Sets General Properties

The general rule set properties are on the **General** tab.



The screenshot shows the 'Rules' tab of the Rule Sets General Properties dialog. The 'Name' field contains 'ruleSet2'. The 'Label' field also contains 'ruleSet2'. The 'Description' field is empty. The 'Tag' field is empty. The 'Default' checkbox is unchecked. The 'Reduces' dropdown is set to '<NONE>'. The 'Type' dropdown is set to 'VALIDATION'. The 'Source' dropdown is set to 'com.conceptwave.system.DataStructure'. The 'Condition' field contains the following JavaScript code:

```
function ruleSetCondition() {  
}  
}
```

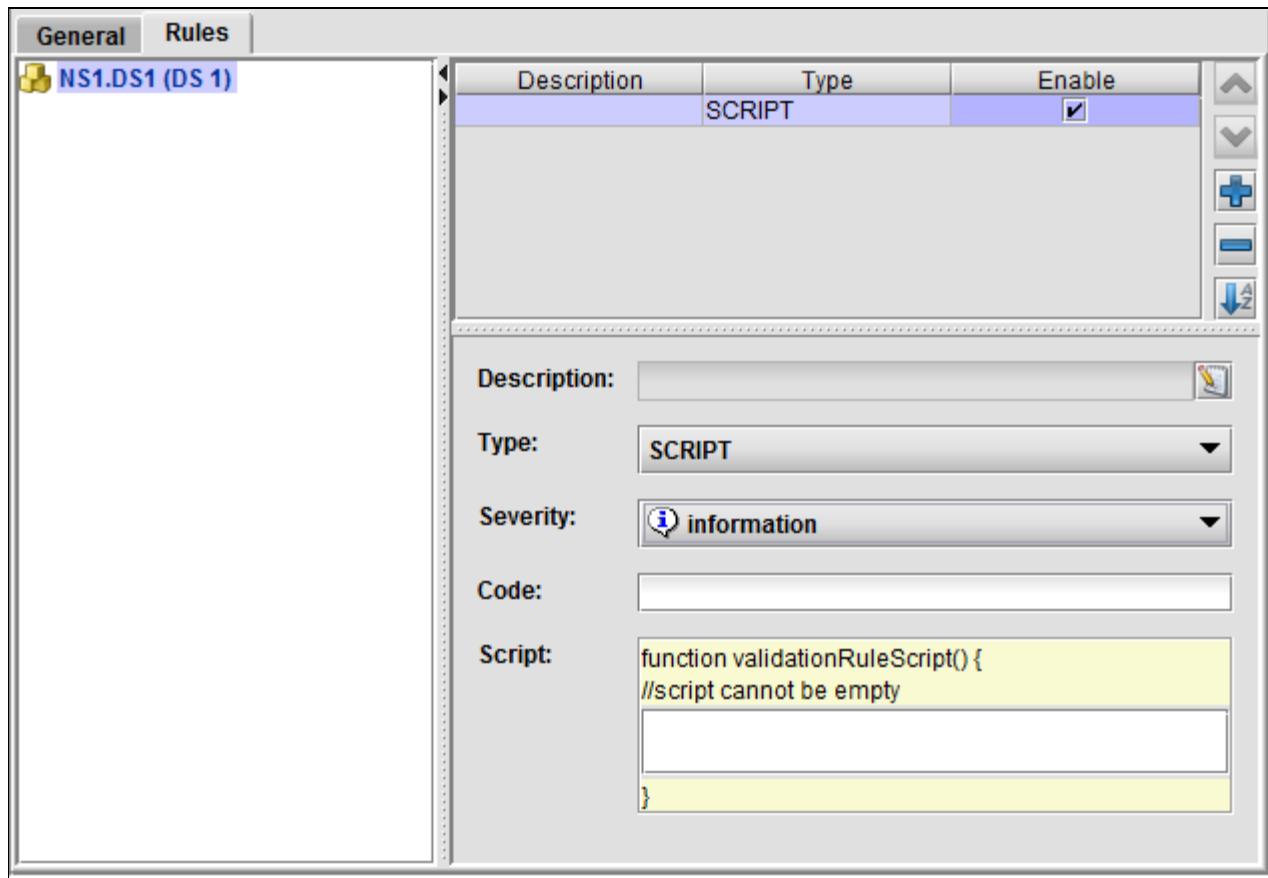
The following table describes the fields:

Field	Description
<b>Name</b>	Name of the metadata object (used in scripts; must conform to JavaScript naming conventions). Read-only after creation. To change it, use the <b>Rename</b> command in the popup menu by right-clicking the metadata object.
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as product metadata (that is, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as product metadata (that is, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Label</b>	Metadata object display label.
<b>Description</b>	Description of the metadata object for documentation.
<b>Tag</b>	This field is optional and takes a string value. Either This field must be set or the rule set must have its <b>Default</b> field's checkbox selected. Select this field indicates that this rule set can only be applied to data objects that have a matching tag set. Tags can be set on data objects with either the <code>setRuleSetTag()</code> script method or when the object is instantiated. See the DataObject API documentation for more details.
<b>Default</b>	This property must be set or a tag must be defined in the <b>Tag</b> field. Selecting this field and the condition is evaluated as true, this rule set is used if the metadata does not have a tag associated with it. Otherwise, leaving this field unselected means that this rule set can only be applied to data objects with matching tags.
<b>Reduces</b>	If selected, the rules defined in a base Rule Set can be deleted, replaced or reused.

Type	Select one out of Initialization, Validation, or Permission. This field is mandatory.
Source	Source is a metadata object of the type Document, Data Structure or Top-Level UI. If source rule set extends another, then it defaults to the base Rule Set's source.
Condition	By default, the condition returns true. The condition is evaluated in the context of the data object, which allows the script to access the data object instance with <b>this</b> .

## Rule Sets Rules Properties

The rule properties of rule sets are on the **Rules** tab.



The following table describes the fields:

Field	Description
<b>Description</b>	Description of the Rule Set for documentation.
<b>Enabled</b>	This field is mandatory and its default value is true. Editable only if the Rule Set reduces another and this Rule is defined in the base.
<b>Type</b>	The value of this field depends on Rule Set Type. <ul style="list-style-type: none"><li>• For Initialization Rules, this field only takes <b>Script</b>.</li><li>• For Validation Rules, this field can be either <b>Script</b> or <b>Mandatory</b>. If <b>Mandatory</b>, the <b>Script</b> property cannot be defined.</li><li>• For Permission Rules, type can be one of the followings depending on node type:<ul style="list-style-type: none"><li>◦ Visible</li><li>◦ Editable</li><li>◦ Optional</li><li>◦ Add</li><li>◦ Delete</li></ul></li></ul>
<b>Severity</b>	This field is Mandatory and Editable for Validation Rules only. select from one of the following values: <ul style="list-style-type: none"><li>• Information</li><li>• Warning</li><li>• Error</li></ul>
<b>Code</b>	This field is Mandatory and Editable for Validation Rules only.

<b>Script</b>	This field is Mandatory and Editable if Rule Type is Script. The script is evaluated in the context of the data object, allowing the script to access the data object instance with <b>this</b> . <ul style="list-style-type: none"> <li>• For initialization rules, the return value from this script is assigned to the Rule's node.</li> <li>• For validation rules, if this script returns true, the Rule is <i>Triggered</i>.</li> <li>• For permission rules, if this script returns true, it indicates that the user has the given permission type on this node.</li> </ul>
---------------	--

The following table describes the action icons:

	Use these buttons to add or remove the items from the list.
	Use these buttons to move up or move down the items in the list.
	Use this button to sort the items in alphabetical order.

# Rules

Rules are created for implementing different types of application business rules. When defining a Rule, determine the following:

- Rule Document.
- Input parameters.
- Rule Instances, each containing a particular rule that will be evaluated at runtime.

## Types of Rules

---

The following types of Rules are supported:

- *Initializer Rules* that assign values to a Document.
- *Finder Rules* that select a subset of Document records from the database.
- *Action Rules* that run some actions, usually coded in a global script.

## Create a Rule

To create a rule, complete these steps:

1. Right-click the **Rules** folder in the **Business Processes** folder in the **Navigation** pane of Velocity Studio.
2. Select the **New Rule** command. Enter a Name and Label for the rule and then select the Type (that is, initializer, action, or finder).
3. Click the **Finish** button to create the Rule.
4. The **General** and **Config Instances** tabs will appear where the Rule properties are defined. The **Config Instances** tab is enabled when a database connection is available. If the Rule has the **Type** of *finder* in the **General** tab, a **Result Map** tab will also appear. This tab will only be enabled if a Rule document and a Finder result Document is defined in the **General** tab.

The screenshot shows the 'General' tab of the Velocity Studio configuration interface. The tab bar includes 'General', 'Config Instances', and 'Result Map'. The 'General' tab is active. The form contains the following fields:

Name:	TestRule	<input type="checkbox"/> Private	<input type="checkbox"/> Restricted	<input type="checkbox"/> Deprecated	<input type="checkbox"/> Final
Label:	TestRule	...			
Description:	<input style="width: 150px; height: 20px;" type="button" value="..."/>				
Help:	<input style="width: 150px; height: 20px;" type="button" value="..."/>				
Overrides:	<NONE>	<input style="width: 150px; height: 20px;" type="button" value="..."/>			
Type:	finder	<input style="width: 150px; height: 20px;" type="button" value="..."/>			
<input type="checkbox"/> Not exclusive					
Rule document:	com.conceptwave.system.Document	<input style="width: 150px; height: 20px;" type="button" value="..."/>			
Finder result:	<NONE>	<input style="width: 150px; height: 20px;" type="button" value="..."/>			
Input 1:	<input type="radio"/> Optional	<NONE>	<input style="width: 150px; height: 20px;" type="button" value="..."/>		
Input 2:	<input type="radio"/> Optional	<NONE>	<input style="width: 150px; height: 20px;" type="button" value="..."/>		
Input 3:	<input type="radio"/> Optional	<NONE>	<input style="width: 150px; height: 20px;" type="button" value="..."/>		

Once the Rule is created, the icon will be added under the **Rules** folder.



## Rules General Tab

The general Rule properties are defined on the **General** tab. The table that follows describes the common fields on the **General** tab.

The following image shows the General tab of the Rule properties:

The screenshot shows the 'General' tab selected in a tab bar. Below are several configuration fields:

- Name:** TestRule
- Label:** TestRule
- Description:** (empty)
- Help:** (empty)
- Overrides:** <NONE>
- Type:** finder
- Not exclusive:**
- Rule document:** com.conceptwave.system.Document
- Finder result:** <NONE>
- Input 1:**  Optional <NONE>
- Input 2:**  Optional <NONE>
- Input 3:**  Optional <NONE>

Field	Mandatory/Optional	Comment
<b>Name</b>	Mandatory	Unique name of the Rule within the namespace.
<b>Private</b>	Optional	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	Optional	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	Optional	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Final</b>	Optional	Metadata elements can be marked as final, which includes any Boolean final property that is available on any metadata element. Additionally, any metadata element that has the <b>Final</b> property set to true can neither be extended, nor overridden.
<b>Label</b>	Mandatory	Label to identify the Rule.
<b>Description</b>	Optional	Description of the Rule for documentation purposes.
<b>Help</b>	Optional	This is the popup <a href="#">Help</a> to be displayed at runtime.
<b>Overrides</b>	Optional	Specifies an overridden element Only Rules from libraries that have instances can be overridden. The instances of the overridden Rule will not be executed at runtime.
<b>Type</b>	Mandatory	Type of Rule. Can be either Initialization, Finder or Action. After a Rule is created, the Rule type cannot be changed and this field will be disabled.
<b>Not exclusive</b>	Optional	When checked, all Rule instances are processed. When unchecked (default), the Rule instances are processed executed until the first instance with a true condition is reached.
<b>Rule</b>	Mandatory	Mandatory for Initialization and Finder Rules. Specifies a Rule Document.

<b>document</b>		
<b>Finder result</b>	Optional	Click the drop-down menu and specify the output of your finder result (for example, a tree document).
<b>Input1, Input2, Input3</b>	Optional	Input parameters.

If a Rule has the **Type** of *finder* there will be additional fields on the **General** tab (see figure below) including a **Finder result** field and **Optional** radio buttons.

The **Finder result** field allows specifying a Document in the result list of the Rule. This field remains disabled until the Rule has been created by clicking the **Create** button. If the **Finder result** is not specified, the **Rule document** will be used as the Finder Rule result Document. If the **Finder result** is specified, the Finder Rule result Document must be mapped to the rule parameters (**Rule document** and/or input Documents) and/or to the global variables defined for the Rule (refer to section on Rule Instances).

Each input Document parameter (**Input 1**, **Input 2** and **Input 3**) has an additional radio button, **Optional**, that allows marking one of the parameters as optional. An optional Document can be used to narrow down the search result (that is, it can specify additional conditions that will be skipped if the parameter is not specified). An expression in the Rule Instance that uses an *optional* Document is called a *filter* and is used as search criteria.

## Rule Instances

Rule Instances are created for evaluating particular rules of a given type of business rule. Rule Instances can be created in these ways:

- In the Velocity Studio application **Metadata** tab: on the **Config Instances** tab of the Rule properties.
- In the Velocity Studio application **Library** tab: on the **Config Instances** tab of the Rule properties.

### Notes:

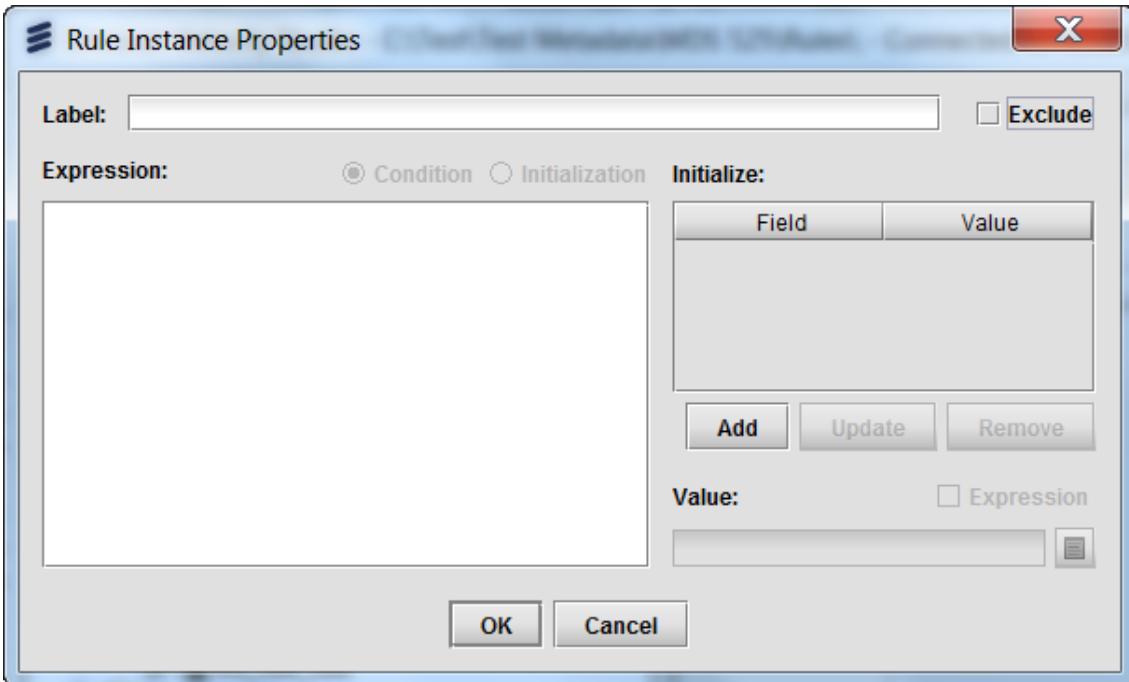
- Instances of overridden rules cannot be changed. Overridden rules will have the **Save** button disabled as any other library object.
- When specifying a rule instance value for an object that has reference finder values, you must start runtime before any reference finder values can be used in setting up rules.

## Commands for Rule Instances

You can perform the following commands for rule instances:

- [Export rule instances](#)
- [Import rule instances](#)
- [Migrate rule instances from command line](#)

Click the Add button on the ConfigInstance tab to open the **Rule Instance Properties** dialog that allows you to view the properties of a selected rule instance. The appearance of this dialog depends on the rule type (can be either an instance of an Initializer Rule, Finder Rule, or an Action Rule). You can double-click a rule instance to view its properties.

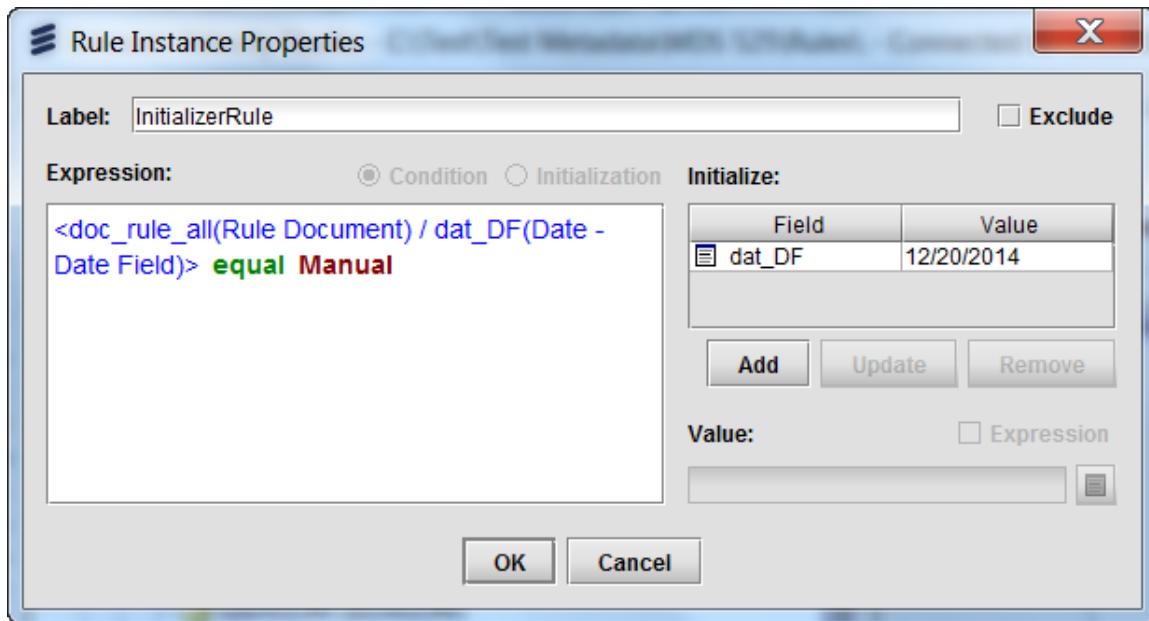


The major element in this dialog is an expression pane that allows the user to specify the conditions of application business logic. Editing expressions is explained in [Expression Editing](#).

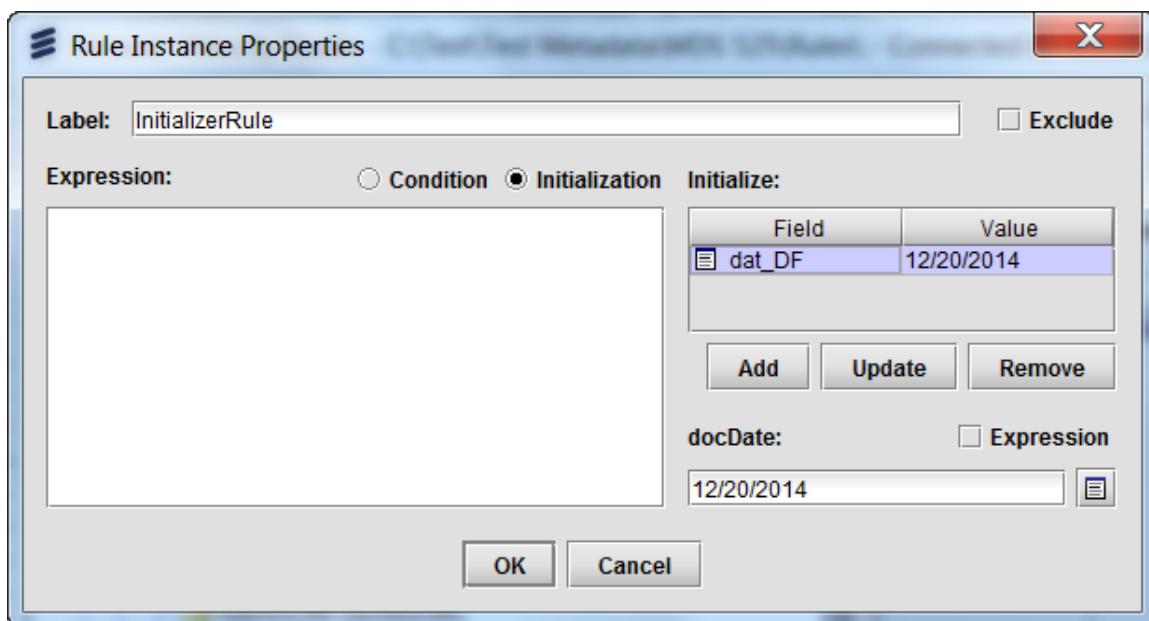
The **Label** and **Exclude** fields are common to all types of instances. The **Label** is mandatory. The **Exclude** check box, when selected, means that the instance will be excluded from the evaluation sequence, but can be invoked from other instances.

### Instance of Initializer Rule

The **Rule Instance Properties** dialog for an instance of an Initialization Rule contains an expression and an initialization list. When the **Condition** radio-button is selected the expression is conditional.



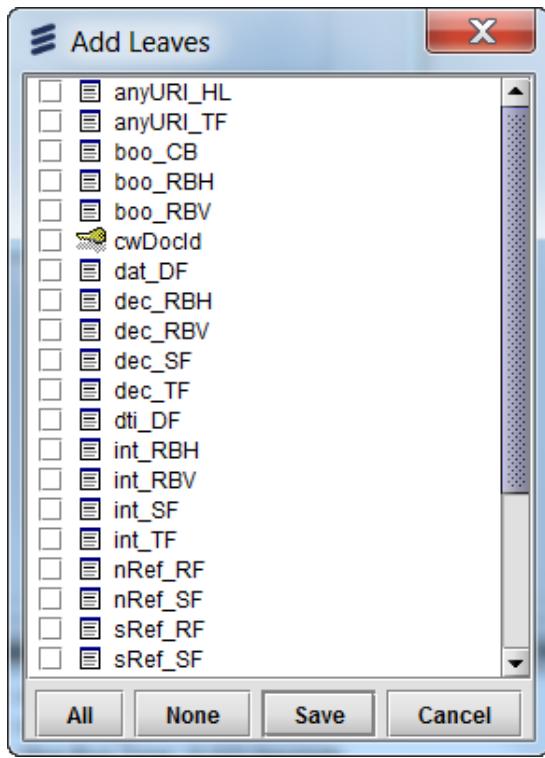
When the **Initialization** radio-button is selected there is no condition.



Each row in the **Initialize** list shows a Document leaf in the **Field** column and its value in the **Value** column. The **Value** can be either an expression, integer, string, etc.. Maintenance of the **Initialize** list is achieved using the buttons located below the list as described in the table below:

Button	Description
Add	Opens the <b>Add Leaves</b> dialog so that new leaves can be added by selecting one or more leaves from the list in the dialog.
Update	Used to change the value in the <b>Initialize</b> list to either the value specified in the <b>Value</b> field located beneath the <b>Initialize</b> list or to <expression> if an expression has been defined in the <b>Expression</b> pane.
Remove	Removes the selected leaf from the list.

The **Add** button will open the Add Leaves dialog button that allows you to select more variables.



If the value is an expression, the **Expression** check box will be checked. The **Expression** pane will then contain the expression for the selected value.

**Note:** Either the internal or label name is shown in the Rule Instance Properties dialog if they are different in the expression panel.

If the Value is a value, the **Expression** check box will be unchecked. The **Expression** pane becomes disabled and the Document leaf will be initialized by the value specified in the **Value** field located beneath the **Initialize** list. The **Value** field will prompt the user for a value of the same data type as the leaf. The label of the **Value** field will change according to the data type name. If the leaf data type is overridden in the current metadata, the overriding data type will be shown. The data type location in the metadata hierarchy can be viewed by clicking the button to the right of the **Value** field. The read-only **Data Type** dialog will open.

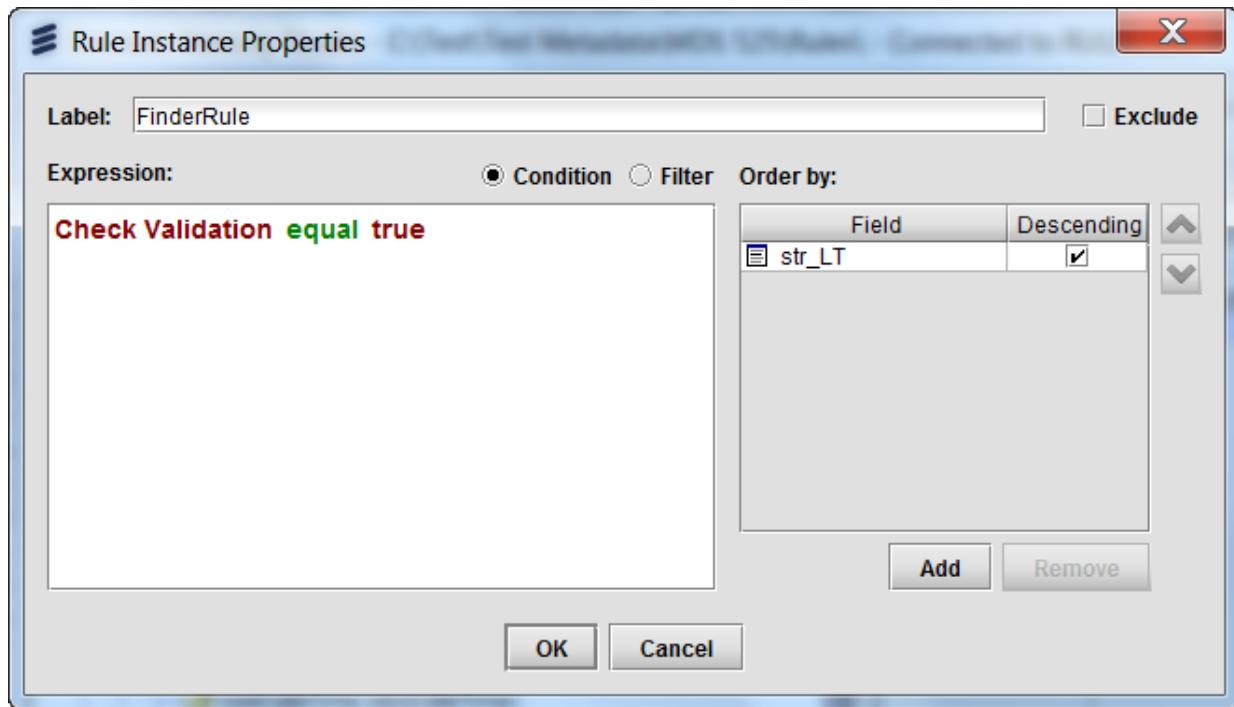
Field	Value
Participant	Admin
DueDate	07/01/2010
cwDocId	hf001

DateTime: 07/01/2010  Expression

**Data Type**  
 Quick pick:   
 Enter object name prefix or pattern(\*,?):   
 Matching items:  
 RuleNS.ruleDate (docDate)  
 RuleNS.ruleDateTime (docDateTime)

#### Instance of Finder Rule

The **Rule Instance Properties** dialog for an instance of a Finder Rule contains radio-buttons for **Condition** and **Filter**. These are used to switch between condition and filter expressions. If the input parameter is not marked as optional by selecting the **Optional** radio-button on the **General** tab of the **Rules** properties, the filter expression will not be available (the radio-buttons will not appear and the only expression visible will be the conditional expression). The conditional expression specifies the condition which must be true for all search results. The filter expression specifies an additional condition.

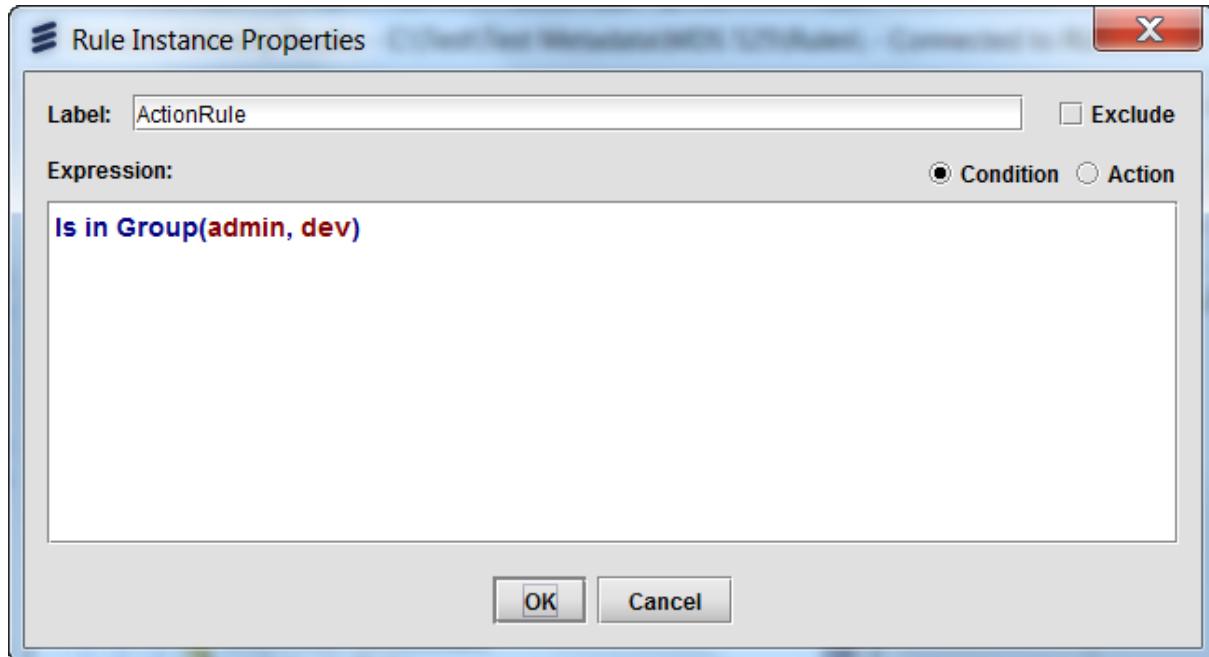


The **Order by** list specifies the order in which the results will be sorted. This list contains one or more Document leaves and a sort order, ascending or descending, for each leaf. The up and down arrow buttons on the right side of the table can be used to change the position of the selected row. Maintenance of the **Order by** list is achieved using the **Add** and **Remove** buttons located below the list which are described in Table.

Button	Description
Add	Opens the <b>Add Leaves</b> dialog so that new leaves can be added by selecting one or more leaves from the list in the dialog.
Remove	Removes the selected leaf from the list.

#### Instance of Action Rule

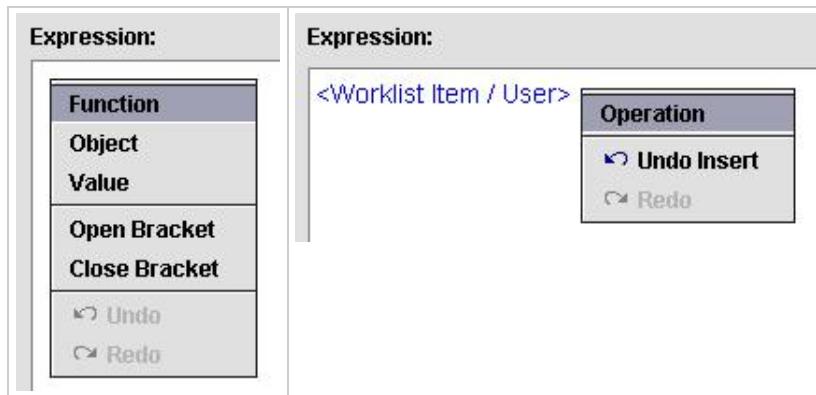
The **Rule Instance Properties** dialog for an instance of an Action Rule contains a conditional expression and an action expression. The **Condition** and **Action** radio-buttons are used to switch between their expression panes. The action expression specifies one of the global or system predefined scripts that will be executed when the condition is true.



#### Expression Editing

An expression consists of operands and operations. The operands can be either a function, object, or value. The operations consist of comparison, mathematical and boolean operations and brackets.

To insert either an operand or an operation, right-click in the **Expression** pane of the **Rule Instance Properties** dialog and select an option from the pop-up menu . The values included in the pop-up menu will depend on the whether the next part of the expression should be an operand or an operation. If the **Function**, **Object** or **Value** command is selected, the **Insert** dialog for inserting an operand will open. If the **Operation** command is selected, the **Insert Operation** dialog will open. The **Open Bracket** and **Close Bracket** commands will add "(" and ")" respectively to the expression. The **Undo** and **Redo** commands can be used for the five most recent changes.

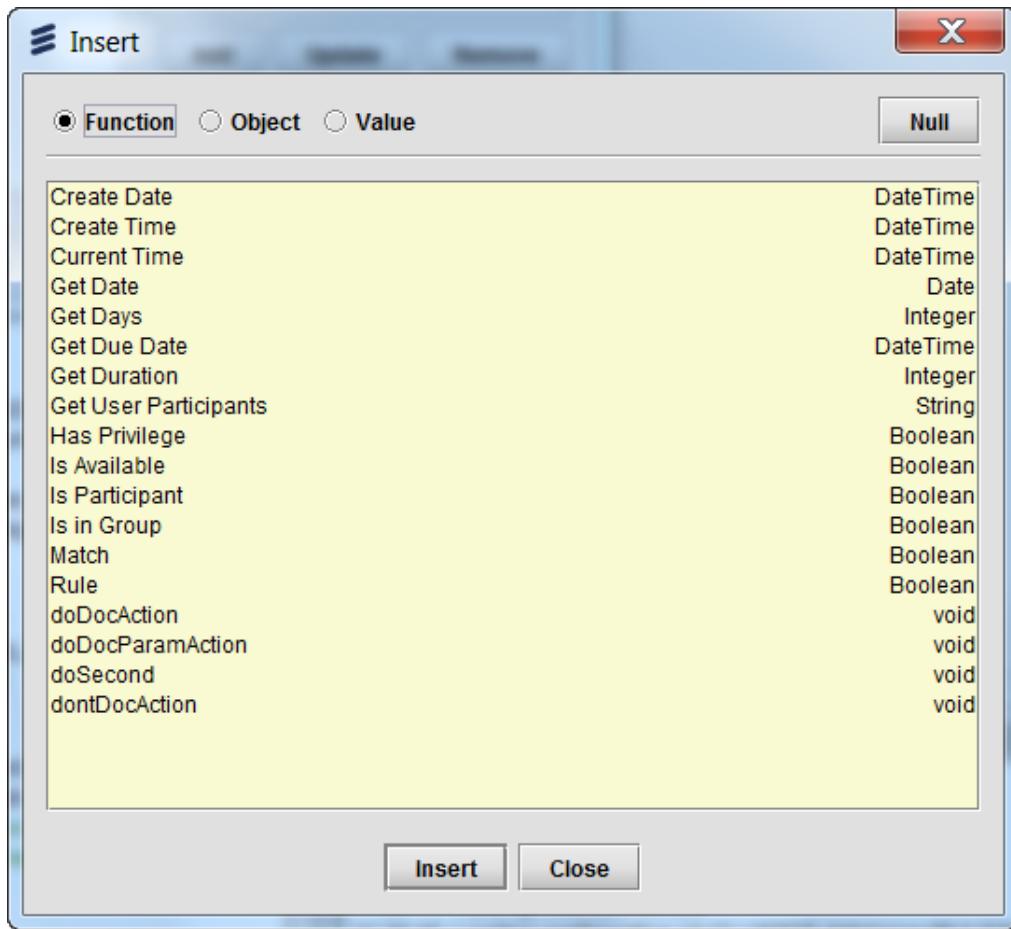


Another option when editing an expression is to press the spacebar when the focus is in the **Expression** pane. This will open the dialog for inserting either an operand or an operation depending on which logically comes next in the expression.

When clicking in the **Expression** pane, the cursor will automatically be positioned at the beginning or end of the expression element, depending on which one is closer. Selection will also be aligned automatically with the expression element borders. Once the cursor is in position, the user can press the Enter key to insert a line break in the cursor position and the Backspace key to remove the line break. When there is no line break, the Backspace key will delete the last element of the expression proceeding the cursor. One or more elements can also be selected and deleted by pressing the Backspace key. Arrow keys can be used to move cursor from element to element.

## Operands

The operands can be either a function, object or value. The **Insert** dialog is used to select an operand and insert it in the expression. The user can use this dialog to change the selected operand type from one to another by selection one of the three radio buttons: **Function**, **Object** or **Value** or by clicking the **Null** button.



## Function

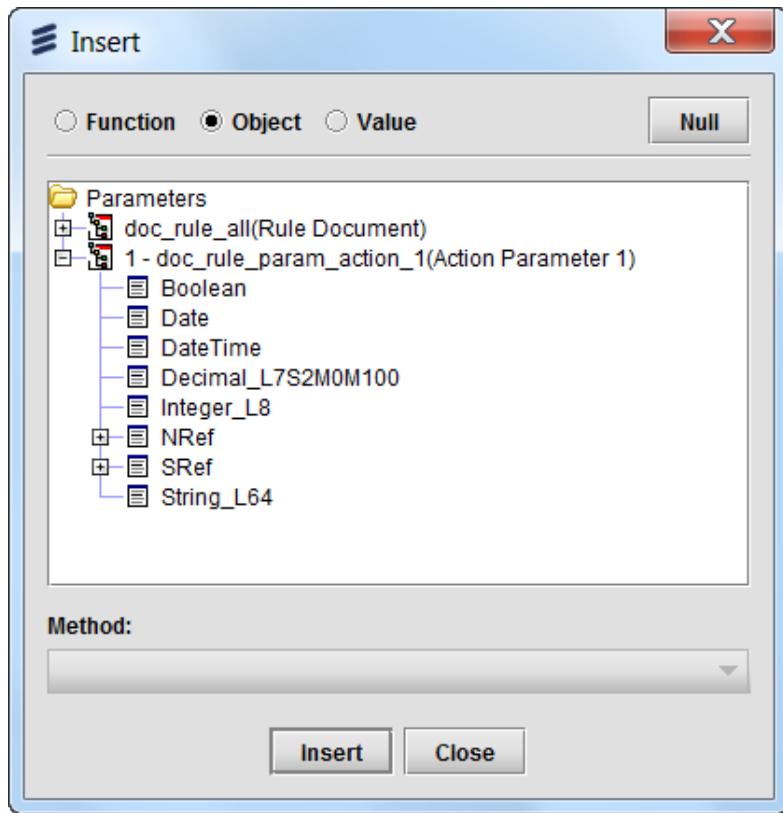
If the **Function** radio button has been selected in the **Insert** dialog then the user will be prompted to select from a list of functions. This list contains system predefined functions (refer to [Appendix A: Functions in the Rule Editor](#)) and all global script functions that are marked for use in the instance expressions. The list of functions can be extended by using [global script](#) objects defined in the metadata.

Once the function is either double-clicked or highlighted and the **Insert** button is pressed, the function will appear in the **Expression** pane. The placeholders for the function parameters are shown in the <> brackets. To define the parameters of the function, select the placeholder and either press the spacebar or right-click and select the **Properties** command from the pop-up menu. The **Properties** dialog will open. The appearance of this dialog will depend on the parameter type. Once a replacement value is defined, click the **Replace** button to replace the placeholder in the expression.



## Object

If the **Object** radio button has been selected in the **Insert** dialog then the user will be prompted to select an object from the hierarchical list and a method to be applied to this object from the **Method** combo box. The objects tree will contain the rule document, input documents and variables. Each input document or variable will have a prefix that shows its sequential number in the business rule properties. For example, input parameter 1 will have a prefix 1 (so, doc1 will appear in the UI as 1 - doc1), the first instance variable will have a prefix V1 (so, vardoc1 will appear in the UI as V1 - vardoc1), the first global variable will have a prefix G1 (so, globaldoc1 will appear in the UI as G1 - globaldoc1), then 2, V2, G2, etc.



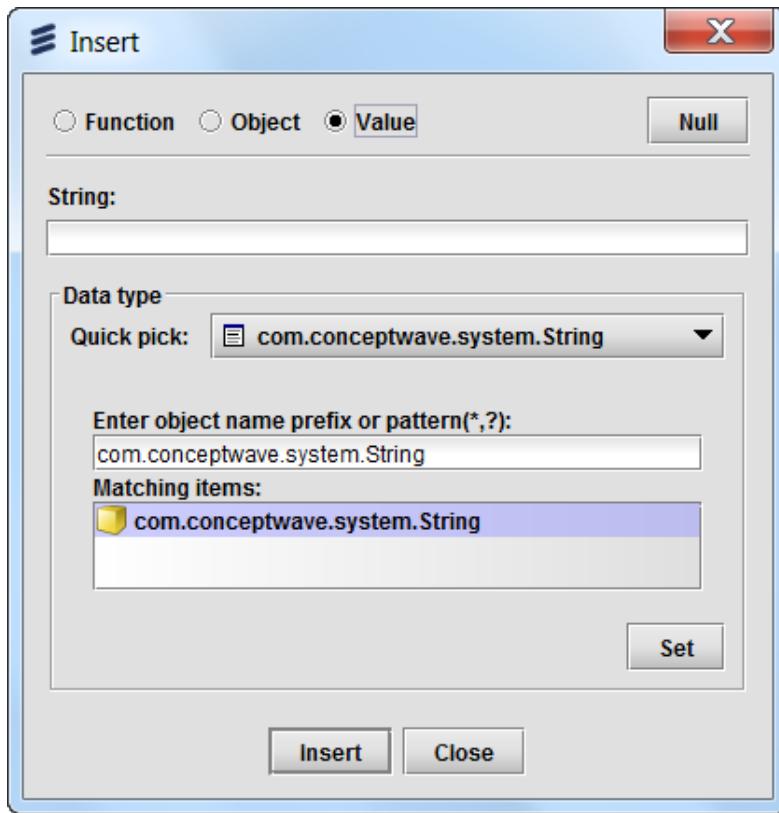
The list in the **Method** combo box will change depending on the object chosen from the objects tree. If the object is a Document leaf, the method list will be defined by the object data type. If the object is a Document, Order or Order Item, the method list will be defined by the object type. For a list of available object methods refer to [Appendix B: Object Methods in The Rule Editor](#). Below is an example of the object and the object method in the expression:

Expression:	Expression:
<Worklist Item / Work Center>	<User / Admin Group>.starts with(<value>)

**Note:** Global variables are additional parameters that can be used in the rule instance expressions the same way as rule instance variables are used. They will appear in the objects tree with the prefix "G" followed by their sequential number and variable label. These global variables are defined in the **Global Variables** dialog by clicking the **Variables** button on the **Instances** tab of the **Rules** properties.

#### Value

If the **Value** radio button has been selected in the **Insert** dialog then the user will be prompted to enter a value with a data type matching the data type of the object that proceeds the value in the expression. If a match cannot be found, the user will be prompted for a string value by default. The **Data type** area below the value field shows the location of the data type object in the metadata hierarchy and will allow the user to select an alternative data type. The alternative data type can be selected from the **Quick Pick** combo box or from the objects tree. The **Quick Pick** combo box contains a list of data types that are used in the leaves of the business rule Documents. The objects tree contains all data types available in the metadata. Press the **Set** button to make the change. This will also add the data type to the **Quick Pick** combo box. The **Restore** button will appear next to the **Set** button in the **Insert** dialog and it is used to restore the initial data type.

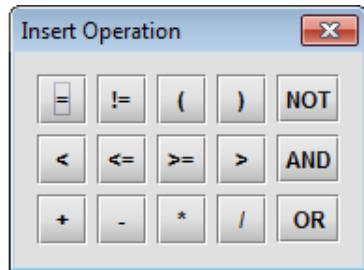


## Null

The "null" value can be added at any place in the expression that requires either an object or string by clicking the **Null** button. The dialog will be closed and "null" will be added to the expression.

## Operations

The **Insert Operation** dialog is used to select an operation and insert it in the expression.



The operations shown in the dialog are supported. When inserted in an expression, the verbal equivalent of the operator is shown for the comparison operators (=, !=, <, <=, >, >=):

<b>Expression:</b>
<Worklist Item / Work Center> <b>equal</b>

Operations can also be entered directly from the keyboard. The table below shows a list of operations and the corresponding keyboard key(s).

Operation	Key(s)	Operation	Key(s)
(	(	+	+
)	)	-	-
=	=	*	*
!=	Ctrl =	/	/
<	<	NOT	!
<=	Ctrl <	AND	&

>	>	OR	
=	Ctrl >		

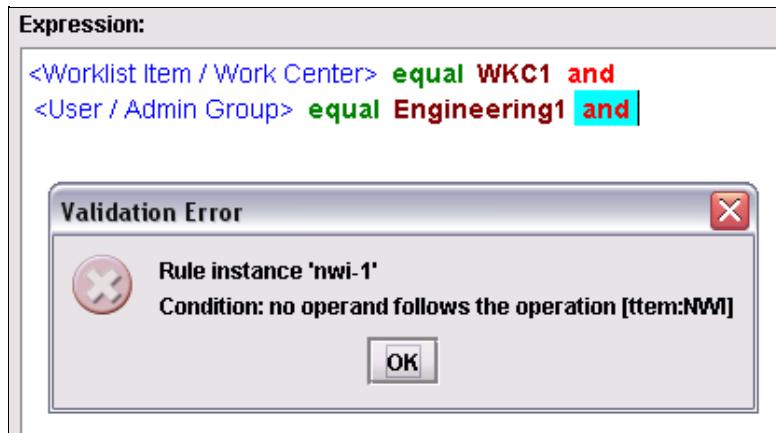
## Modifying Expressions

To modify an element of an existing expression select it with the mouse and press the spacebar. You can also right-click and select the **Properties** command from the pop-up menu. The **Properties** or **Operation Properties** dialog will open, which are very similar to the **Insert** and **Insert Operation** dialogs described above.

You can only replace an operand with another operand and an operation with another operation.

## Validating Expressions

All expressions will be validated for correct syntax when the **OK** button is clicked and an instance is saving. If an error occurs, the **Validation Error** dialog will open with the error message. The item that caused the error, or the closest item to the cause, will be highlighted in the **Expression** pane. A full validation of metadata in the Velocity Studio will also check the integrity of all rule instances to ensure that no deleted elements are used in any expressions. Below is an example of a validation error in an expression:

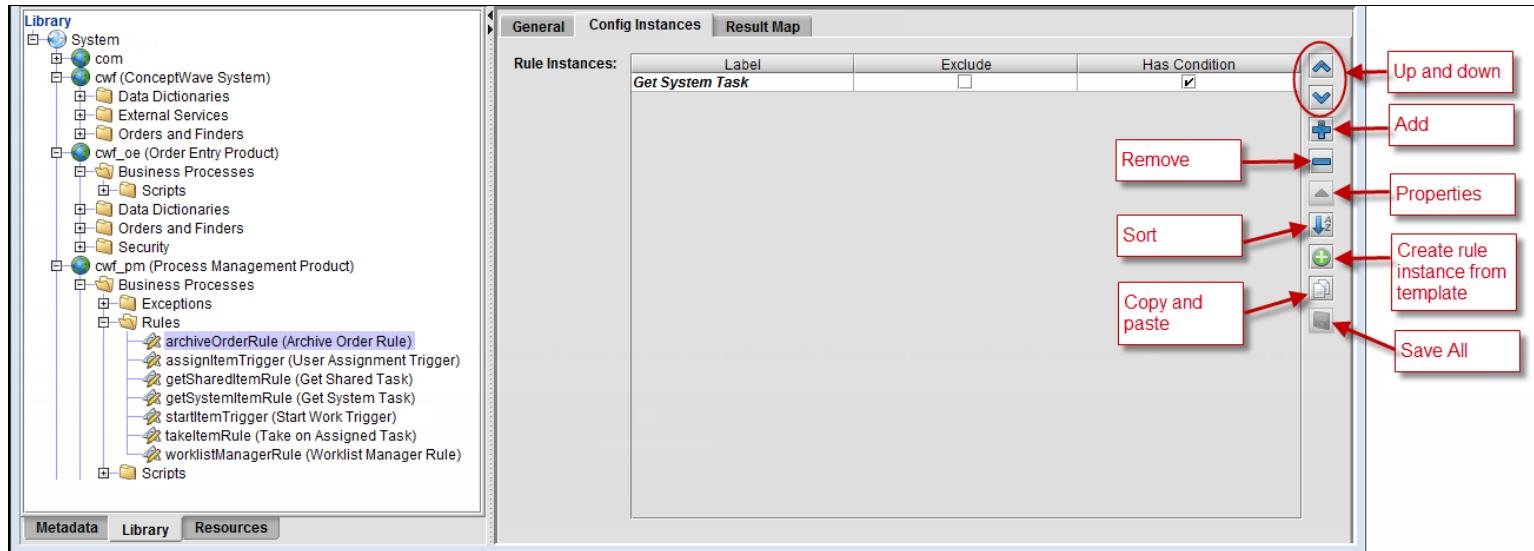


## Rules Config Instances Tab

The **Config Instances** tab allows you to maintain a list of database or deployment instances. These instances are stored in a BLOB within a CLUSTER record in the CWVMINFO table. The **Config Instances** tab is available for metadata and templates (that is, from the **Library** tab), and is enabled when a database connection is available. This tab contains Rule Instance field.

The order of instances in the list defines the order in which the instances will be evaluated at runtime. This order can be changed using the **Up** and **Down** arrow buttons on the right side of the table and the **Sort** button.

**Note:** The Rule properties are read-only. When connected to the database, on the **Config Instances** pane, you can add and change rule instances for a rule either from the library or in your metadata.



The **Rules Instances** tab contains the following fields:

Column	Description
<b>Label</b>	The instance label.
<b>Exclude</b>	When checked, the instance will be excluded from the evaluation sequence, but can be invoked from other instances.
<b>Has Condition</b>	When checked, the instance has a condition.

### Notes:

- The system must be restarted once runtime instances are changed.
- When you have a rule that contains both instance rules and configuration instances, the configuration instances are processed first.

The **Rule Instance** tab contains the following buttons:

Button	Description
<b>Up</b>	Moves the selected instance rule up in the instance list.
<b>Down</b>	Moves the selected instance rule down in the instance list.
<b>Add</b>	Opens the <b>Rule Instance Properties</b> dialog so that a new instance can be added for the rule selected in the <b>Rule</b> combo box. The new instance will be placed below the selected row. New instances can be added to any rule, including rules from libraries.
<b>Remove</b>	Removes the selected instances from the list.
<b>Properties</b>	Opens a <b>Rule Instance Properties</b> dialog allowing the user to view or change the selected instance.
<b>Sort</b>	Sorts the instances list table in ascending order.
<b>Create rule instance from template</b>	Click this button to either enter the object name that you want or select the rule instance available from the dialog.

<b>Copy and Paste</b>	Create a copy of the selected instance and pastes it below the selected row.
<b>Save All</b>	Saves all changes that you have made.

When an instance has the same name as a metadata instance, the instance overrides the metadata instance (that is, a database rule replaces a metadata one). To create a config instance from a template, click the **Add** button. When a database instance name is unique, it is added at runtime to the set of metadata rule instances.

## Migrate Rule Instances from the Command Line

You can migrate rules instances from version 5.1 to version 5.1.11 and later metadata, including templates other than CW system templates, and export legacy rule instances from the CWVMINFO table to an XML file. You can migrate rule instances from the command line by using the ruleInstancesMigrate51 command, which can be found in the <installation\_folder>\designer\env\ folder:

```
ruleInstancesMigrate51 JDBC_URL metadata output_file
```

Where:

- *JDBC\_URL* is any JDBC URL or takes the form *user@host:port:sid*,*user@host:port/service\_name*
- *output\_file* is the name of your XML file

Examples:

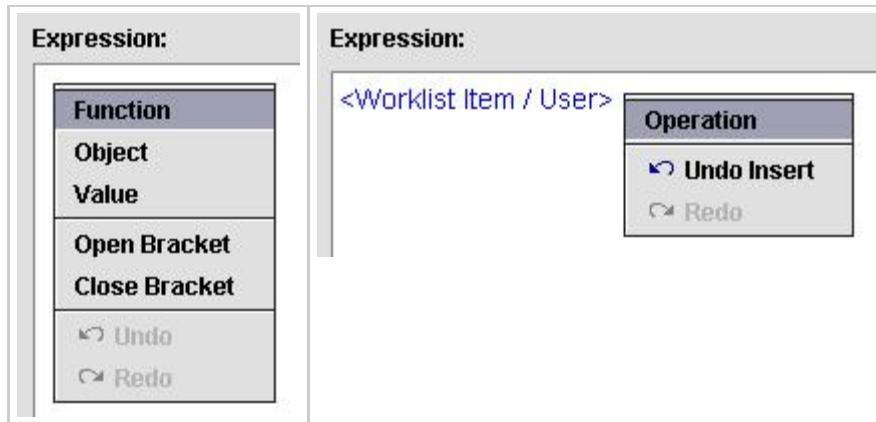
```
ruleInstancesMigrate51 cw@localhost:1521:orcl c:\metadata c:\ruleInstances.xml  
ruleInstancesMigrate51  
"cw@oracle.jdbc.driver.OracleDriver;jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on)(ADDRESS=(PROTOCOL=TCP)(HOST=c:\metadata c:\ruleInstances.xml
```

**Note:** As of version 5.1.11.x, rule instance data has been moved to a separate CWDYNAMICMETADATA table from the CWVMINFO.RULE\_INSTANCES column.

## Expression Editing

An expression consists of operands and operations. The operands can be either a function, object, or value. The operations consist of comparison, mathematical and boolean operations and brackets.

To insert either an operand or an operation, right-click in the **Expression** pane of the **Rule Instance Properties** dialog and select an option from the pop-up menu . The values included in the pop-up menu will depend on whether the next part of the expression should be an operand or an operation. If the **Function**, **Object** or **Value** command is selected, the **Insert** dialog for inserting an operand will open. If the **Operation** command is selected, the **Insert Operation** dialog will open. The **Open Bracket** and **Close Bracket** commands will add "(" and ")" respectively to the expression. The **Undo** and **Redo** commands can be used for the five most recent changes.

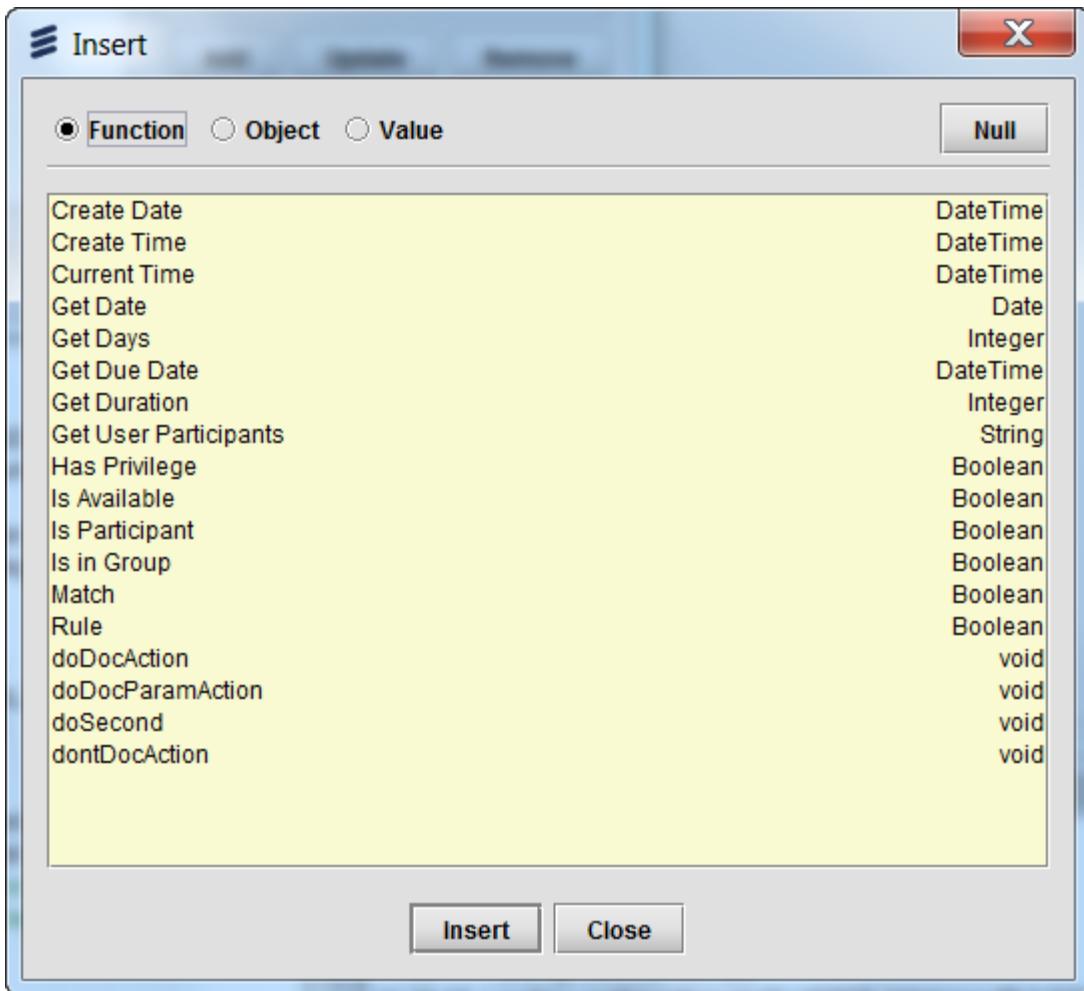


Another option when editing an expression is to press the spacebar when the focus is in the **Expression** pane. This will open the dialog for inserting either an operand or an operation depending on which logically comes next in the expression.

When clicking in the **Expression** pane, the cursor will automatically be positioned at the beginning or end of the expression element, depending on which one is closer. Selection will also be aligned automatically with the expression element borders. Once the cursor is in position, the user can press the Enter key to insert a line break in the cursor position and the Backspace key to remove the line break. When there is no line break, the Backspace key will delete the last element of the expression proceeding the cursor. One or more elements can also be selected and deleted by pressing the Backspace key. Arrow keys can be used to move cursor from element to element.

## Operands

The operands can be either a function, object or value. The **Insert** dialog is used to select an operand and insert it in the expression. The user can use this dialog to change the selected operand type from one to another by selection one of the three radio buttons: **Function**, **Object** or **Value** or by clicking the **Null** button.



## Function

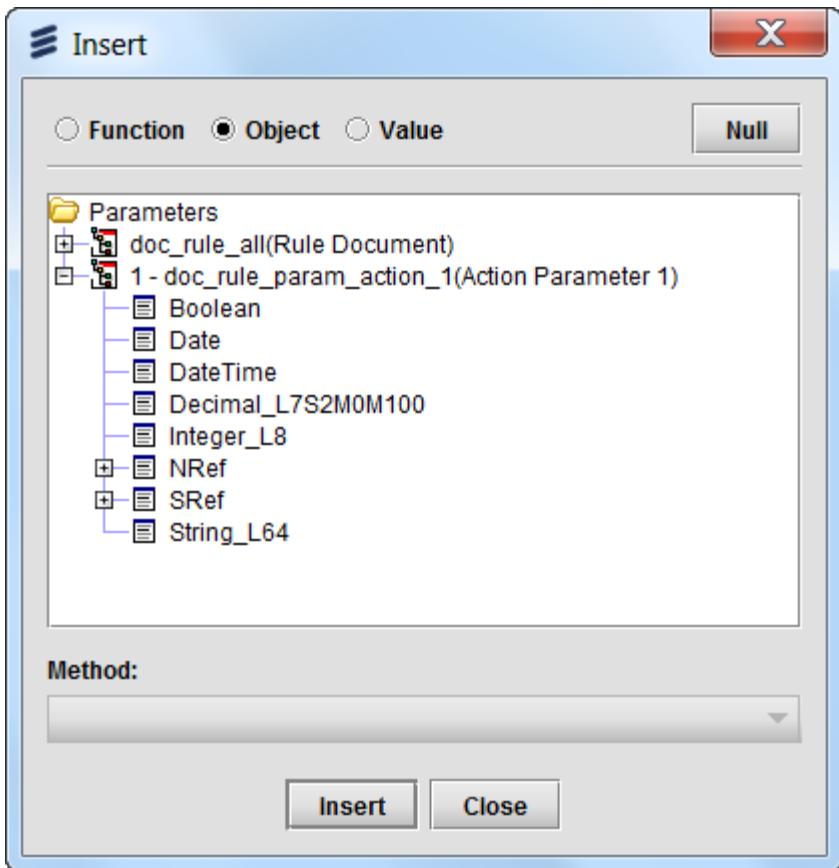
If the **Function** radio button has been selected in the **Insert** dialog then the user will be prompted to select from a list of functions. This list contains system predefined functions (refer to [Appendix A: Functions in the Rule Editor](#)) and all global script functions that are marked for use in the instance expressions. The list of functions can be extended by using [global script](#) objects defined in the metadata.

Once the function is either double-clicked or highlighted and the **Insert** button is pressed, the function will appear in the **Expression** pane. The placeholders for the function parameters are shown in the <> brackets. To define the parameters of the function, select the placeholder and either press the spacebar or right-click and select the **Properties** command from the pop-up menu. The **Properties** dialog will open. The appearance of this dialog will depend on the parameter type. Once a replacement value is defined, click the **Replace** button to replace the placeholder in the expression.

<b>Expression:</b>
<b>Is in Group(&lt;user&gt;, &lt;group&gt;)</b>

## Object

If the **Object** radio button has been selected in the **Insert** dialog then the user will be prompted to select an object from the hierarchical list and a method to be applied to this object from the **Method** combo box. The objects tree will contain the rule document, input documents and variables. Each input document or variable will have a prefix that shows its sequential number in the business rule properties. For example, input parameter 1 will have a prefix 1 (so, doc1 will appear in the UI as 1 - doc1), the first instance variable will have a prefix V1 (so, vardoc1 will appear in the UI as V1 - vardoc1), the first global variable will have a prefix G1 (so, globaldoc1 will appear in the UI as G1 - globaldoc1), then 2, V2, G2, etc.



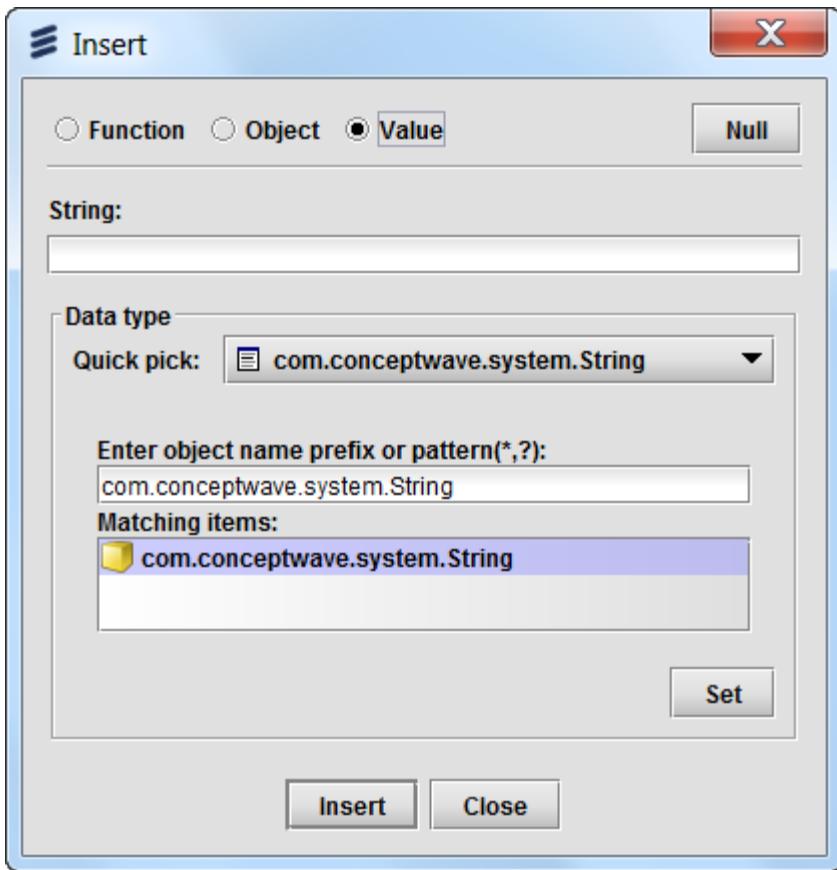
The list in the **Method** combo box will change depending on the object chosen from the objects tree. If the object is a Document leaf, the method list will be defined by the object data type. If the object is a Document, Order or Order Item, the method list will be defined by the object type. For a list of available object methods refer to [Appendix B: Object Methods in The Rule Editor](#). Below is an example of the object and the object method in the expression:

<b>Expression:</b> <code>&lt;Worklist Item / Work Center&gt;</code>	<b>Expression:</b> <code>&lt;User / Admin Group&gt;.starts with(&lt;value&gt;)</code>
--	--

**Note:** Global variables are additional parameters that can be used in the rule instance expressions the same way as rule instance variables are used. They will appear in the objects tree with the prefix "G" followed by their sequential number and variable label. These global variables are defined in the **Global Variables** dialog by clicking the **Variables** button on the **Instances** tab of the **Rules** properties.

#### Value

If the **Value** radio button has been selected in the **Insert** dialog then the user will be prompted to enter a value with a data type matching the data type of the object that proceeds the value in the expression. If a match cannot be found, the user will be prompted for a string value by default. The **Data type** area below the value field shows the location of the data type object in the metadata hierarchy and will allow the user to select an alternative data type. The alternative data type can be selected from the **Quick Pick** combo box or from the objects tree. The **Quick Pick** combo box contains a list of data types that are used in the leaves of the business rule Documents. The objects tree contains all data types available in the metadata. Press the **Set** button to make the change. This will also add the data type to the **Quick Pick** combo box. The **Restore** button will appear next to the **Set** button in the **Insert** dialog and it is used to restore the initial data type.

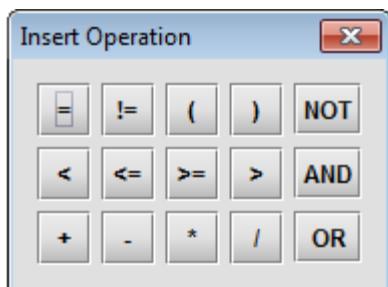


## Null

The "null" value can be added at any place in the expression that requires either an object or string by clicking the **Null** button. The dialog will be closed and "null" will be added to the expression.

## Operations

The **Insert Operation** dialog is used to select an operation and insert it in the expression.



The operations shown in the dialog are supported. When inserted in an expression, the verbal equivalent of the operator is shown for the comparison operators (=, !=, <, <=, >, >=):

<b>Expression:</b>
<Worklist Item / Work Center> equal

Operations can also be entered directly from the keyboard. The table below shows a list of operations and the corresponding keyboard key(s).

Operation	Key(s)	Operation	Key(s)
(	(	+	+
)	)	-	-

=	=	*	*
!=	Ctrl =	/	/
<	<	NOT	!
<=	Ctrl <	AND	&
>	>	OR	
>=	Ctrl >		

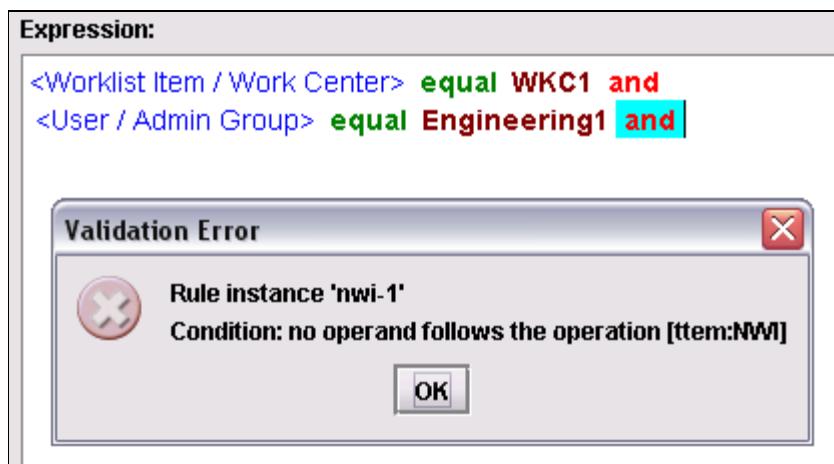
## Modifying Expressions

To modify an element of an existing expression select it with the mouse and press the spacebar. You can also right-click and select the **Properties** command from the pop-up menu. The **Properties** or **Operation Properties** dialog will open, which are very similar to the **Insert** and **Insert Operation** dialogs described above.

You can only replace an operand with another operand and an operation with another operation.

## Validating Expressions

All expressions will be validated for correct syntax when the **OK** button is clicked and an instance is saving. If an error occurs, the **Validation Error** dialog will open with the error message. The item that caused the error, or the closest item to the cause, will be highlighted in the **Expression** pane. A full validation of metadata in the Velocity Studio will also check the integrity of all rule instances to ensure that no deleted elements are used in any expressions. Below is an example of a validation error in an expression:



## Rules Result Map Tab

The **Result Map** tab is used to map the rule parameters (**Rule document** and input Documents) and/or global variables to the **Finder result** Document defined on the **General** tab. This tab is only visible if a Rule has the **Type** of *finder* on the **General** tab and it will only be enabled if a **Finder result** Document is defined. The user can select one of the pre-created Conversion Maps (refer to section on Conversion Maps) to do the mapping. Not all fields of the Finder Rule result Document need to be mapped. The Finder Rule result Document does not need to be mapped to the database.

The screenshot shows the 'Result Map' tab selected in a software interface. A modal dialog box titled 'Select Conversion Map' is displayed in the center. The dialog contains two dropdown menus: 'Source document:' set to 'com.conceptwave.system.Docu...' and 'Conversion map:' set to '<NONE>'. At the bottom are 'Save' and 'Cancel' buttons. In the background, the main window shows a table with two columns: 'Source Document' and 'Conversion Map'. The first row lists 'Document (system)' under 'Source Document' and is empty under 'Conversion Map'. A 'Modify' button is located at the bottom right of the main window.

The table in the **Result Map** tab contains a list of the existing mapping Document sources in the **Source Document** column including the **Rule document** and input Documents defined on the **General** tab. The **Conversion Map** column shows the selected Conversion Map between the **Source Document** and the **Finder result** Document defined on the **General** tab. Clicking the **Modify** button opens the **Select Conversion Map** dialog that shows the available Conversion Map selections for the given pair of objects.

## Appendix A: Functions in The Rule Editor

---

The following system defined functions are available in the rule instance editor.

### Create Date

Creates a Date object.

Returns: A new Date object.

Parameters:

Name	Type	Description
year	integer	A year.
month	integer	A month (0 - 11).
day	integer	A day of the month (1 - 31).
hour	integer	An hour (0 - 23).
minutes	integer	Minutes (0 - 59).
seconds	integer	Seconds (0 - 59).

### Create Time

Creates a Date object with provided time values.

Returns: A new Date object.

Parameters:

Name	Type	Description
hour	integer	An hour (0 - 23).
minutes	integer	Minutes (0 - 59).
seconds	integer	Seconds (0 - 59).

### Current Time

Creates a Date object with the current date and time.

Returns: The created Date object.

Parameters: None.

### Get Date

Extracts a date from either a Date or DateTime object.Sets the time component of the date to 00:00:00.

Returns: A Date object.

Parameters:

Name	Type	Description
date	date	A Date or DateTime object.

### Get Days

Calculates the number of working days between two dates for a given calendar.

Returns: The number of working days between the dates.

Parameters:

Name	Type	Description
calendar	Calendar	Metadata name of calendar.
from	date	A Date or DateTime object.
to	date	A Date or DateTime object.

## Get Due Date

Calculates the due date by adding a duration to a starting date. Non-working hours are skipped in the calculation.

*Returns:* A Date object.

*Parameters:*

Name	Type	Description
calendar	Calendar	Metadata name of calendar.
date	date	Starting date.
duration	integer	Duration to add to date.
duration type	Integer	Type of duration. May be one of: Calendar._DATE_DAY, Calendar._DATE_HOUR, or Calendar._DATE_MINUTE.

## Get Duration

Calculates duration in working seconds between two dates for a given calendar.

*Returns:* Duration in seconds.

*Parameters:*

Name	Type	Description
calendar	Calendar	Metadata name of calendar.
from	date	A Date or DateTime object.
to	date	A Date or DateTime object.

## Get User Participants

Gets a list of participants that a user belongs to.

*Returns:* A string containing the list of participants.

*Parameters:*

Name	Type	Description
user	string	User ID.

## Has Privilege

Return true if the user has the specified privilege.

*Returns:* True if the user has the privilege.

*Parameters:*

Name	Type	Description
user	string	User ID.
privilege	group	Privilege name.

## Is Available

Returns true if the user is available for worklist distribution.

*Returns:* True if the user is available.

*Parameters:*

Name	Type	Description
user	string	User ID.

## Is Participant

Returns true if the user belongs to the specified participant.

*Returns:* True if the user belongs to the participant.

*Parameters:*

Name	Type	Description
user	string	User ID.
participant	participant	Participant name.

## Is in Group

Returns true if the user is a member of the specified group.

*Returns:* True if the user is in the group.

*Parameters:*

Name	Type	Description
user	string	User ID.
group	string	Group name.

## Match

Searches a string for a given pattern and returns true if the match is found.

*Returns:* True if a match is found.

*Parameters:*

Name	Type	Description
text	string	String to be searched.
pattern	string	The pattern string that may contain the wildcard characters ? (represents any single character) and * (represents any number, including zero, of any characters).

## Rule

Evaluates another instance of the same business rule.

*Returns:* A boolean value as a result of instance evaluation.

*Parameters:*

Name	Type	Description
rule instance	Rule instance	The full name of the rule instance that will be evaluated.

The following system functions are available only in the variable [definition](#).

## Document

Reads a document from the database.

*Returns:* A document.

*Parameters:*

Name	Type	Description
key	Document leaf	Document ID.

## Order

Reads an order from the database. The business rule document or items from its hierarchy cannot be used as the function parameter.

*Returns:* An order.

*Parameters:*

Name	Type	Description
key	Order	Top level order.

## Order Item

Reads an order item.

*Returns:* An order item.

*Parameters:*

Name	Type	Description
key	Order item	Order item.

## User Profile

Gets the current user profile document.

*Returns:* A user profile document.

*Parameters:* None.

## Appendix B: Object Methods in the Rule Editor

---

The following object methods are available in the rule instance editor.

### **String object methods**

#### **contains**

Searches for the first occurrence of the specified substring in the string object.

*Returns:* True if the string argument occurs as a substring within the string object.

*Parameters:*

Name	Type	Description
value	string	Any string.

#### **ends with**

Tests if the string ends with the specified substring.

*Returns:* True if the string object ends with the specified string argument .

*Parameters:*

Name	Type	Description
value	string	Any string.

#### **starts with**

Tests if the string starts with the specified substring.

*Returns:* True if the string object starts with the specified string argument.

*Parameters:*

Name	Type	Description
value	string	Any string.

#### **match**

Searches the string for a given pattern and returns true if the match is found.

*Returns:* True if a match is found.

*Parameters:*

Name	Type	Description
pattern	string	The pattern string that may contain the wildcard characters ? (represents any single character) and * (represents any number, including zero, of any characters).

### **String or number object methods**

## in

Returns true if the string matches one of the specified substrings or array members.

*Returns:* True if the entire string matches one of the substrings or array members.

*Parameters:*

Name	Type	Description
value(s)	string	A string containing comma separated substrings (the substrings must not contain any commas).

## ***Document object methods***

### **parent**

Gets a parent document containing a reference to this document.

*Returns:* The parent document.

*Parameters:* None.

### **order**

Gets a top-level order for this document.

*Returns:* The order.

*Parameters:* None.

## ***Date or DateTime object methods***

### **year**

Extracts a year from the object.

*Returns:* A full year number.

*Parameters:* None.

### **month**

Extracts a month from the object.

*Returns:* A month number.

*Parameters:* None.

### **day**

Extracts a day from the object.

*Returns:* A day number.

*Parameters:* None.

## ***Time or DateTime object methods***

### **hour**

Extracts an hour from the object.

*Returns:* An hour number.

*Parameters:* None.

### **minutes**

Extracts the minutes from the object.

*Returns:* Number of minutes.

*Parameters:* None.

### **seconds**

Extracts the seconds from the object.

*Returns:* Number of seconds.

*Parameters:* None.

## Action Auditing

---

Action auditing refers to tracking specific information when an action is invoked within the application. The action auditing feature involves the following:

- [Associate an action with a logging action](#), meaning that when an action is invoked, the associated logging action is also invoked. The logging action then provides a script to fill in the field values of a system logging document, which you can override.
- [Keep track of certain user information](#) during the login process and use a system finder to search for users by relying on the information gathered.

You can also audit system events, such as user login, logout, timeout, login failure, password change, and so on by [using a generic user event table](#).

## Associate an Action with a Logging Action

Before you complete the procedure to associate an action with a logging action, you need to create the following:

- [Create a system document](#)
- [Create a system global script](#)

### Create a System Document

To create a system document, do the following:

1. Right-click the namespace and select **New** from the menu.
2. On the **Choose a metadata type** page, expand the **Data Dictionaries** node, select **Document**, and then click the **Next** button.
3. On the New Document page, do the following:
  - a. Fill in the **Name** and **Label** fields of your new system document.
  - b. For the **Extends** field, click the drop-down menu, select **cwf.actionAuditDoc**, and then click the **Finish** button.

### Create a System Global Script

To create a system global script, complete these steps:

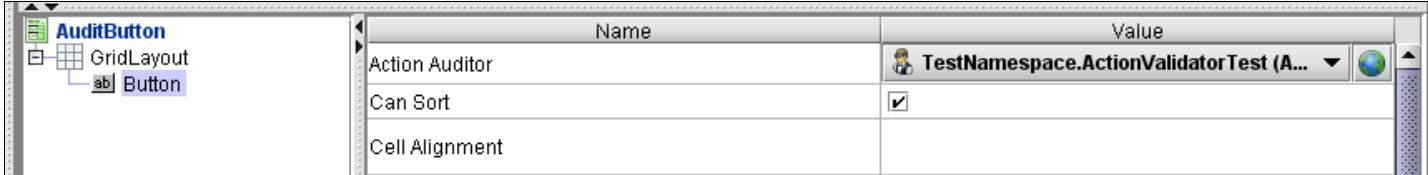
1. Right-click the namespace and select **New** from the menu.
2. On the **Choose a metadata type** page, expand the **Global Scripts** node, select **Action Auditor**, and then click the **Next** button.
3. On the New Action Auditor page, fill in the **Name** and **Label** fields of your new system global script, and then click the **Finish** button.

To associate an action with a logging action, complete these steps:

1. Use your Web browser to access the Configuration application. Once you are logged in, click the **System** vertical tab, and then click the **Logging** vertical tab.
2. On the [Logging](#) page, select the **Enable action logging** parameter, and then click the **Save** button.
3. In Velocity Studio, create an Action Validator object in your metadata by doing the following:
  - a. Right-click the namespace and select **New** from the menu.
  - b. On the **Choose a metadata type** page, expand the **Global Scripts** node, select **Action Auditor**, and then click the **Next** button. On the New Action Auditor page, you can specify the following fields:

Field	Description
<b>Name</b>	The name of your Action Validator object.
<b>Label</b>	The label name of your Action Validator object.
<b>Description</b>	A brief description on the Action Validator object.
<b>Category</b>	The category that the Action Validator object belongs to.
<b>Action Name</b>	The unique name of the Action.
<b>Audit Script</b>	Click the drop-down to select an audit script from the list provided. Alternatively, you can click the <b>Globe</b> icon to find a script within your metadata.

4. Create a User Interface with a [Menu](#), [Button](#), or [Image](#) element.
5. Click the element (the following screenshot uses the Button element), and then click the **Action Auditor**'s drop-down menu to select the **Action Validator** object.



6. Start runtime by clicking **Runtime > Run** from the menu bar.
7. In runtime, click the Menu, Button, or Image element. Proceed to check the database for the document recreated by the Action Validator.

### Notes:

- You can use the `Global.enableActionLogging()` API to either turn action logging on or off. This API sets a flag in the AVM that takes precedence over the flag set by the Configuration application. When the API is running, if an element with an action validator is activated and the action validation is

enabled by both the Configuration application and the AVM, the action validator's script is run after the original action. When enableActionLogging is set to false, no documents are saved in the database. Otherwise, when the API is set to true, documents are created. The finder that shows user action information is cwf.userActionFinder.

- You can also create an [event handler](#) using the ACTION\_AUDIT\_LOG event, which creates an auditLog document, and then runs the specified audit script. This event takes the following mandatory parameters:
  - ScriptName
  - category
  - actionPerformed

## Keep Track of and Use a System Finder to View User Information

The cwf.userLogDocument system user document keeps track of the following details during the login process:

Search Criterion	Description
Node ID	The unique node identification number associated with the user.
User Name	The user who is logged in.
Login Time	The time that the user logged in to the system.
Logout Time	The time that the user logged out of the system.

On user login, the user information document is always updated and saved. When the user logs out, the user information document is also updated and saved.

The system document finder, cwf.userFinder, uses the system user document as both the input and output documents.

To keep track of specific user information during the login process and to search for users by relying on the information gathered, perform the following steps:

1. In the Configuration application, click the **System** vertical tab, and then click the **Logging** horizontal tab to display the system logging levels.
  2. Check the CWPUSERID database table to make sure that both the **LOGIN** and **LOGOUT** fields are updated whenever users log in and out. In addition, ensure that these tables are only updated when the configuration flag is enabled.
  3. Check that the logout time is recorded in cases other than direct logout using the **Administration** menu, such as when the user session expires.
- Note:** If the server shuts down, the logout time is not recorded.
4. To view the user login information, use the cwf.userFinder finder.

## Audit System Events Using a Generic User Event Table

A table is available, which allows you to audit system events, such as user login, logout, timeout, login failure, password change, and so on.

The templateInit script method is available in the cwf\_oe namespace. The system uses this script method to register all system events. The AVM automatically calls it during system startup.

The loggingUserAction script method, also in cwf\_oe namespace, populates the cwf.actionAuditDoc from user action and persists it to the database. The loggingUserAction method takes following parameters:

- USERID
- ACTIONNAME
- TIME
- NODE\_ID

The following constants for pre-defined system events are accessible from the JavaScript API:

- CW\_LOGIN
- CW\_LOGOUT
- CW\_TIMEOUT
- CW\_LOGIN\_FAILED
- CW\_PASSWORD\_CHANGE
- CW\_WORKLIST
- CW\_MILESTONE
- CW\_SYS\_STARTUP
- CW\_SYS\_STOP
- CW\_CONFIG\_RELOAD

The following is an example of how to use both the templateInit and loggingUserAction scripts:

```
function cwf_oe:templateInit(){  
    subscribeForEvent('cwf_oe: loggingUserAction', 'USER_LOGIN', 'CW');  
    subscribeForEvent('cwf_oe: loggingUserAction', 'USER_LOGOUT', 'CW');  
    subscribeForEvent('cwf_oe: loggingUserAction', 'USER_TIMEOUT', 'CW');  
    ...  
}
```

In your metadata, use subscribeForEvent() to register event and publishEvent() to trigger event. You also need to open the Configuration application and ensure that your [events](#) are turned on.

See the following related links for more information:

- [Action auditing](#)
- [Event handlers](#)

## Decision Trees

A **Decision Tree** is a metadata object that encapsulates complex logic using BPML (Business Process Modeling Language) model, as well as executing at runtime. It presents the complex decision flow graphically, as a tree of decision making units called **Activities**. The Decision Tree reuses the basic ideas of BPML process model (see section [Business Processes](#)). This encourages a piecewise, divide-and-conquer approach to defining a decision logic. Compared to pure scripting, Decision Tree can increase readability and decrease overall maintenance effort when you are faced with managing complicated business logic.

Although the metadata of Decision Tree is akin to Business Process Metadata, they serve different purposes. Decision Tree is to be logically-oriented, to perform tasks based on criteria. Conversely, Business Process is to carry out the workflow required to coordinate the course of business activities. The expected execution life-span of a Decision Tree instance is instantaneous, while completing business process instances are to be in hours, days, weeks or months (for example, Order orchestration). From a functional perspective, the key differences are:

Decision Tree	Business Process
Decision Tree can run in any AVM environment (in UI Server or in Process Engine)	Processes can only be executed by Process Engines
Runs synchronously as uninterrupted units in a single session (server thread).	Multi-threaded, dedicated technology (process engine) to execute and maintain process states.
Not possible to have persistent "in-progress" instance. As such, no need to indicate instance status, or migration of in-progress instances.	Persists progress in database upon completing activities in workflow; has <a href="#">process status</a> and need to <a href="#">migrate in-progress process instances</a> when Process definition is changed.
Operations can only be of type <i>one-way</i> or <i>request-response</i> , but not <i>notification</i> , because it cannot wait for external interface to asynchronously trigger the flow of Decision Tree.	Operations can be of any type.

### Activities in Decision Tree

Similar to Activities in Processes, an Activity in Decision Tree is a component that performs a specific function within the tree, such as message passing (that is, operation) or execute a script. Activities are either *atomic* or *complex*. An *atomic* activity cannot have other sub-activities beneath them in the tree. Atomic activities tend to be "executable" activities, performing Decision Tree actions. A *complex* activity is composed of other activities. Complex activities are "control" activities that directs the execution of these sub-activities.

Conceptually, the Decision Tree presents the logic by the use of the following complex activities:

1. **Sequence** - executes the sub-activities one-by-one, in sequence
2. **Switch** - spans the control logic to branches, where the execution/selection of each branch is conditional to the case sub-activity. This is equivalent to the switch/case statement available in most programming languages.
3. **While** - repeats the sub-activities in sequence, until the condition in the While activity becomes false. It can contain the **Break** sub-activity to interrupt the loop.

As well, there is a **Decision Tree** Activity which runs another Decision Tree. This is equivalent to function call in programming languages.

### Notable properties in Decision Tree

A Decision Tree can contain a parameter list (see **Parameters** tab), which is a list of (name, type) pairs. It can also contain an initialization script and completion script (see **Actions** tab), which is run before the first activity and after terminal activity respectively. Together, a Decision Tree is capable of holding parameters, initializing them before the Decision Tree's execution.

By default, Decision Tree runs as a single database transaction, but this can be altered by **Tree Transaction** property in **General** tab.

### Create New Decision Tree

To create a new Decision Tree:

In the **Metadata** tab of **Navigation** pane, either:

1. Right-click a Namespace and select **New ...**
2. **New Metadata Object** wizard appears as a popup. Expand **Business Processes**, and select **Decision Tree**, and then click the **Next** button.
3. Step two of wizard appears. Type in the name of Decision Tree (must conform to JavaScript naming conventions), and other parameters (see [Decision Tree general properties](#) for description). Then click **Finish**.

OR

1. Right-click the **Decision Trees** folder or the **Business Processes** folder in a Namespace, if it is present. Select **New Decision Tree**
2. Follow step3 from above.

After the Decision Tree is created, the newly created Decision Tree is added under **Decision Trees** folder.



## Decision Tree General Properties

The general Decision Tree properties are defined on the **General** tab.

General	Parameters	Diagram	Actions
<b>Name:</b> creditAssure	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated		
<b>Label:</b> creditAssure			...
<b>Description:</b>  			
<b>Max. Iterations:</b>  <input type="text"/>			
<b>Tree Transaction:</b> <input checked="" type="checkbox"/>			

Field	Description
<b>Name</b>	Mandatory. Name of the Decision Tree within the namespace.
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Label</b>	Mandatory. Label to identify the Decision Tree.
<b>Description</b>	Description of the Decision Tree for documentation purposes.
<b>Max Iterations</b>	Number from one to 64,000 or zero. If zero there is no limitation of the number of operations. If more than zero the Decision Tree will throw exception if any of its activities is executed more than the specified number of times.
<b>Tree Transaction</b>	If checked, the whole Decision Tree runs as a single DB transaction. In unchecked, each script is a separate transaction and can control its auto-transaction mode of operation. Default is checked.

The notable field is the **Tree Transaction**, which is checked by default, which runs Decision Tree runs as a single database transaction. However, if unchecked, it allows *activity mode* in which each Decision Tree activity is independent transaction(s). In *activity mode*, the activity scripts may turn off the auto-transaction mode, and perform many database transactions within a single script or perform each script as independent transaction.

## Decision Tree Parameters

The Decision Tree may contain a list of parameters, as a means to generalize its logic flow and execution. Define this list of parameters, in name/type pairs, in the **Parameters** tab. The type may be one of the following specific metadata object:

- Data Type
  - Document
  - Order
  - Data Structure

or one of the following generic metadata object types:

- Document
  - Order
  - Data Structure
  - Object

Note that if the type is *Object*, the parameter can be any scriptable object.

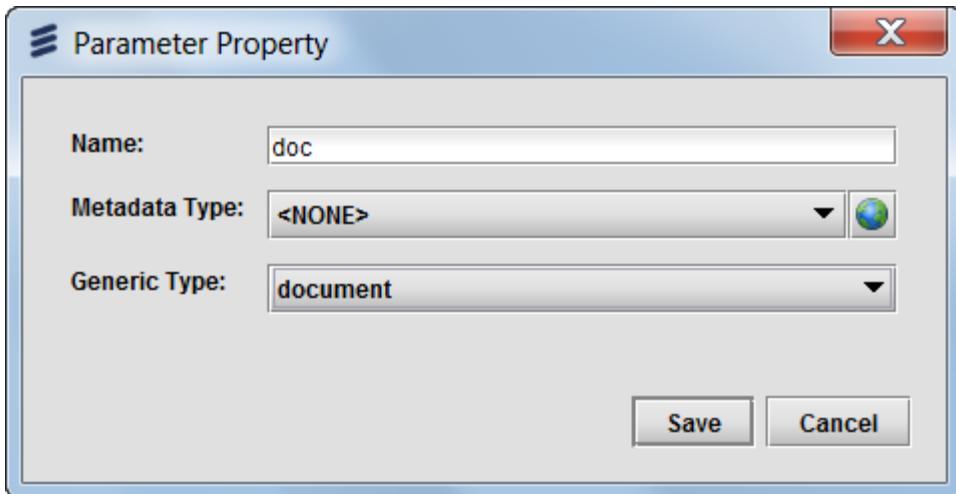
General	Parameters	Diagram	Actions
Name			Type
doc			document
equipmentID			catalogBase:equipmentId

At runtime, the Decision Tree has read/write access to its parameters that are properties of '*this*' object accessible by parameter name. For example, the value of parameter *p1* can be accessed and set by notation "*this.p1*".

**Note:** Refer to the JavaScript Extensions HTML documentation for detailed information on JavaScript objects and their methods.

**The parameter names must not start with prefix cw.** For certain activities, the Decision Tree may create temporary "system" parameters which names start with prefix cw.

The **Parameter Properties** dialog appears upon adding or editing a parameter.



Field	Description
Name	Mandatory. Name of the parameter.
Metadata type	Choose a specific metadata object that is of Data Type, Document, Order or Data Structure as the parameter's type. Choosing an entry here will nullify the <b>Generic type</b> field.
Generic type	Choose from generic <i>Document</i> , <i>Order</i> , <i>Data Structure</i> or <i>Object</i> metadata type as the parameter's type. Choosing an entry here will set the <b>Metadata type</b> field as <None>.

#### Passing parameters from one Decision Tree to another

The *Decision Tree* Activity allows invoking a new Decision Tree and passing some of the parameters of the calling Decision Tree to the called Decision Tree. The parameters are passed by address (not value), which means the changes in the content of complex parameters in the called Decision Tree are propagated to the calling Decision Tree. However, the two trees have independent parameter lists, so if the called tree changes the parameter value, it is not propagated back to the calling tree. Particularly, changes in simple parameters (such as strings, numbers and dates) is never propagated back to the calling tree.

#### Example:

Decision Tree *tree1* has two parameters:

- name *pa1* of type *document*, which has leaf with name *s*
- name *pa2* of type *string*

Decision Tree *tree2* has two parameters:

- name *pb1* of the same type as *pa1* type
- name *pb2* of the same type as *pa2* type

Decision Tree *tree1* contains a Decision Tree Activity that calls *tree2*, and maps the parameters of "tree1" to "tree2".

Before the call of *tree2*, *tree1* parameters have the following values:

- *pa1.s* is "abc"
- *pa2* is "xyz"

In *tree2*, the values are changed accordingly:

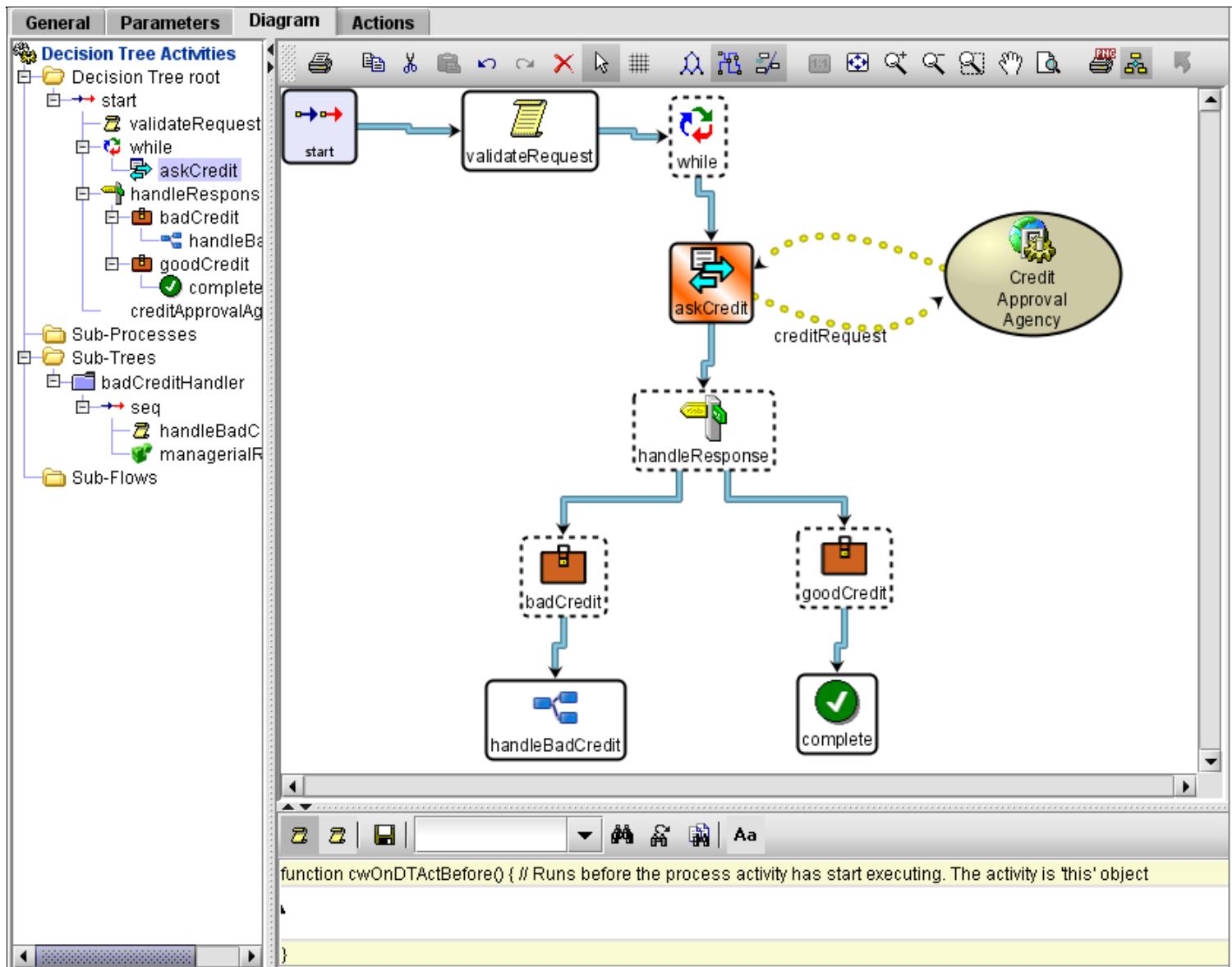
- *pb1.s* to "ddd"
- *pb2* to "qqq"

After the called tree *tree2* completes, the *tree1* parameters will be:

- *pa1.s* is "ddd"
- *pa2* - "xyz"

## Decision Tree Diagram

The **Diagram** tab of the Decision Tree presents the Decision Tree activities and their relationships. This interface is the same as in the [Diagram tab of Processes](#).



The tab is divided into these sections:

- Decision Tree Activities** tree: The left side area shows a hierarchical tree of Decision Tree, and sub-process and sub-tree activities.
- Graphic Diagram** pane: The right side top area contains a graphic diagram of the activities' relationships. Table below contains a description of the button functions.
- Script**: The right side bottom area contains a toolbar and a JavaScript editing field that allows script editing for the selected activity. The toolbar has a set of buttons on its left side that allows fast switching between the activity scripts. The Save button is used to save all changes. An editable combo box and a set of buttons on the right side can be used to search for particular words in the current activity or in all Decision Tree activities. The tooltips can be used to find out the button meaning.

The areas on the left and right sides of the **Diagram** tab are synchronized. If you click an item in the **Decision Tree Activities** tree, it is also selected in the **Graphic Diagram** pane, and vice versa.

The small triangular controls on the split bars between the sections can be used to hide some of the areas and to restore their previous size again.

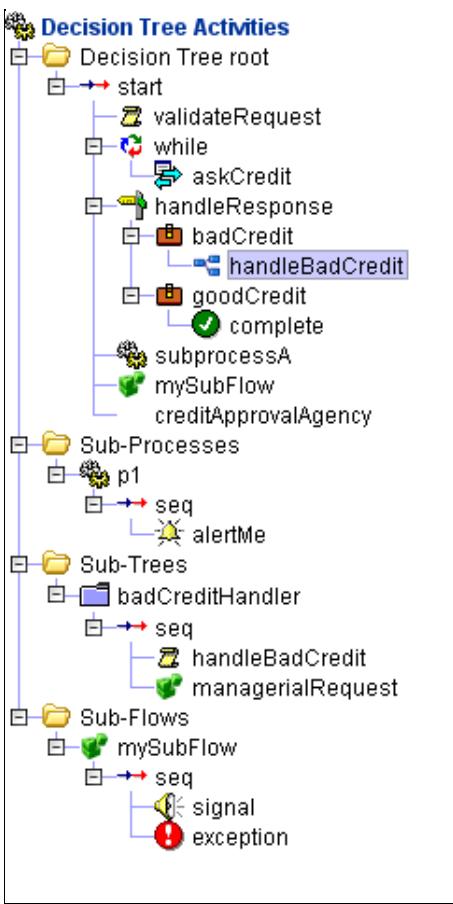
Button	Function
--------	----------

	Print the diagram.
	Copy selected activity.
	Cut selected activity.
	Paste selected activity.
	Undo action.
	Redo action.
	Delete selected activity.
	Select (shows the arrow cursor which should be used to select activities on the diagram).
	Grid (shows a grid on the diagram).
	Automatic node layout.
	Automatic link layout.
	Automatic labels layout.
	Show diagram in actual size.
	Fit the diagram into visible area.
	Zoom in.
	Zoom out.
	Zoom box (click the button then select an area on the diagram to zoom in).
	Pan (click the button, and then select a node on the diagram and drag it holding the left mouse button).
	Print Preview; a Preview pop-up box appears.
	Export as an image (exports a snapshot of the currently visible area into a JPG or PNG bitmap file which then can be imported into MS Word or any other application that supports that format).
	Flow direction (click the button, and then select an area on the diagram; the nodes and links included in the selection will be highlighted).
	Return to the previously viewed Decision Tree, sub-process or sub-tree.

The **Sub-Processes** folder of the **Decision Tree Activities** tree contains the sub-process activities as a separate tree. Click the sub-process item in the **Sub-Processes** folder to view the **sub-process** diagram in the **Graphic Diagram** pane. These sub-processes are read-only here; they can edit them by finding their metadata objects in **Processes** folder in Navigation Pane.

Similarly, The **Sub-Trees** folder of the **Decision Tree Activities** tree contains the Decision-tree activities (that is, calling another Decision Tree) as a separate tree. Click the sub-tree item in the **Sub-Trees** folder to view the **sub-tree** diagram in the **Graphic Diagram** pane. These sub-trees are read-only here; they can edit them by finding their metadata objects in **Decision Trees** folder in Navigation Pane.

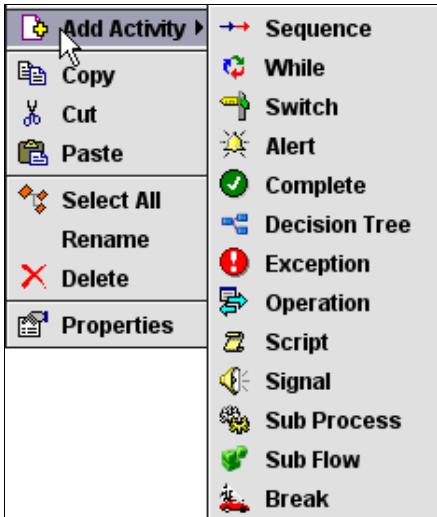
Lastly, the **Sub-Flows** folder of the **Decision Tree Activities** tree contains the sub-flow activities as a separate tree. You can add a sub-flow by right-clicking the **Sub-Flows** folder and then select **Add Subflow** at the pop-up menu. To view or edit existing sub-flows in this Decision Tree, simply click the sub-flow item in the **Sub-Flows** folder to view and edit the **sub-flow** diagram in the **Graphic Diagram** pane.



### Diagram Elements

To add the first Decision Tree activity node to the diagram, right-click the **Decision Tree root** folder. A right-click pop-up menu will appear where a Decision Tree activity can be selected from the **Add Activity** sub-menu. A dialog will open where the activity properties can be defined and an activity node will be added in the **Graphic Diagram** pane.

Once the top level Decision Tree activity node has been added to the diagram, additional child level Decision Tree activity nodes can be added by right-clicking the node in the **Decision Tree Activities** tree or in the **Graphic Diagram** pane. A pop-up menu will appear.

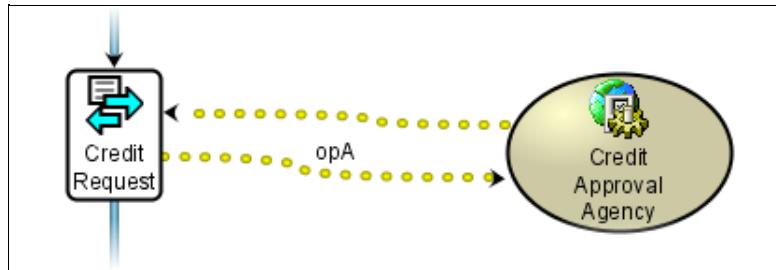


The set of options in the menu will vary depending on the actions available for selected activity node.

Menu Item	Description
<b>Add Activity</b>	Available for activities that have children activities. A list of allowed children activities is shown as a sub menu. The list of Activities that can be added can be found in section 4.18.4 <i>Decision Tree Activity Types and Execution</i> .
<b>Copy</b>	Copies the activity into the internal buffer.

<b>Cut</b>	Removes the activity from its current position in the tree after the paste operation.
<b>Paste</b>	Activated once <b>Copy</b> or <b>Cut</b> is selected.
<b>Select All</b>	Available if the activity has children activities. Highlights all children activities.
<b>Rename</b>	Renames the activity to another name.
<b>Delete</b>	Deletes the activity.
<b>Properties</b>	Used to view and edit the activity properties. When selected, a dialog opens with the activity properties (refer to section 4.18.5 <i>Decision Tree Activity Properties</i> ).
<b>Display</b>	Available for Decision Tree activity, Sub-Process and Sub-Flow activities only. Shows the activity as a separate diagram.
<b>Edit</b>	Available for Decision Tree, Sub-Process and Sub-Flow activities only. Will select the corresponding Decision Tree or Sub-Process in the <b>Navigation pane</b> of Velocity Studio, where the <b>Details pane</b> allows editing of the diagram.

As an Activity is added to the **Graphic Diagram** pane, an **Activity Node** is presented as an icon with an image corresponding to the Activity type and label. **Activity Links** are presented by solid lines ended with an arrow to link and indicate flow between activities. In addition, if an *Operation Activity* is added, a **Participant Node** is added in addition to the Operation Node. The Participation Node is presented as an oval icon with the participant name, with dotted yellow lines with arrow(s) indicating the message passing of the Operation between the Decision Tree runtime and Participant.



# Decision Tree Activities Types and Execution

## Common Attributes of Decision Tree Activities

In general, Decision Tree activities may have the following optional attributes defined. Overall, all activities (except *While*, *Break* and *Decision Tree* – which are not available in processes) behave similarly as in the business process, except that they cannot have schedule and delay. See section [Decision Tree Activity Types](#) below on properties of each specific activity type.

### General Tab

Field	Description
Name	Mandatory. Name of the Decision Tree Activity.
Label	Mandatory. Display label of the Decision Tree Activity.
Description	Description of the Decision Tree Activity for documentation.

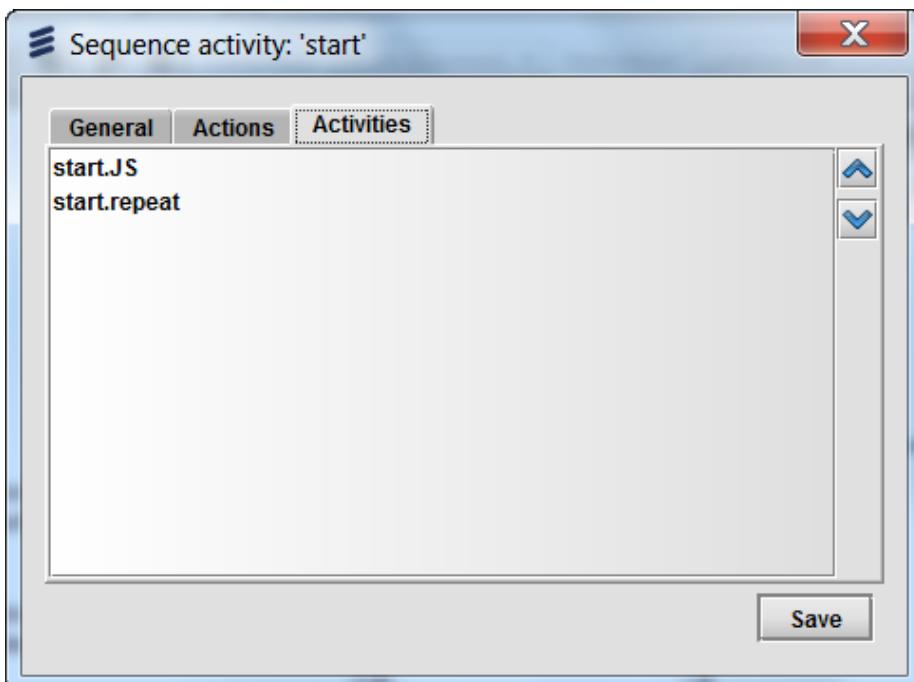
### Actions Tab

Field	Description
Before Script	A set of JavaScript statements that are executed before an activity-specific function is performed. For example, the <b>Before</b> script of an <i>operation</i> activity would usually initialize input message data (Document or Data Structure).
After Script	A set of JavaScript statements that are executed after an activity-specific function is performed. For <i>complex</i> activities, the script is executed after all sub-activities are completed. For example, the <b>After</b> script of the <i>request-response</i> Operation activity would usually process output message data (Document or Data Structure).
Condition Script	A Condition script can be used to skip an activity altogether.

### Activities Tab

If it is a *complex* activity with multiple sub-activities, the **Activities** tab appears and contains a **Children List** which lists all the sub-activities.

The sub-activities will be executed accordingly to their order in the **Children list**, from top to bottom. The arrow buttons located at the side of the list are used to move the selected element in the list to the desired location, thus changing the execution order of the selected activity.



## Decision Tree Activity Types

Decision Tree has a single start activity. However it may have many *Complete* activities as well as terminal activities - activities at the end of the group activities which have no more siblings. After performing the terminal activity the Decision Tree completes the same way as with *Complete* activity. This way the *Complete* activity is not mandatory but it makes the Decision Tree diagram more readable and is an easy way to implement "location" specific completion scripts (which depends on where in the Decision Tree diagram the terminal activity is located).

### Alert



An *atomic* activity that executes a specified Alert.

#### Type

- Simple

#### Definition

- Mandatory metadata in **General** tab: **Alert** object
- Mandatory **Before** script that returns an array in pairs: (processID, and the corresponding alert message). The alert is sent on behalf of the corresponding processes. For the alert message,
  - is the message content when it is for *email* and *pager* Alerts
  - is the alert description when it is for *worklist* Alerts
  - is output message initialization when it is for *external service* Alerts
- Optional **Condition** script. If false, the activity is skipped
- No **After** script
- Not defined in BPML specification

#### Execution

1. Runs the **Condition** script if any, and if the result is false skips the activity.
2. Runs the **Before** script and takes the result. If the result is null, empty array or empty string does nothing. If the result is an id (number) or array of ids sends the specified alert for behalf of the specified processes. If one or more processes are not found in the DB, throws exception.

### Break



Interrupts the loop of the **While** activity.

#### Type

- Simple

#### Definition

- Mandatory metadata in **General** tab: **While Loop** to interrupt. By default, this is the innermost **While** activity in which the **Break** activity resides. However, you can optionally specify another While activity in outer-loops, if any, which completes the specified While group.
- Optional **Condition** script. If false, the activity is skipped.
- No **Before** or **After** script

#### Execution

1. Runs the **Condition** script if any, and if the result is false skips the activity.
2. Runs the **After** script of the specified While group. Continues with the activity following the group.

### Case



Associates a condition with a *Switch* activity.

#### Type

- Simple

#### Definition

- Can contain zero or one sub-activity.
- Optional **Condition** script. If false the activity is skipped. Each Switch operator must have one case activity without conditional script (default Case operator).
- No **Before** script or **After** script allowed.

#### Execution

- Runs the **Condition** script and if the result is false, skips the activity. If true, runs the sub-activity if any.

#### Complete



An activity that completes the Decision Tree.

#### Type

- Simple

#### Definition

- Optional **Condition** script. If false, the activity is skipped.
- Optional **Before** script which runs before the Decision Tree is completed
- Can appear as the last activity in a sequence, or once in a *Switch* activity.
- The *complete* activity causes the Decision Tree to complete immediately. It does not cause children or parent Decision Trees to complete.
- No **After** script

#### Execution

- Runs the **Condition** script and if the result is false, skips the activity.
- Runs the **Before** script if any. Completes the tree. All activities that are queued for execution are ignored.

#### Decision Tree (Activity)



An activity that calls another Decision Tree. The *Decision Tree* activity may be used for breaking a complex logic-flow into separate components. A *Decision Tree* activity executes within the context of the current Decision Tree.

#### Type

- Special

#### Definition

- Mandatory metadata in **General** tab: **Decision Tree** to call
- Optional **Parameters Mapping** property, which is a list of pairs that define the parameter passing between the caller tree parameters and the called Decision Tree parameters.

Decision Tree activity: 'start.decisionTree'

Called Tree Parameters		Caller Tree Parameters	
order	object	Start	object
orderFullfilled	object	End	object

**Save**

The dialog lists all the parameters of the called Decision Tree. Click the corresponding **Caller Tree Parameters** to select the parameter upon which the **Called Tree Parameter** is to be mapped. Click the **Caller Tree Parameters** to view the caller Decision Tree parameters (or to edit, by double-clicking a parameter in the subsequent pop-up dialog box).

You are strongly encouraged to see [parameter passing in Decision Tree Parameters](#) for details on this topic.

- Optional **Before** script, which runs before the Decision Tree is called. When the script is invoked the new Decision Tree object is already created and its parameter list is populated, provided that the **Parameter Mapping** is specified. The Decision Tree system parameter *cwCalledTree* contains the called Decision Tree object.
- Optional **After** script, which runs after the called Decision Tree completes. The decision tree system parameter *cwCalledTree* contains the called Decision Tree object.
- Optional **Condition** script. If false, the activity is skipped.
- Not defined in BPML specification.

#### Execution

- Runs the **Condition** script if any, and if the result is false, skips the activity.
- Creates the called Decision Tree object and populates its parameter based on the specified **Parameters Mapping** (if any).
- Runs the **Before** script (if any).
- Runs the called Decision Tree.
- Runs the **After** script if any.

#### Exception (Fault)



An activity causes an exception to be sent to a process.

#### Type

- Simple

#### Definition

- Mandatory metadata in **General** tab: **Exception** object
- Mandatory **Before** script that returns the ID of one or several processes, as array. The exception is sent to the corresponding processes.
- Optional **Condition** script. If false, the activity is skipped.
- No **After** script

#### Execution

- Runs the **Condition** script, if any, and if the result is false skips the activity.
- Runs the **Before** script and takes the result. If the result is null, empty array or empty string, nothing is sent.
- If the result is an id (number) or array of ids, the specified exception is sent to the processes. If one or more processes are not found in the

DB, an exception is thrown.

## Operation



An activity that performs a message exchange with a Participant.

### Type

- Simple

### Definition

- Mandatory metadata in **General** tab: **Participant** and **Operation** objects. The operation must be of type *One-Way* or *Request-Response*.
- Optional **Before** script which runs before the operation is called. The Decision Tree system parameter *cwOutputMessage* contains the outgoing message for the operation.
- Optional **After** script which runs after the operation returns, and can be specified only for *Request-Response* operations. The Decision Tree system parameters *cwInputMessage* and *cwErrorMessage* contain the operation result and error message respectively.
- Optional **Condition** script. If false, the activity is skipped.

### Execution

- Runs the **Condition** script, if any, and if the result is false skips the activity.
- Creates the operation input data object and calls the **Before** script (if any) passing it as parameter.
- Performs the operation.
- Calls the **After** script, if any, and sets the output message and error message (if any) as parameters.

## Script



An activity that executes a **Before** script.

### Type

- Simple

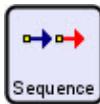
### Definition

- Mandatory **Before** script.
- Optional **Condition** script. If false, the activity is skipped.
- No **After** script

### Execution

- Runs the **Condition** script, if any, and if the result is false skips the activity.
- Runs the **Before** script.

## Sequence



An activity that completes after all its sub-activities have completed. All sub-activities are executed in sequence.

### Type

- Complex

### Definition

- Optional **Before** script which runs before the content of the group is executed.
- Optional **After** script which runs after the content is executed.
- Optional **Condition** script. If false, the activity is skipped.

### Execution

- Runs the **Condition** script, if any, and if the result is false skips the activity.
- Runs the **Before** script (if any).
- When (and if) all activities in the Sequence group are executed, runs the **After** script (if any).

## Signal



An activity causes a Signal to be sent to another process.

### Type

- Simple

### Definition

- Mandatory metadata in **General** tab: **Signal** object
- Mandatory **Before** script that returns the ID of one or several processes (as array). The signal is sent to the corresponding processes.
- Optional **Condition** script. If false, the activity is skipped.
- No **After** script.

### Execution

- Runs the **Condition** script, if any, and if the result is false skips the activity.
- Runs the **Before** script and takes the result. If the result is null, empty array or empty string, nothing is sent.
- If the result is an id (number) or array of ids sends the specified signal to the processes. If one or more processes are not found in the DB, an exception is thrown.

## Sub-flow



An activity that calls a Decision Tree subflow (akin to a procedure call). The *Sub-flow* activity is used for breaking a complex Decision

Tree into separate components, but in contrast to a [Decision Tree activity](#), the activities in *Sub-flow* become part of the parent Decision Tree metadata with access to the same parameters, instead of a separate Decision Tree metadata. In both cases, the execution is synchronous (in the same thread).

### Type

- Special

### Definition

- Mandatory metadata in **General** tab: **Sub-Flow** to call
- Optional **Before** script, which runs before the Sub-Flow is called.
- Optional **After** script, which runs after the called Sub-Flow completes.
- Optional **Condition** script. If false, the activity is skipped.
- Not defined in BPML specification.

### Execution

- Runs the **Condition** script if any, and if the result is false, skips the activity.
- Runs the **Before** script (if any).
- Runs the called Sub-Flow.
- Runs the **After** script if any.

## Sub-process (Spawn)



An activity that spawns a child process. It instantiates a new instance of the sub-process.

### Type

## Simple

### Definition

- Mandatory metadata in **General** tab: **Sub-Process** object.
- Optional **Before** script. The Decision Tree system parameter *cwProcessDoc* contains the process document of the specified process.
- Optional **Condition** script. If false, the activity is skipped.
- No **After** script.

### Execution

- Runs the **Condition** script, if any, and if the result is false skips the activity.
- Creates and initializes a process document for the specified process.
- Calls the **Before** script (if any) and passes the product document as parameter. If the script returns a value it is considered ID of the parent process. If the parent process is not found in the database, an exception is thrown.
- Creates the asynchronous process in the database to be picked up by Process Engine.

## Switch



An activity which completes after zero or more of its sub-activities have completed, subject to one or more conditions. This models branching in programming languages.

### Type

- Complex

### Definition

- Optional **Condition** script. If false, the activity is skipped.
- Can only have **Case** sub-activities.
- Can only have one default **Case** activity (a case with no condition).
- Can execute more than one **Case** activity (not exclusive). When *Switch* activities are marked as none-exclusive, all **Case** activities that have a condition evaluated to true execute the same way.

### Execution

- Runs the **Condition** script, if any, and if the result is false skips the activity.
- Checks the conditions of all case statements and run the first that returns true. If no one returns true, runs the default case activity.

## While



A complex activity that establishes a loop in its sub-activities. The **While** group performs its sub-activities only if the **While** condition script returns true. The **While** activity completes if the script returns false.

### Type

- Complex

### Definition

- Mandatory **Condition** script. If false, the activity completes.
- Optional **Before** script which runs before the first iteration (if any).
- Optional **After** script which runs after the activity completes, but only if the loop has been executed at least one time (that is, the loop has not been skipped altogether)

### Execution

- Runs the **Condition** script. If the result is false, the activity and its sub-activities is skipped.
- If it returns true, runs the **Before** script (if any).
- Runs all sub-activities in sequence.
- After the last activity is completed, runs again the **Condition** script.
- If the result is true, runs all sub-activities in sequence.

- The iteration continues until **Condition** script returns false.
- Runs the **After** script (if any), if at least one iteration has been performed.
  - Keep in mind that if the While group is interrupted by a **Complete** or **Break** operation, the **After** script is not performed.

## Decision Tree Actions

The **Actions** tab centralizes Decision Tree actions, which has *Initialization script* and *Completion script*.

The screenshot shows the Velocity Studio interface with the 'Actions' tab selected. On the left, there's a tree view with 'creditAssure' expanded, showing 'cwOnDTInit' and 'cwOnCompletion'. The main panel has three fields: 'Name:' with 'cwOnDTInit', 'Description:' (empty), and 'Script:' containing the code 'function cwOnDTInit() { // Runs before first operation of the decision tree'.

The **Actions** tab follows Velocity Studio's convention of a [Scripting Interface](#), but with no parameters and no return objects.

Notably, the *Decision Tree* object is accessed by using *this* keyword in script.

**Note:** Refer to the [JavaScript Extensions HTML documentation: Decision Tree object](#) for reference.

There are two system-defined scripts for Decision Tree -- initialization script and completion script. The Decision Tree always performs its initialization script (if any) at the moment it is run (see section [Running Decision Tree](#)). After a terminal or "Complete" activity is performed, the Decision Tree runs its completion script (if any).

Action	Script Name	Description
Initialization	<i>cwOnDTInit</i>	It is run before the first activity is processed.
Completion	<i>cwOnCompletion</i>	It is run after a terminal activity or Complete activity (see below) is performed.

## Running Decision Trees

The runtime Decision Tree object is an interpreter of the Decision Tree diagram that runs the activities of the tree and implements its logic.

To run a Decision Tree, first create a Decision Tree runtime object using the constructor `new DecisionTree(<metadata Decision Tree name>)` and then execute the method `run()`. After the creation of the object but before executing the run method, the tree parameters could be populated if needed.

Example: Run a Decision Tree of type `ns:dtree` with parameters `data` (Document), `number` (integer) and `result` (Document). The Decision Tree creates and populates the parameter `result`.

```
var dt = new DecisionTree("ns:dtree"); // Create the object
dt.number = 5; // Set the numeric parameter
dt.data = doc; // Set the document parameter
dt.run(); // Execute the Decision Tree
// use the document dt.result
var v = dt.result.leaf1 ...
...
```

Alternative way of running the Decision Tree is to specify the parameters in the run statement. In this case the order of parameter should match their order in the parameter list. The example above can be written as:

```
var dt = new DecisionTree("ns:dtree");
dt.run(doc, 5);
// use the document dt.result
var v = dt.result.leaf1 ...
```

If the Decision Tree uses empty result document instead of creating it, the Decision Tree can be run as single expression. This expression (similar to a constructor) returns the Decision Tree object:

```
ns.dtree.DecisionTree(doc, 5, result).run();
// use the document result
var v = result.leaf1 ...
```

## Open ID

---

The recommended identity management server to use with OpenID is Atlassian Crowd (<http://www.atlassian.com/software/crowd/>). This server supports the following directories:

- Microsoft Active Directory
- OpenLDAP
- Sun
- Novell eDirectory
- Apache Directory Server

For a full listing of directories that Atlassian Crowd supports, see  
<http://confluence.atlassian.com/display/CROWD/Supported+Applications+and+Directories>.

Once you have logged in to the OpenID server, your login can be shared with several OpenID-enabled applications in the same Web browser. OpenID is a flexible solution and a de facto standard for integrating disparate applications into portals in the cloud. Applications can be built using different technologies as OpenID support is not limited to Java™ application servers.

For more information on configuring OpenID, see [Appendix A: OpenID Authentication](#).

## Kerberos

---

Once you have logged in to your OS account, your Web browser automatically considers you as being logged in to the application. This single sign-on solution is more transparent, but is less flexible as its setup is application server-dependent.

The following sections describe special browser configuration settings for single sign-on using Kerberos in Internet Explorer:

- [Configure Microsoft Clients to Use Windows Integrated Authentication](#)
- [Configure WebSphere 7.x Kerberos and Active Directory Single Sign-on Authentication](#)
- [Configure Weblogic 10.3 Kerberos and Active Directory Single Sign-on Authentication](#)
- [Configure JBoss 5.1 Kerberos and Active Directory Single Sign-on Authentication](#)

# Configure Microsoft Clients to Use Windows Integrated Authentication

---

This procedure contains the following tasks:

- [Configure Internet Explorer to use Windows authentication](#)
- [Configure intranet authentication](#)
- [Verify the proxy settings](#)

## Configure Internet Explorer to use Windows Authentication

To configure your Internet Explorer browser to use Windows authentication, follow these procedures:

1. Open Internet Explorer, and click **Tools > Internet Options** from the menu bar.
2. Click the **Security** tab.
3. Click the **Local intranet** icon, and then click the **Sites** button.
4. In the Local intranet dialog, ensure that you have checked the following settings:
  - **Include all local (intranet) sites not listed in other zones**
  - **Include all sites that bypass the proxy server**
5. Proceed to click the **Advanced** button.
6. In the Local intranet dialog, add all relative domain names that will be used for WebLogic Server instances participating in the single sign-on configuration (for example, *myhost.example.com*), and then click the **OK** button.

## Configure Intranet Authentication

To configure your Internet Explorer browser for intranet authentication, complete these steps:

1. In Internet Explorer, click **Tools > Internet Options** from the menu bar.
2. Click the **Security** tab.
3. Click the **Local intranet** icon, and then click the **Custom level...** button.
4. In the **Security Settings** dialog, scroll to the **User Authentication section** and select **Automatic logon only in Intranet zone**. This option prevents you from having to re-enter logon credentials.
5. Click the **OK** button to save your changes.

## Verify the Proxy Settings

If you have a proxy server enabled, do the following:

1. In Internet Explorer, click **Tools > Internet Options** from the menu bar.
2. Click the **Connections** tab, and then click the **LAN settings** button.
3. Verify that the proxy server **Address** and **Port** number fields are correct, and then click the **Advanced** button.
4. In the **Proxy Settings** dialog box, ensure that all desired domain names are entered in the **Exceptions** field.
5. Click the **OK** button to close the **Proxy Settings** dialog and save your changes.

# Configure WebSphere 7.x Kerberos and Active Directory Single Sign-on Authentication

Before proceeding with configuring WebSphere 7.x Kerberos and Active Directory for single sign-on authentication, add the following parameters to your WebSphere JVM runtime environment to help debug:

```
-Dcom.ibm.security.jgss.debug=all  
-Dcom.ibm.security.krb5.Krb5Debug=all
```

**Note:** After you are done with the procedure that follows, remove these parameters.

This procedure consists of four steps:

1. [Create a Kerberos identification](#)
2. [Configure the Kerberos configuration file](#)
3. [Configure Microsoft Active Directory as a WebSphere user repository](#)
4. [Configure SPNEGO Web authentication in WebSphere](#)

## Step 1: Create a Kerberos identification

To create a Kerberos identification, complete these steps:

1. In Active Directory, right-click the **Users** folder in the left navigation menu, and click **New > User**. Create a user XXX in Active Directory as guided by the wizard.
2. Open the newly created user account and click the **Account** tab. For the **Account options** field, scroll down and select **Use DES encryption types for this account**.
3. Click the **Apply**, and then **OK** buttons to save your changes.
4. Proceed to reset the password for this user by right-clicking the user account and selecting **Reset Password**.
5. Set the **Service Principal Name** by entering the following at a command prompt:

```
setspn -a HTTP/ XXX.CONCEPTWAVE.COM XXX
```

6. Test the service principal name by entering the following command:

```
setspn -L XXX
```

7. Generate a key tab by entering the following command:

```
ktpass -out c:\temp\ws.keytab -princ HTTP/ XXX.conceptwave.com@CONCEPTWAVE.COM  
-mapUser XXX -mapOp set -pass XXX -crypto DES-CBC-MD5 -pType KRB5_NT_PRINCIPAL +DesOnly
```

8. Test the keytab file by using this command:

```
klist -k XXX.keytab
```

## Verify Kerberos Environment

To verify that the Kerberos environment is set up correctly, perform these steps:

**Note:** JAVA\_HOME in the following commands indicate your IBM JDK home.

1. Verify that the Kerberos keytab file is valid with the correct encryption by running the following command:

```
JAVA_HOME/jre/bin/java com.ibm.security.krb5.internal.tools.Klist -e -k <keytab file>
```

2. Verify that the Kerberos configuration file (krb5.ini for Windows and krb5.conf for other platforms) points to the keytab by running the following command:

```
JAVA_HOME/jre/bin/java -Djava.security.krb5.conf=<krb5 file> com.ibm.security.krb5.internal.tools.Ktab
```

3. Verify that the Kerberos configuration file points correctly to the Key Distribution Centre (KDC) by using this command:

```
JAVA_HOME/jre/bin/java  
-Djava.security.krb5.conf=<krb5 file> com.ibm.security.krb5.internal.tools.Kinit <User or SPN>
```

4. If only one key is inside the keytab, verify that the Kerberos configuration file and Kerberos keytab are valid using the following command:

```
JAVA_HOME/jre/bin/java  
-Djava.security.krb5.conf=<krb5 file> com.ibm.security.krb5.internal.tools.Kinit -k
```

## Step 2: Configure the Kerberos Configuration File

To create a Kerberos configuration file for WebSphere's use, follow these steps:

1. Copy the keytab file to the WebSphere Application Server system. In this scenario, the file is copied to *c:\windows*.
2. Create the associated Kerberos configuration file by selecting one of the following methods:
  - a. Use wsadmin by entering the following command:

```
wsadmin>$AdminTask createKrbConfigFile {-krbPath c:/WINDOWS krb5.ini  
-realm CONCEPTWAVE.COM -kdcHost carta.conceptwave.com -dns conceptwave.com  
-keytabPath c:/WINDOWS/ws.keytab}
```

The configuration file located in *c:/WINDOWS/krb5.ini*.

- b. Manually create a *krb5.ini* file. The following is a sample file:

```
[libdefaults]  
default_realm = MYDOM.COM //Identifies the default realm. Set its value to your Kerberos realm.  
default_tkt_enctypes = des-cbc-crc  
default_tgs_enctypes = des-cbc-crc  
ticket_lifetime = 600  
  
[realms]  
MYDOM.COM = {  
    kdc = <IP address for MachineA> //Host running the KDC  
    //For Unix systems, you need to specify port 88, as in <IP-address>:88  
    admin_server = MachineA  
    default_domain = MYDOM.COM  
}  
  
[domain_realm]  
.mydom.com = MYDOM.COM  
  
[appdefaults]  
autologin = true  
forward = true  
forwardable = true  
encrypt = true
```

## Step 3: Configure Microsoft Active Directory as a WebSphere User Repository

To configure Microsoft Active Directory as a WebSphere user repository, follow these steps:

1. Log on to the WebSphere Application Server administrative console and navigate to **Security > Global security**.
2. In the **Authentication** section, select the **Kerberos and LTPA** option.
3. In the **Web and SIP security** section, click **Single sign-on (SSO)**. Select the **Enabled** and **Web inbound security attribute propagation** options. In the **Domain name** field, set it to **CONCEPTWAVE.COM**.
4. Click **External authorization providers**. Proceed to select **Built-in authorization**, and then click the **OK** button.
5. In the **User account repository** section, click the **Available realm definitions** field, select **Standalone LDAP registry**, and then click the **Configure** button. Configure the Microsoft Active Directory as the user registry with the following **General Properties**:
  - o **Primary administrative user name** - Enter your user ID (for example, **build**)
  - o **Server user identity** - Select the **Automatically generated server identity** option
  - o **Type of LDAP server** - Select **Microsoft Active Directory**
  - o **Host** - Name of the LDAP server (for example, **carta.conceptwave.com**)
  - o **Port** - Port of the LDAP server (for example, **389**)
  - o **Base distinguished name (DN)** - The base distinguished name of the directory service
  - o **Bind distinguished name (DN)** - Enter the distinguished name for a user ID that is authorized to run queries on your server
  - o **Bind password** - Enter the distinguished password for a user ID that is authorized to run queries on your server
  - o **Search timeout** - Set to **120** seconds
  - o **Reuse connection** - Select this option

- o **Ignore case for authorization** - Select this option

Click the **Apply** button to save your changes.

**Note:** If you have selected the **SSL enabled** option, you must set up your SSL configuration.

6. In the **Configuration** section, click the **Test connection** button. This test must be successful before you continue with this procedure.
7. Return to the **Global security** panel. In the **Available realms** drop-down menu, select **Standalone LDAP registry**, and then click **Set as current**.
8. In the **Administrative security** section, verify that you have selected the **Enable administrative security** option.
9. In the **Application security** section, select the **Enable application security** option.
10. Click the **Apply** button to apply all settings in the **Global security** panel.
11. In the **Administrative security** section, click **Administrative user roles** and assign the **Administrative user** role to at least one user (for example, an LDAP - AD repository user such as **CN=build,CN=Users,DC=conceptwave,DC=com**) in the Microsoft Active Directory.
12. Save the configuration and restart WebSphere Application Server.

#### Step 4: Configure SPNEGO Web authentication in WebSphere

To configure the Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) Web authentication in WebSphere Application Server using the administrative console, follow these steps:

1. Navigate to **Security > Global security**, and then expand **Web and SIP security**.
2. Click **General Properties**, select **Authenticate when any URI is accessed** for **Web authentication behavior**, and then click **SPNEGO Web authentication**.
3. You need to define a filter for each application that will participate in single sign-on. From the **SPNEGO Filters** list, click the **New** button. In the **General Properties** panel, complete the following steps:
  - a. Enter the fully qualified host name of the WebSphere Application Server in the **Host name** field and enter the realm name in the **Kerberos realm name** field (for example, **CONCEPTWAVE.COM**).
  - b. In the **Filter Criteria** field, enter a value for **request-url%** that equals the name of the Web application or the prefix for the Web application (for example, **request-url%=cwf**), so that requests can be verified by SPNEGO Web authentication.
  - c. Select **Trim Kerberos realm from principal name** to remove the suffix of the principal user name.
  - d. Click the **Apply** button.
4. On the **Global security panel**, in the **Authentication** section, click **Kerberos configuration**. Then, in the **Kerberos Authentication Mechanism** section, complete the following steps:
  - a. Change the **Kerberos service name** to **HTTP**, which depends on how you create the Kerberos identification.
  - b. Enter **CONCEPTWAVE** as the **Kerberos realm name**.
  - c. Select **Trim Kerberos realm from principal name**, and then clear **Enable delegation of Kerberos credentials**.
  - d. Click the **Apply** button.

**Note:** If you are opening an order directly using parameters in the Web address, when you log out and try opening the same Web address again, the page will not load. The issue is that when you log out, [http://host:port/cwf/v/cwf\\_oe.logout.UserInterface.js](http://host:port/cwf/v/cwf_oe.logout.UserInterface.js) attempts to perform a single sign-on again. To resolve this issue, in **Global security > SPNEGO Web authentication**, in the **Filter Criteria** field for **request-url%=cwf**, add **request-url!=cwf/v/cwf\_oe.logout.UserInterface.js** as another filter.

# Configure Weblogic 10.3 Kerberos and Active Directory Single Sign-on (SSO) Authentication

This procedure consists the following steps:

1. [Create a Kerberos identification for WebLogic Server](#)
2. [Configure your network domain to use Kerberos](#)
3. [Create a JAAS login file](#)
4. [Configure Weblogic Startup Script for Kerberos Authentication](#)
5. [Configure Microsoft Clients to Use Windows Integrated Authentication](#)
6. [Configure Security Providers](#)
7. [Configure Users and LDAP Security Provider in Administration Application](#)
8. [Modify Your Application's web.xml and weblogic.xml](#)

## Step 1: Create a Kerberos Identification for WebLogic Server

### Notes:

- Creating a Kerberos identification is environment-related. For the AVM environment, see [Create Kerberos in an AVM environment](#).
- The following steps are from [Oracle's](#) Web site.

To create a Kerberos identification for WebLogic Server, do the following:

1. In the Active Directory server, create a user account for the host computer on which WebLogic Server runs by clicking **New > User**. When creating the user account, use the simple name of the computer. For example, if the host is named *myhost.example.com*, create a user in Active Directory called *myhost*.  
  
Note the password you defined when creating the user account. You will need it in step 3. Do not select the **User must change password at next logon** option, or any other password options.
2. Configure the new user account to comply with the Kerberos protocol. The user account's encryption type must be DES and the account must require Kerberos pre-authentication.
  - a. Right-click the name of the user account in the **Users** tree in the left pane and select **Properties**.
  - b. Click the **Account** tab and check the **Use DES encryption types for this account** checkbox. Make sure that no other boxes are checked, particularly the **Do not require Kerberos pre-authentication** option.
  - c. Setting the encryption type may corrupt the password. Therefore, reset the user password by right-clicking the name of the user account, selecting **Reset Password**, and then re-entering the same password specified earlier.
3. Use the setspn utility to create the Service Principal Names (SPNs) for the user account created in step 1. Enter the following commands:

```
setspn -a host/myhost.example.com myhost
setspn -a HTTP/myhost.example.com myhost
```

4. Check which SPNs are associated with your user account, using the following command:

```
setspn -L account name
```

**Note:** If the same service is linked to a different account in the Active Directory server, the client will not send a Kerberos ticket to the server.

5. Create a user mapping using the ktpass utility. In Windows, run the following command:

```
ktpass -princ host/myhost@Example.CORP -pass password -mapuser myhost -out c:\temp\myhost.host.keytab
```

6. Create a keytab file. On Windows, the ktab utility manages principal name and key pairs in the key table, and allows you to list, add, update, or delete principal names and key pairs. On Unix, it is preferable to use the ktpass utility.

For Windows, follow these steps:

- a. Run the ktab utility on the host on which WebLogic Server is running to create the keytab file:

```
ktab -k keytab-filename -a myhost@Example.CORP
```

- b. Copy the keytab file to the startup directory in the WebLogic Server domain.

For Unix, complete these steps:

- a. Create a user mapping using the ktpass utility, using a command such as the following, where *password* is the password for the user account created in step 1:

```
ktpass -princ HTTP/myhost@Example.CORP -pass password -mapuser myhost -out c:\temp\myhost.HTTP.keytab
```

- b. Copy the keytab file created in Step a to the startup directory in the WebLogic Server domain.
- c. Log in as root and then merge them into a single keytab using the ktutil utility as follows:

```
ktutil: "rkt myhost.host.keytab"
ktutil: "rkt myhost.HTTP.keytab"
ktutil: "wkt mykeytab"
ktutil: "q"
```

7. Run the kinit utility to verify that Kerberos authentication is working properly:

```
kinit -k -t keytab-file account-name
```

The output should be something similar to the following:

```
New ticket is stored in cache file C:\Documents and Settings\Username\krb5cc_MachineB
```

### Create Kerberos in an AVM environment

To create Kerberos in your AVM environment, complete these steps:

1. In Active Directory, right-click the **Users** folder in the left navigation menu, and click **New > User**. Create a user XXX in Active Directory as guided by the wizard.
2. Open the newly created user account and click the **Account** tab. For the **Account options** field, scroll down and select **Use DES encryption types for this account**.
3. Click the **Apply**, and then **OK** buttons to save your changes.
4. Proceed to reset the password for this user. Right-click the user account and select **Reset Password**.
5. Set the **Service Principal Name** by entering the following at a command prompt:

```
setspn -a HTTP/ XXX.CONCEPTWAVE.COM XXX
```

6. Test the service principal name by entering the following command:

```
setspn -L XXX
```

7. Generate a key tab by running one of the following commands:
  - o ktab:

```
ktab -k XXX.keytab a XXX@ CONCEPTWAVE.COM
```

- o ktpass:

```
ktpass -out c:\temp\ws.keytab -princ HTTP/ XXX.conceptwave.com@CONCEPTWAVE.COM
-mapUser XXX -mapOp set -pass XXX -crypto DES-CBC-MD5 -pType KRB5_NT_PRINCIPAL +DesOnly
```

8. Test the keytab file by using this command:

```
klist -k XXX.keytab
```

9. Copy the generated keytab file, XXX.keytab, to the domain directory of WebLogic
10. Place the krb5.ini file in c:\windows.

### Step 2: Configure Your Network Domain to Use Kerberos

A Windows domain controller can serve as the Kerberos Key Distribution Centre (KDC), using the Active Directory and the Kerberos services. On any domain controller, the Active Directory and the Kerberos services are running automatically.

Java™ GSS requires a Kerberos configuration file. The default name and location of the Kerberos configuration file depends on the operating system being

used. Java GSS uses the following order to search for the default configuration file:

1. The file referenced by the java.security.krb5.conf Java property.
2. \${java.home}/lib/security/krb5.conf.
3. %windir%\krb5.ini on Microsoft Windows platforms.
4. /etc/krb5/krb5.conf on Solaris platforms.
5. /etc/krb5.conf on other Unix platforms.

To configure Kerberos in your Windows domain controller, you need to configure each workstation that will access the KDC to locate the Kerberos realm and available KDC servers. The following is a sample krb5.ini file:

```
[libdefaults]
default_realm = MYDOM.COM //Identifies the default realm. Set its value to your Kerberos realm.
default_tkt_enctypes = des-cbc-crc
default_tgs_enctypes = des-cbc-crc
ticket_lifetime = 600

[realms]
MYDOM.COM = {
kdc = <IP address for MachineA> //Host running the KDC
//For Unix systems, you need to specify port 88, as in <IP-address>:88
admin_server = MachineA
default_domain = MYDOM.COM
}

[domain_realm]
.mydom.com = MYDOM.COM

[appdefaults]
autologin = true
forward = true
forwardable = true
encrypt = true
```

### Step 3: Create a JAAS Login File

Create a JAAS login file for Kerberos authentication (krb5Login.conf), and put this file in the WebLogic domain folder.

```
com.sun.security.jgss.krb5.initiate {
    com.sun.security.auth.module.Krb5LoginModule required
    principal="myhost@Example.CORP" useKeyTab=true
    keyTab=mykeytab storeKey=true;
};

com.sun.security.jgss.krb5.accept {
    com.sun.security.auth.module.Krb5LoginModule required
    principal="myhost@Example.CORP" useKeyTab=true
    keyTab=mykeytab storeKey=true;
};
```

### Step 4: Configure Weblogic Startup Script for Kerberos Authentication

Add the following system parameters into your WebLogic startup command:

```
java.security.krb5.realm=domain name
java.security.krb5.kdc=AdHostName
javax.security.auth.useSubjectCredsOnly=false
WebLogic.security.enableNegotiate=true
java.security.auth.login.config=krb5Login.conf (Note: This is the JAAS file created in step 3 above.)
java.security.krb5.conf=krb5.conf (Note: This is the configuration file created in step 3 above. This file name is for UNIX environment; for windows, the file will be named krb5.ini and can be placed directly under
C:\WinNT\)
```

For debug purpose, you can add the following system parameters into your WebLogic startup command:

```
sun.security.krb5.debug=true
sun.security.jgss.debug=true
DebugSecurityAdjudicator=true
WebLogic.StdoutDebugEnabled=true
WebLogic.Debug.DebugSecurityAtz=true
WebLogic.Debug.DebugSecurityAtn=true
WebLogic.log.StdoutSeverity=Debug
WebLogic.StdoutSeverityLevel=64
```

## Step 5: Configure Microsoft Clients to Use Windows Integrated Authentication

Configure your Microsoft clients to use Windows Integrated Authentication by following the steps provided in [Configure Microsoft Clients to Use Windows Integrated Authentication](#).

## Step 6: Configure Security Providers

Configure the security providers from the WebLogic console. Login to WebLogic console and go to **Home > Summary of Security Realms > myrealm > Providers**.

1. Select WebLogic's existing default authenticator **DefaultAuthenticator** and change control flag of DefaultAuthenticator to **SUFFICIENT**.
2. Follow these steps to create Negotiate Identity Assertion provider:
  - a. Navigate to **Home > Summary of Security Realms > myrealm > Providers > New**.
  - b. Enter **NegotiateIdentityAssertion** as the name of the new provider.
  - c. Select **NegotiateIdentityAssertion** from **Type** drop-down list.
  - d. Click the **OK** button to save your changes.

Under **Provider Specific**, uncheck **Form Based Negotiation Enabled**.

3. Follow these steps to create Active Directory Authentication provider:
  - a. Navigate to **Home > Summary of Security Realms > myrealm > Providers > New**.
  - b. Enter **ActiveDirectoryAuthenticator** as the name of the new provider.
  - c. Select **ActiveDirectoryAuthenticator** from **Type** drop-down list.
  - d. Click the **OK** button to save your changes.

Under **Provider Specific**, specify your AD server's setup. Below is the configuration set and can be seen under **/wldomain/config/config.xml**.

```
<sec:authentication-provider xsi:type="wls:active-directory-authenticatorType">
    <sec:name>ActiveDirectoryAuthenticator</sec:name>
    <sec:control-flag>OPTIONAL</sec:control-flag>
    <wls:propagate-cause-for-login-exception>false</wls:propagate-cause-for-login-exception>
    <wls:host>xxx.conceptwave.com</wls:host>
    <wls:user-object-class>*</wls:user-object-class>
    <wls:user-name-attribute>sAMAccountName</wls:user-name-attribute>
    <wls:principal>zzz</wls:principal>
    <wls:user-base-dn>DC=conceptwave,DC=com</wls:user-base-dn>
    <wls:credential-encrypted>{3DES}BYgR6W3RUTCXJ9Krwu0zA==</wls:credential-encrypted>
    <wls:user-from-name-filter>(&#38;(sAMAccountName=%u)(objectclass=*))</wls:user-from-name-filter>
    <wls:group-base-dn>DC=conceptwave,DC=com</wls:group-base-dn>
    <wls:group-from-name-filter>(&#38;(cn=%g))</wls:group-from-name-filter>
    <wls:static-group-name-attribute></wls:static-group-name-attribute>
    <wls:static-member-dn-attribute>memberOf</wls:static-member-dn-attribute>
    <wls:static-group-dns-from-member-dn-filter>(&#38;(objectCategory=group))</wls:static-group-dns-
from-member-dn-filter>
</sec:authentication-provider>
```

4. Activate the changes and restart the server.

## Step 7: Configure Users and LDAP Security Provider in Administration Application

To configure users and LDAP security provider in Administration application, follow these steps:

1. In the runtime environment select Administration application and navigate to **Providers > Provider** menu.
2. The **Provider** dialog appears; select **LDAP** from the drop-down list of **Security Provider** field.
3. In the **Data** field enter the following value:

```
ldap://<yourdomaincontroller>:389;{user}@conceptwave.com
```

4. Click the **Save** button to save your changes.
5. [Create the users](#) you want to authenticate with active directory. These are the users which exist in the active directory using their sAMAccountName, such as their NTID (network ID) and assign their appropriate groups.

**Note:** The users settings should be already done in real production environment in case application is currently running without SSO also. But for testing purpose you need to create few users you want to test with.

## Step 8: Modify Your Application's web.xml and weblogic.xml

Stop the already running server and modify web.xml and WebLogic.xml.

Add the following to the web.xml:

```
<security-constraint>
```

```
<display-name>Security Constraint for SSO </display-name>
<web-resource-collection>
    <web-resource-name>My webapp</web-resource-name>
    <description>Group of Users</description>
    <url-pattern>/*</url-pattern>
</web-resource-collection>
<auth-constraint>
    <role-name>SSORole</role-name>
</auth-constraint>
</security-constraint>
<login-config>
    <auth-method>CLIENT-CERT</auth-method>
</login-config>
<security-role>
    <description>Role description</description>
    <role-name>SSORole</role-name>
</security-role>
```

Add following into the WebLogic.xml:

```
<security-role-assignment>
    <role-name>SSORole</role-name>
    <principal-name>everyone</principal-name>
</security-role-assignment>
```

After making all the above changes and deployment of war file with above web.xml and weblogic.xml restart the application server. You should be able to login to application as SSO, such as without login.

# Configure JBoss 5.1 Kerberos and Active Directory Single Sign-on Authentication

Before proceeding with configuring JBoss 5.1 Kerberos and Active Directory for single sign-on authentication, you can use the following parameters to help debug:

```
-Dsun.security.krb5.debug=true  
-Dsun.security.spnego.debug=true
```

**Note:** After you have performed the procedure that follows, remove these parameters.

This procedure consists of six steps:

1. [Change web.xml inside your cwf.war](#)
2. [Create your krb5.conf file and put it into JBoss bin folder](#)
3. [Change your login-config.xml file](#)
4. [Register a Service Principal Name](#)
5. [Download spnego.jar and place it into the JBoss library](#)
6. [Configure Web browser settings](#)

## Step 1: Change web.xml inside your cwf.war

Add the following XML to web.xml. You can append it after <display-name>ConceptWave Framework</display-name>.

```
<filter>  
    <filter-name>SpnegoHttpFilter</filter-name>  
    <filter-class>net.sourceforge.spnego.SpnegoHttpFilter</filter-class>  
    <init-param>  
        <param-name>spnego.allow.basic</param-name>  
        <param-value>false</param-value>  
    </init-param>  
    <init-param>  
        <param-name>spnego.allow.localhost</param-name>  
        <param-value>false</param-value>  
    </init-param>  
    <init-param>  
        <param-name>spnego.allow.unsecure.basic</param-name>  
        <param-value>true</param-value>  
    </init-param>  
    <init-param>  
        <param-name>spnego.login.client.module</param-name>  
        <param-value>spnego-client</param-value>  
    </init-param>  
    <init-param>  
        <param-name>spnego.krb5.conf</param-name>  
        <param-value>krb5.conf</param-value>  
    </init-param>  
    <init-param>  
        <param-name>spnego.login.conf</param-name>  
        <param-value>login.conf</param-value>  
    </init-param>  
    <init-param>  
        <param-name>spnego.preauth.username</param-name>  
        <param-value>username</param-value>  
    </init-param>  
    <init-param>  
        <param-name>spnego.preauth.password</param-name>  
        <param-value>password</param-value>  
    </init-param>  
    <init-param>  
        <param-name>spnego.login.server.module</param-name>  
        <param-value>spnego-server</param-value>  
    </init-param>  
    <init-param>  
        <param-name>spnego.prompt.ntlm</param-name>  
        <param-value>false</param-value>  
    </init-param>  
    <init-param>  
        <param-name>spnego.logger.level</param-name>  
        <param-value>1</param-value>  
    </init-param>  
</filter>
```

```

<filter-mapping>
    <filter-name>SpnegoHttpFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

```

#### Notes:

- The `spnego.prauth.username` and `spnego.prauth.password` represent the pre-authenticated username and password, respectively.
- If you are using keytab to do Kerberos authentication, leave `spnego.prauth.username` and `spnego.prauth.password` empty:

```

<init-param>
    <param-name>spnego.prauth.username</param-name>
    <param-value></param-value>
</init-param>
<init-param>
    <param-name>spnego.prauth.password</param-name>
    <param-value></param-value>
</init-param>

```

## Step 2: Create your krb5.conf file and put it into JBoss bin folder

The following is a sample krb5.conf file:

```

[libdefaults]
    default_realm = XXXX.COM
    default_tkt_enctypes = aes128-cts rc4-hmac des3-cbc-shal des-cbc-md5 des-cbc-crc
    default_tgs_enctypes = aes128-cts rc4-hmac des3-cbc-shal des-cbc-md5 des-cbc-crc
    permitted_enctypes = aes128-cts rc4-hmac des3-cbc-shal des-cbc-md5 des-cbc-crc

[realms]
    XXX.COM = {
        kdc = xxx.xxx.com
        default_domain = xxx.COM
    }

[domain_realm]
    .xxx.com = XXX.COM

```

**Note:** `xxx.xxx.com` represents your domain control machine.

After you have created your krb5.conf file, copy it to your `<JBOSS_HOME>/bin` folder (for example, `/home/appserv/AppServ/jboss-5.1.0.GA/bin`).

## Step 3: Change your login-config.xml file

The login-config.xml file is located in `<JBOSS_HOME>/server/default/conf` (for example, `/home/appserv/AppServ/jboss-5.1.0.GA/server/default/conf/login-config.xml`).

Append the following XML to the file, before the last end-tag named `policy`:

```

<application-policy name="spnego-client">
    <authentication>
        <login-module code="com.sun.security.auth.module.Krb5LoginModule" flag="required" />
    </authentication>
</application-policy>

<application-policy name="spnego-server">
    <authentication>
        <login-module code="com.sun.security.auth.module.Krb5LoginModule" flag="required">
            <module-option name="storeKey">true</module-option>
        </login-module>
    </authentication>
</application-policy>

```

If you are using keytab to do Kerberos authentication, add the following XML to your login-config.xml file:

```

<application-policy name="spnego-client">
    <authentication>
        <login-module code="com.sun.security.auth.module.Krb5LoginModule" flag="required" />
    </authentication>
</application-policy>

```

```

<application-policy name="spnego-server">
  <authentication>
    <login-module code="com.sun.security.auth.module.Krb5LoginModule" flag="required">
      <module-option name="debug">true</module-option>
      <module-option name="principal"><principal_in_keytab></module-option>
      <module-option name="storeKey">true</module-option>
      <module-option name="useKeyTab">true</module-option>
      <module-option name="doNotPrompt">true</module-option>
      <module-option name="keyTab"><location_of_keytab></module-option>
    </login-module>
  </authentication>
</application-policy>

```

#### Notes:

- <**principal\_in\_keytab**> is the principal name inside the keytab, such as **HTTP/pe05.conceptwave.com @CONCEPTWAVE.COM**
- <**location\_of\_keytab**> is the physical file-system location of the keytab file, such as **/home/appserv/abc.keytab**.

### Step 4: Register a Service Principal Name

The best way to register a Service Principal Name (SPN) is to have your Administration or Operations team perform this task. The registration process consists of using the setspn.exe utility to register all your SPNs by logging on as Administrator on any machine that has the support tools installed and run the pertinent commands.

**Warning:** A given SPN can be registered to only one Domain Account. If you decide to register a given SPN, ensure that it is not already registered to another Windows NT Domain Account. Otherwise, you will have to de-register or delete the mapping before registering the SPN to the other account.

The following is an example:

```

setspn.exe -A HTTP/pe05 domainAccount
setspn.exe -A HTTP/pe05.conceptwave.com domainAccount

```

Where:

- **pe05** is the host machine for JBoss.
- **domainAccount** is the pre-authenticated account. It can be a regular account that logs you into your domain.
- This domainAccount and its password would be used in **spnego.prauth.username** and **spnego.prauth.password** in [web.xml](#).

If you are using keytab to do Kerberos authentication, ask your Administration or Operations department to create a keytab file:

```

ktpass -out c:\temp\b.keytab -princ <principal> -mapUser <account> -mapOp set -pass <password>

```

#### Notes:

- <**principal**> is the principal in the keytab file, such as [HTTP/pe05.conceptwave.com@CONCEPTWAVE.COM](#). The format is [HTTP/host.domain@Realm](#).
- <**account**> is the domainAccount.
- <**password**> is the password for this domain account.

### Step 5: Download spnego.jar and place it into the JBoss library

The spnego.jar file is available from <http://sourceforge.net/projects/spnego/files/>. Download the latest version of the file and place it in <JBoss\_HOME>/server/default/lib (for example, /home/appserv/AppServ/jboss-5.1.0.GA/server/default/lib).

### Step 6: Configure Web browser settings

See the [Configure Microsoft Clients to Use Windows Integrated Authentication](#) section for configuring your Internet Explorer browser settings.

# Configure JBoss 7.1 Kerberos and Active Directory Single Sign-on Authentication

This procedure for configuring JBoss 7.1 Kerberos and Active Directory for single sign-on authentication consists of six steps:

1. [Change your cwf.war file](#)
2. [Change your standalone.xml file](#)
3. [Register a Service Principal Name](#)
4. [Configure Web browser settings](#)

## Step 1: Change your cwf.war file

To change your cwf.war file, complete these steps:

1. Add the following XML to web.xml. You can append it at the end of web.xml (that is, before </web-app>).

```
<login-config>
  <auth-method>SPNEGO</auth-method>
  <realm-name>SPNEGO</realm-name>
</login-config>
<security-constraint>
  <web-resource-collection>
    <web-resource-name>SSO Application</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>*</role-name>
  </auth-constraint>
</security-constraint>
<security-role>
  <description>The role that is required to log in to the Application</description>
  <role-name>*</role-name>
</security-role>
```

2. Add jboss-web.xml into cwf.war, in cwf.war\WEB-INF. The jboss-web.xml file contains the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web>
  <context-root>cwf</context-root>
  <security-domain>SPNEGO</security-domain>
  <valve>
    <class-name>org.jboss.security.negotiation.NegotiationAuthenticator</class-name>
  </valve>
  <jacc-star-role-allow>true</jacc-star-role-allow>

</jboss-web>
```

If required, you can change the context root (that is, <context-root>cwf</context-root>).

3. Change jboss-deployment-structure.xml inside cwf.war, in cwf.war\META-INF by adding the following inside <dependencies></dependencies>:

```
<module name="org.jboss.security.negotiation" />
```

## Step 2: Change your standalone.xml file

The standalone.xml file is located in <JBoss\_HOME>/standalone/configuration (for example, /home/appserv/AppServ/jboss-as-7.1.1.Final/standalone/configuration/standalone.xml).

Perform these steps:

1. Append the following XML to the file as system properties:

```
<property name="java.security.krb5.kdc" value="<KDC Server>"/>
<property name="java.security.krb5.realm" value="<REALM>"/>
```

The following is an example:

```
<system-properties>
  <property name="java.security.krb5.kdc" value="carta.conceptwave.com"/>
  <property name="java.security.krb5.realm" value="CONCEPTWAVE.COM"/>
</system-properties>
```

2. Append the following XML to the `<subsystem xmlns="urn:jboss:domain:security:1.1">` section:

```
<security-domain name="host" cache-type="default">
  <authentication>
    <login-module code="Kerberos" flag="required">
      <module-option name="storeKey" value="true"/>
      <module-option name="useKeyTab" value="true"/>
      <module-option name="principal" value="<principal_in_keytab> " />
      <module-option name="keyTab" value="<location_of_keytab>" />
      <module-option name="doNotPrompt" value="true"/>
      <module-option name="debug" value="true"/>
    </login-module>
  </authentication>
</security-domain>
<security-domain name="SPNEGO" cache-type="default">
  <authentication>
    <login-module code="SPNEGO" flag="requisite">
      <module-option name="password-stacking" value="useFirstPass"/>
      <module-option name="serverSecurityDomain" value="host"/>
      <module-option name="removeRealmFromPrincipal" value="true"/>
    </login-module>
  </authentication>
</security-domain>
```

#### Notes:

- `<principal_in_keytab>` is the principal name inside the keytab, such as `HTTP/pe05.conceptwave.com @CONCEPTWAVE.COM`
- `<location_of_keytab>` is the physical file-system location of the keytab file, such as `/home/appserv/abc.keytab`.

## Step 3: Register a Service Principal Name

The best way to register a Service Principal Name (SPN) is to have your Administration or Operations team perform this task. The registration process consists of using the setspn.exe utility to register all your SPNs by logging on as Administrator on any machine that has the support tools installed and run the pertinent commands.

**Warning:** A given SPN can be registered to only one Domain Account. If you decide to register a given SPN, ensure that it is not already registered to another Windows NT Domain Account. Otherwise, you will have to de-register or delete the mapping before registering the SPN to the other account.

The following is an example:

```
setspn.exe -A HTTP/pe05 domainAccount
setspn.exe -A HTTP/pe05.conceptwave.com domainAccount
```

Where:

- `pe05` is the host machine for JBoss.
- `domainAccount` is the pre-authenticated account. It can be a regular account that logs you into your domain.
- This `domainAccount` and its password would be used in `spnego.prauth.username` and `spnego.prauth.password` in [web.xml](#).

If you are using keytab to do Kerberos authentication, ask your Administration or Operations department to create a keytab file:

```
ktpass -out c:\temp\b.keytab -princ <principal> -mapUser <account> -mapOp set -pass <password>
```

#### Notes:

- `<principal>` is the principal in the keytab file, such as `HTTP/pe05.conceptwave.com@CONCEPTWAVE.COM`. The format is `HTTP/host.domain@Realm`.
- `<account>` is the `domainAccount`.
- `<password>` is the password for this domain account.

## Step 4: Configure Web browser settings

See the [Configure Microsoft Clients to Use Windows Integrated Authentication](#) section for configuring your Internet Explorer browser settings.



## Login Script for Single Sign-On

---

The single sign-on JavaScript plugin is available in Velocity Studio. The plugin is called when a user accesses the application for the first time without having an established session context. The login script returns the user ID of the application user. The login page is bypassed when the returned value matches a user ID recorded in the user profile. Otherwise, the login page appears.

This plugin has full access to HTTP header information. It also has access to all declared third-party interfaces.

### cwf\_up.preLogin Script

The cwf\_up.preLogin global script accepts the following parameters, where each parameter is an array of Strings:

- headerNames and headerValues, which are the name-value pairs for request headers
- paramNames and paramValues, which are the name-value pairs for request headers

It is recommended that you return a non-null value using this script and that you capture all exceptions. If the script returns null or if an exception occurs, you may experience a redundant call to this script. This script returns the SECURITY\_PRE\_LOGIN event type.

When you override this script and it returns a non-null value, the value is considered to be a username. If the username is valid, the user is automatically logged in and either one of the following actions occur:

- The Select Application page displays.
- If the user has permission to access only one application or the application name passes through parameters, the application appears.

**Note:** To track the URL that performed a specific request, do the following:

1. Open your metadata and override the cwf\_up.preLogin global script.
2. During runtime, instead of going to <http://localhost:8080/cwf>, try a different URL, such as <http://localhost:8080/cwf/preTest>.
3. In this script, the **requestedPath** parameter contains **/preTest**.

### Pass Parameters between Applications

To pass parameters between applications, see the following example for assistance:

```
Global.setSessionParameter( "customerName" ,  customerName );
.....
var customerName = Global.getSessionParameter( "customerName" );
```

**Note:** The setSessionParameter and getSessionParameter methods cannot be used in neither user, nor global processes. See the JavaScript documentation for details.

## Session Timeout Warning

---

The session timeout warning indicates that the current session is about to expire and provides a means to cancel the session timeout. The timeout warning appears as a popup with a warning message. However, the user interface displayed in a popup can be customized in the metadata.

### Configure Session timeout warning (seconds) parameter

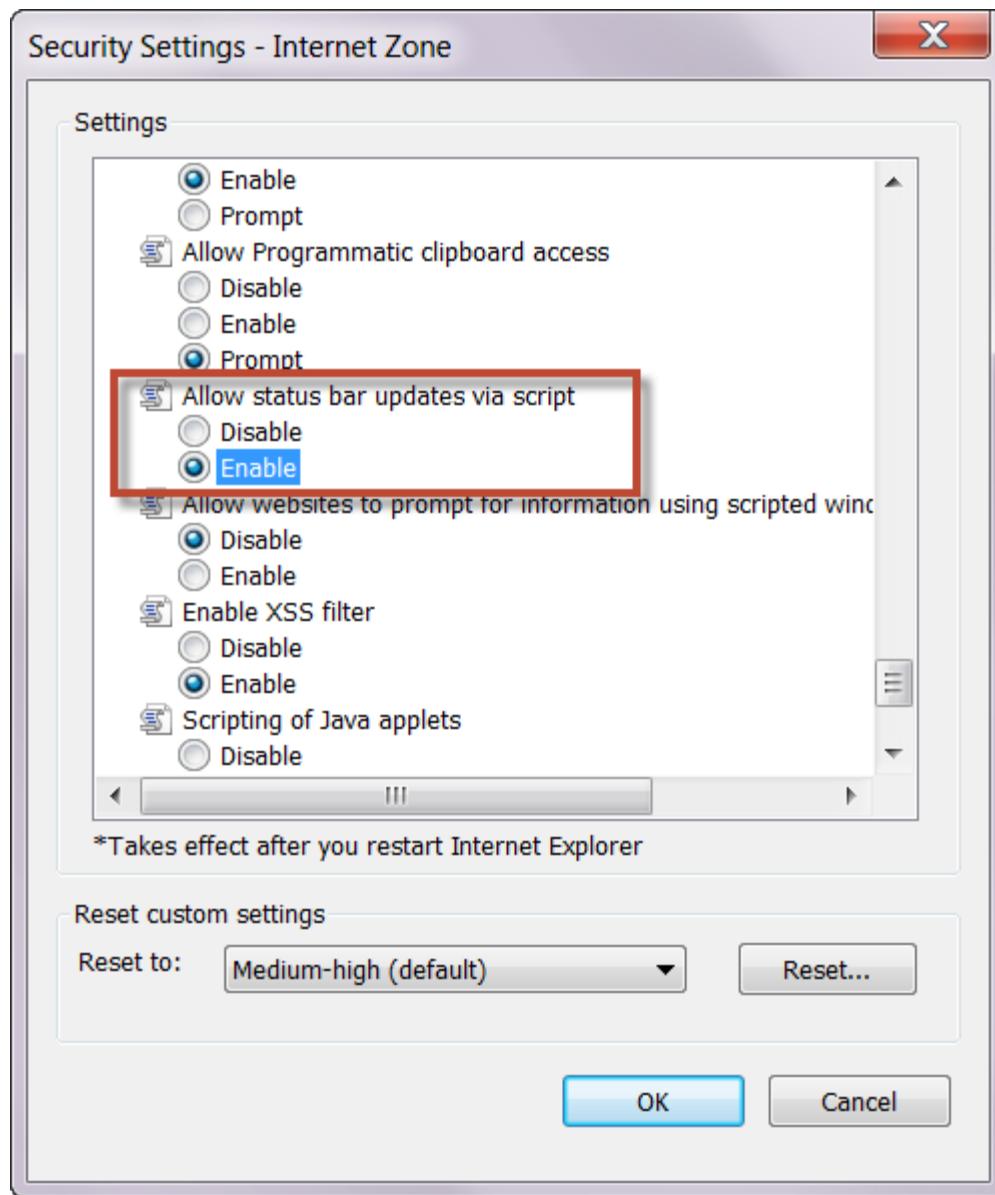
You can configure the [\*\*Session timeout warning \(seconds\)\*\*](#) parameter in the System Configuration application. From the menu bar, click **Configure > System > UI > General**.

### Configure Internet Explorer and Firefox to Display Popup

To see the session timeout warning popup appear in either Internet Explorer or Firefox, you must configure your browser settings.

In Internet Explorer, do the following:

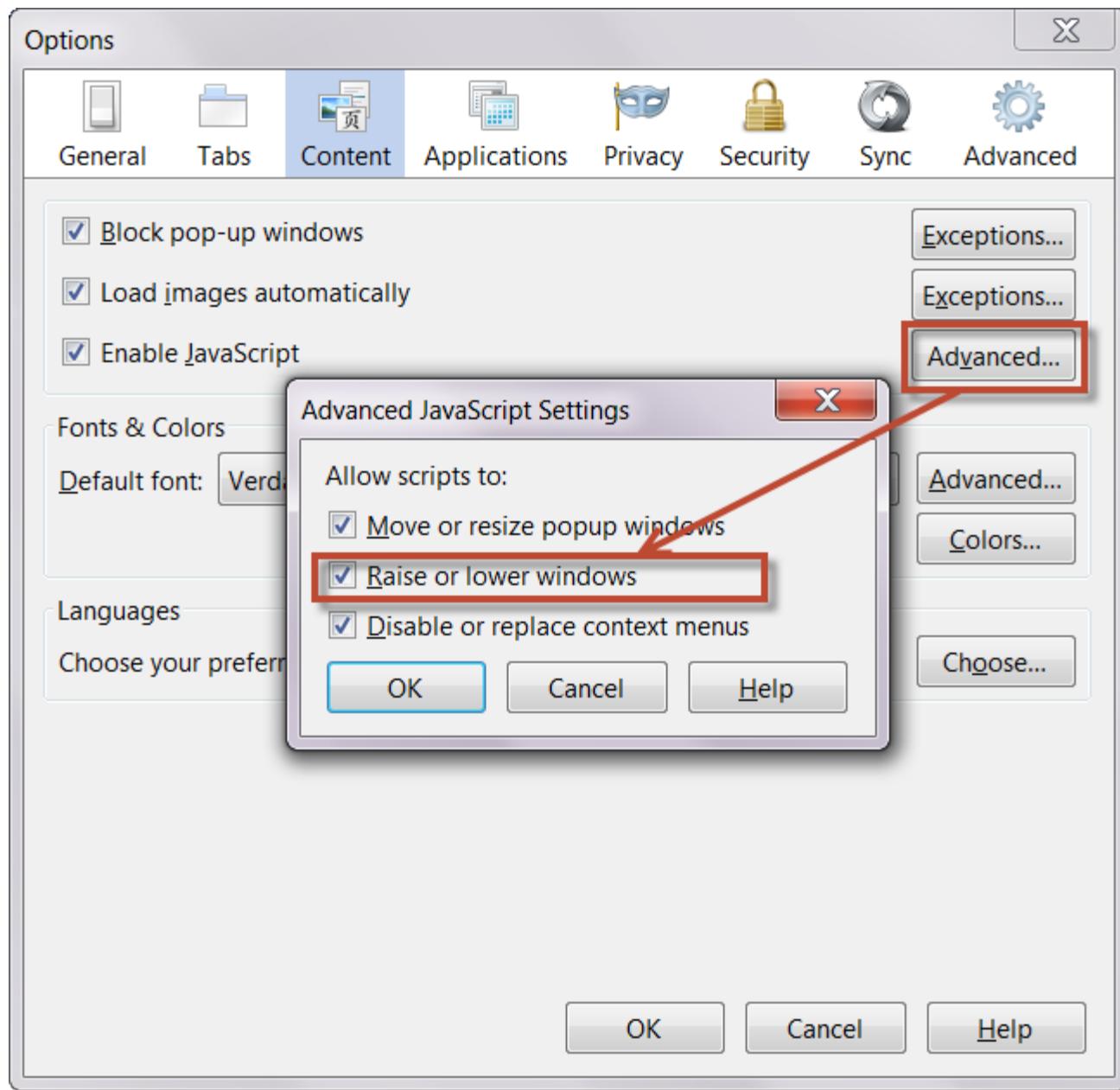
1. Click the **Tools** icon and select **Internet Options** to launch its dialog.
2. Click the **Security** tab and then click the **Custom level...** button to launch the Security Settings dialog.
3. Under the **Scripting** heading, locate **Allow status bar updates via script** and select its **Enable** option.



- e
4. Click the **OK** button. A warning message appears. Confirm your change by clicking the **Yes** button.
  5. On the Security tab, click the **Apply** button, and then click the **OK** button to save your change.

In Firefox, do the following:

1. From the menu bar, click **Tools > Options**.
2. From the Options dialog, click the **Content** icon.
3. Click the **Enable JavaScript** field's **Advanced** button to launch the Advanced JavaScript Settings dialog.
4. Select the **Raise or lower windows** setting and then click the **OK** button.



5. Click the **OK** button to save your changes.

## System Metadata

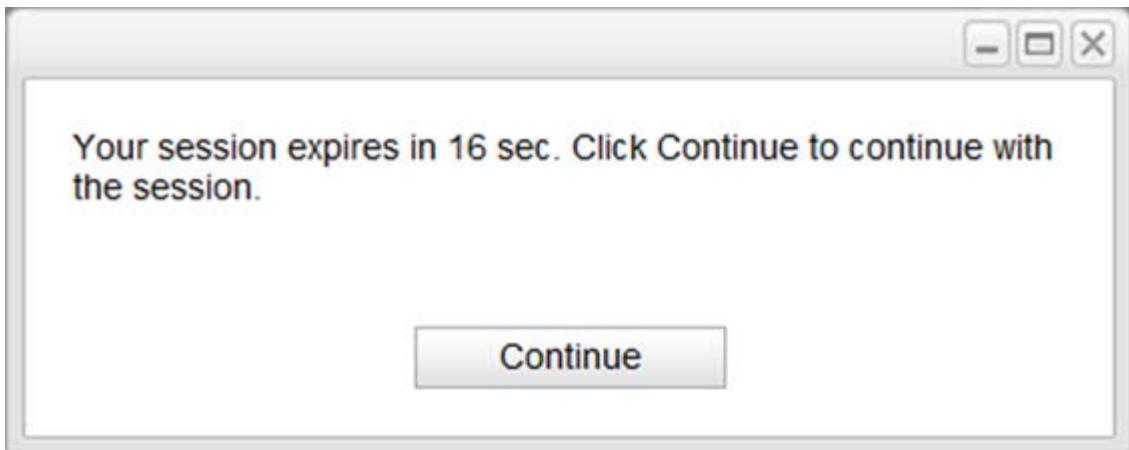
The session timeout warning event invokes the `cwf_oe:sessionTimeoutWarning` global script that generates a popup window displaying the `com.conceptwave.system.SessionTimeoutWarning` user interface.

The system implementation of `com.conceptwave.system.SessionTimeoutWarning` consists of the following warning message:

Your session expires in X seconds. Click Continue to continue with the session.

In the warning message, X represents the session timeout warning time, which counts down to zero. Clicking the **Continue** button resets the session timeout.

The configured session timeout appears as a popup clock counter to show the elapsed time. As an example, at twenty seconds, the clock shows the seconds decrease to zero. When the clock reaches zero, the session times out and the popup disappears.



You can override com.conceptwave.system.SessionTimeoutWarning in your metadata to display any user interface in the warning popup. Any user action (for example, a button click) cancels the session timeout automatically.

## User Role Selection

---

When users log in and belong to multiple user groups, they can select one user group as their active one. Selecting an active user group means that only those privileges associated with that user group are applied to their user account.

On the login page, a reference finder is available, allowing users to select their user group. The reference finder is invisible by default. However, it is visible only if users override the login document and reset the **Visible** permission.

In runtime, users have the following two options:

- Users enter their username and password, and then log in. All privileges associated with the user groups that they belong to are added to their username.
- Users can employ the reference finder to select arbitrary groups. After they log in, only privileges belonging to the selected user group are added to their username.

## Permissions

Permissions grant object operations to participants/privileges. Two types of permissions can be created in the system:

- *Global Permissions methods* that, once created, can be reused by any object in any namespace object.
- *Object Permissions methods (local)* are created on local Objects to extend or override permissions. These permissions are used by the Rules and Forms of this particular Object.

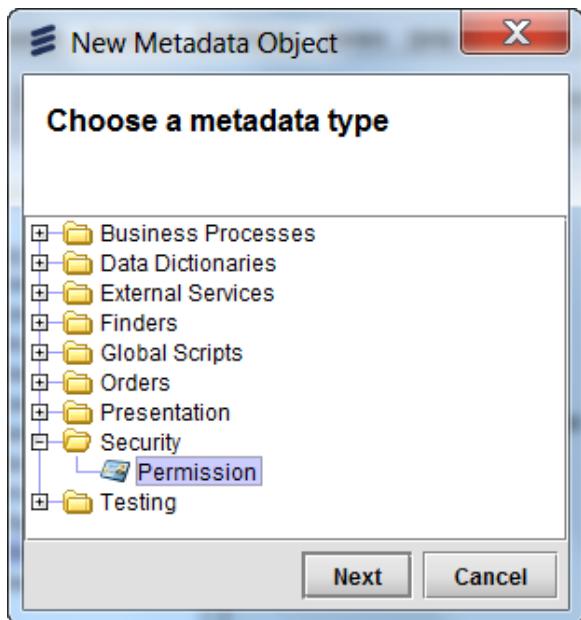
If there are no permissions assigned to an Object, the default permissions (all permissions enabled for all privileges and participants) will be used by the runtime system.

Permissions are methods of the local object. Once created, these methods can be called from any script that instantiates the Object.

### Define Global Permissions

To create a Global Permission, complete these steps:

1. Right-click the **Namespace** folder in the **Navigation** pane of Velocity Studio.
2. Select the **New ...** from the list. This will open a pop-up used to select the Metadata Object type.
3. Double-click the **Security** folder in the **Navigation** pane, or click the **+** before the **Security** folder to expand tree and show the **Permission** folder.



4. Select the **Permission** folder, which enables the **Next** button at the bottom of the pop-up. Click the **Next** button to continue.
5. Enter the **Name** of the Global Permission in the **Name:** field. The **Finish** button is only enabled after a valid name is entered.

**Note:** No spaces are allowed.

6. Click the **Finish** button to add Global Permission to the Metadata.

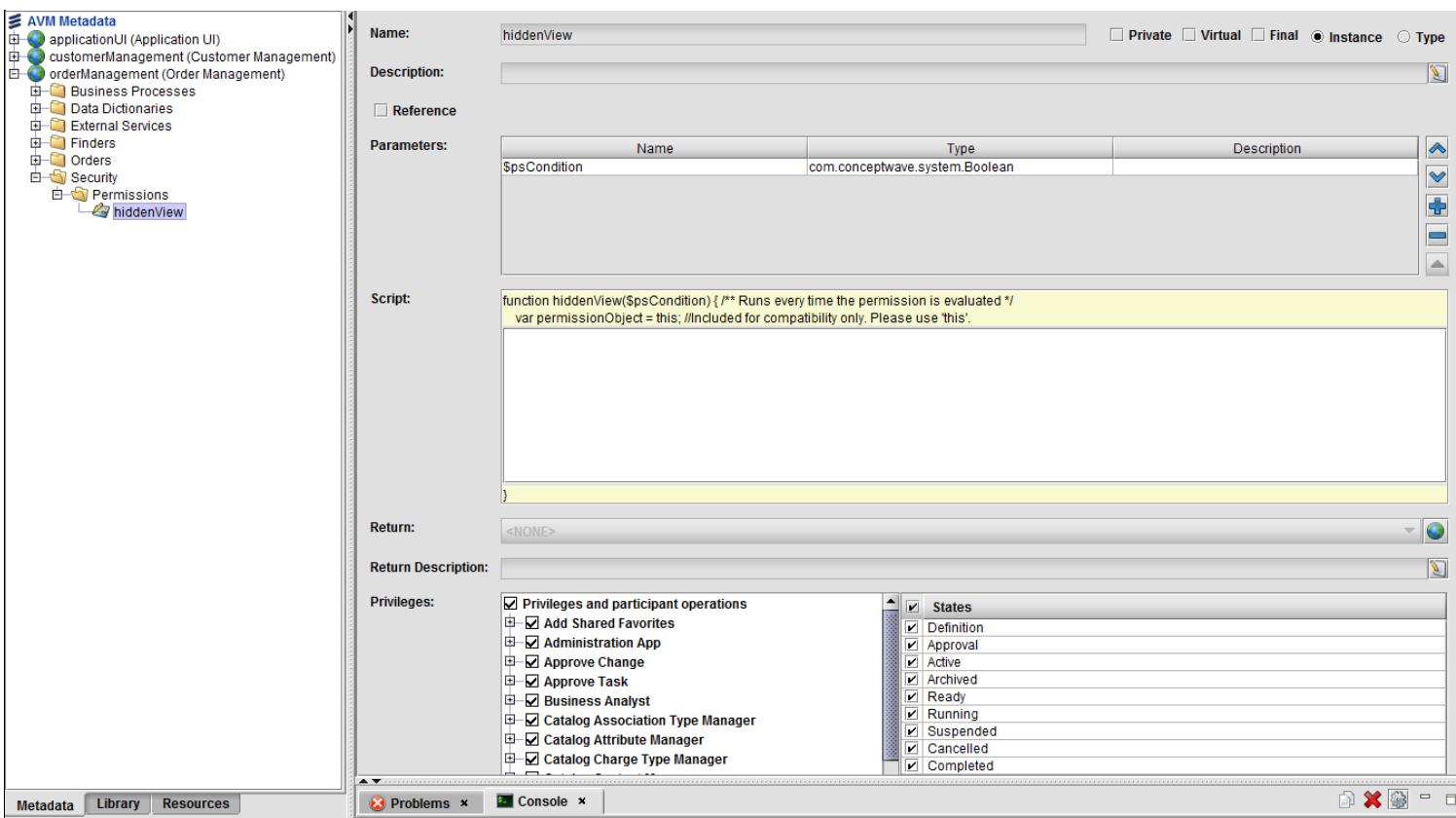
After a Global Permission is created, the icon is added under the **Security > Permissions** folder.

Additional Global permissions can be created within the same namespace more quickly than the above approach once one Permission exists for a namespace by doing the following:

1. Right-click the **Permissions** folder.
2. Select **New Permission** option from the popup menu.

### Permission General Properties

The following image shows the general Permission properties:



The following table describes the fields. The same properties are used for Global and Object Permissions.

Field	Description
<b>Name</b>	Mandatory. Unique name of the Permission.
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Virtual</b>	If checked, it provides the ability to create methods in template metadata that does not have implementation, but can be used in template script and be bound to UI elements. The implementation is done in user metadata (see <a href="#">Virtual Field</a> for more information).
<b>Final</b>	Metadata elements can be marked as final, which includes any Boolean final property that is available on any metadata element. Additionally, any metadata element that has the <b>Final</b> property set to true can neither be extended, nor overridden.
<b>Instance</b>	If checked, the method is accessible by instance objects.
<b>Type</b>	If checked, the method is accessible without an instance object.
<b>Description</b>	Description of the Global Permission for documentation purposes.
<b>Reference</b>	Used to re-apply an existing Object or Global Permission. Select the existing Permission from the <b>reference</b> drop down box after the reference checkbox is selected. The permission properties from the selected item will be used. When the reference checkbox is selected the lower half of the screen that contains the script and privilege sections are hidden and replaced by the <b>reference</b> drop down box.
<b>Parameters</b>	The input arguments that are passed into the function. Parameters can be added or removed to the extended permissions using the "+" and "-" buttons on the right side of the pane. The "^" button can be used to change the order of the passed in parameters. All the pre-defined functions have one parameter \$psCondition defined and no additional parameters can be added. The "+" and "-" buttons will be disabled for default operations.
<b>Script</b>	A JavaScript condition script that is associated with the permission selected. The JavaScript can be used to evaluate any rules or conditions that the object will check.
<b>Return</b>	The data type that the script returns. By default the return value is com.conceptwave.system.Boolean.
<b>Return Description</b>	Description of the return value for documentation purposes.
<b>Privileges</b>	A tree list of available privileges (the <i>cwf:systemPrivileges</i> system data type <i>cwf:systemPrivileges</i> ) and, for each privilege, a list of the associated user participants and their operations as defined in the current metadata. The selections made in this tree are associated with the permission selected.

**States** A list of states (the cwf:allState system data type) appears. If there are no states in the system or they should not be used with the object, the **States** list does not appear in the tab.

The **Privileges and Participant operations** tree shows a list of available privileges. There are 2 modes:

- The <On Task> branch lists for each privilege, a list of related user participants with their operations. They are used to specify that a certain permission operation is allowed/disallowed if the user works on a particular task type (participant operation) from the runtime worklist. Additionally, there is an <All Others> option which allows the permission for all remaining participants (that is, those not exclusively allowed or denied the operation).
- The <No Task> item shown for each privilege allows the privilege without relation to tasks (for example, if the object accessed through a *Finder*).

A selection in the **Privileges and participant operations** tree can be made by using a mouse click or the space bar on the keyboard. Only the one-way operations from the participant interface are shown.

There is also one The **three-state** check box on the left side of the tree item shows the selection state of item's sub-tree.

The **States** list does not depend on selections made in the **Privileges and participant operations tree**. If selected, they apply only if the permission is evaluated at runtime on an Order or Order Document. In such cases, the permission will disallow the permission operation if the Order state is not in the **States** list. A state can be selected or unselected using a mouse click or the space bar on the keyboard. All states in the list can be selected or unselected using the **All** or **None** buttons respectively.

The privileges, participant operations and state assignments can be made for all permissions in the **Permissions** table, however, only applicable settings will be used by the runtime system for each particular object. This allows the user to define one permission that can be used to create reusable permissions for several different types of objects.

<b>Feature Restriction</b>	This field allows you to specify <a href="#">features that you want to restrict</a> . Click the field's <b>Checkbox</b> button ( ) to launch the Select Feature dialog. You can select all features that you want to restrict and then click the <b>Save</b> button.
----------------------------	--

## Define Object Permissions

Object Permissions can be created or modified in the **Methods** tab of the Object. Once created, the Object Permissions can be used in the Rules and Forms of the Object. If no permissions methods are created or overridden then the Object has no restrictions and full access is granted. If neither the script or privilege sections of the permission are changed then the permission does have to be defined.

Permission can be created on any object that allows Permission methods. To see if an Object supports permissions, do the following:

1. Select the Object in the **Navigation** pane.
2. Select the **Methods** tab.
3. Right-click the root node of object **Navigation** pane.
4. If the **New Permission** option exists in the list, the object supports permissions. Enter the name

For the Document Objects there are system pre-defined operations pre-defined system permissions. These system defaults can be extended by the user using **New Permission**. The default Document Permissions are inherited from the underlying Object's composition: the data types, variable, forms, etc. The only way to change this default permission is to override the pre-defined system permission. There are 5 default pre-defined system permissions: Add, Delete, Editable, Attach, Execute. The system permissions use the naming convention *operation* with **Perm** post-fixed to it. System permissions are not shown unless explicitly created using override permission, as the system default grants full access. The overridden system permissions are marked with the icon in the navigation pane, where as the extended permissions use the same icon without the green arrow.

[Descriptions of pre-defined system Permissions operations.](#)

Field	Function	Description
<b>Add</b>	<b>addPerm</b>	Specifies whether the participant can add the object (for example, document instances in an Order Collection).
<b>Delete</b>	<b>deletePerm</b>	Specifies whether the participant can delete the object (for example, document instances in an Order Collection).
<b>Update</b>	<b>editablePerm</b>	Specifies whether the participant may update the object (for example, variables on a document).  <b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.
<b>Attach</b>	<b>attachPerm</b>	Specifies whether attachments are allowed. Applies only to a Document or Order (see Section on General Document

	Properties or Section on Order General Properties for more information on attachments).
<b>Execute</b> <b>executePerm</b>	Execute is related to validation. The Execute permission specifies whether the validation script will run on a save or submit operation (for example, a manager may require override of validation rules).

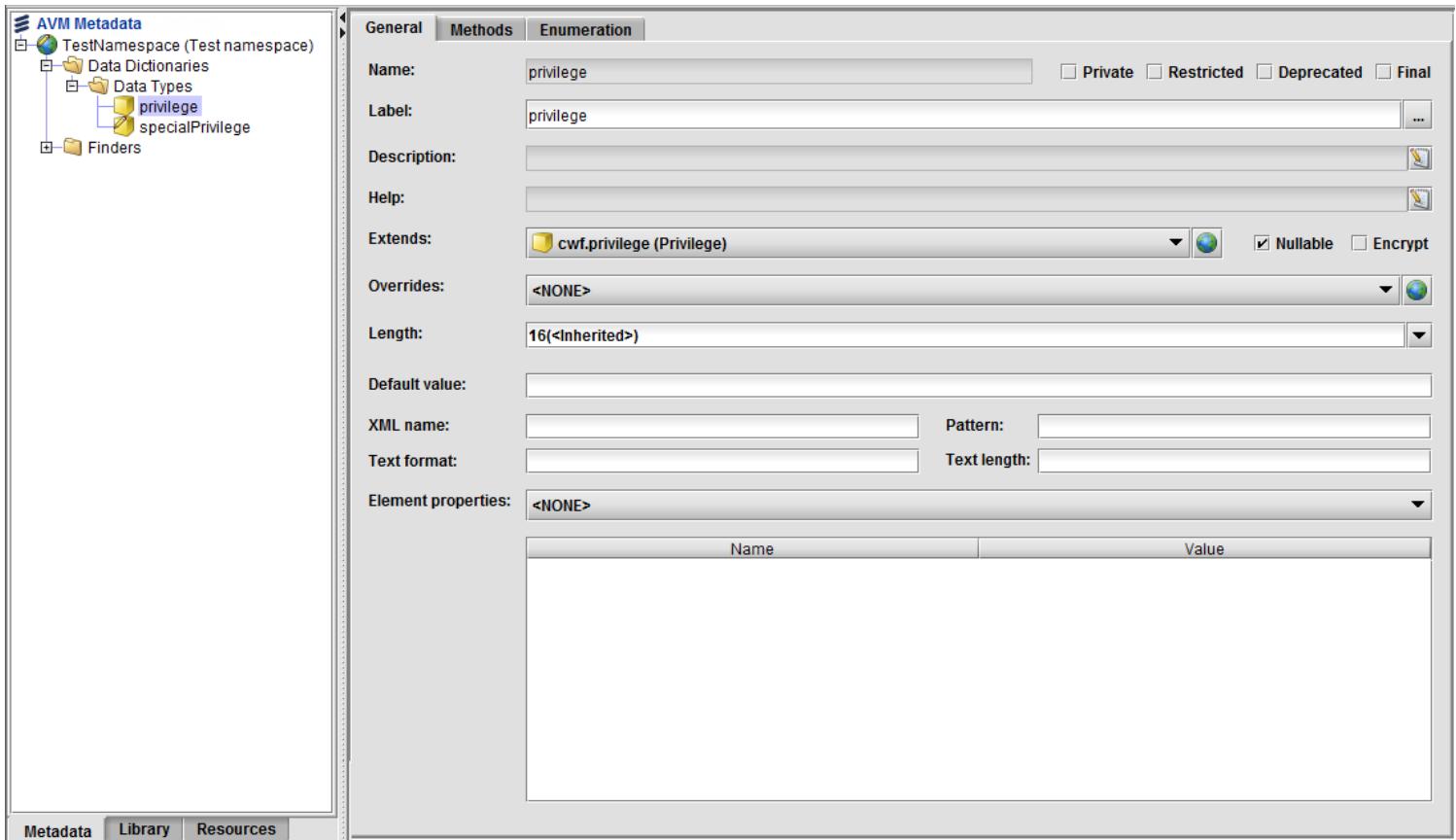
To create a Document Permission, do the following (see figure that follows):

1. Select the **Document** instance folder in the **Navigation** pane of Velocity Studio.
2. Open the **Method** tab on the Document or Object.
3. Right-click the *root instance Object* folder at the top of the method **Navigation** pane.
4. Select **Override <function>Perm** or **New Permission** from list. If extending permission continue, otherwise if overriding create permission is complete.
5. Enter a valid name into the **Name:** field on pop-up, which enables the **Finish** button.

**Note:** No spaces are allowed.

6. Select the **Reference** checkbox to select a global permission from a popup dialog.
7. Enter the name again and then select the global permission you have just defined.
8. Click the **Finish** button to add Permission to the Metadata Object.

Steps used to create a Document Permission.



In a Document that is derived, the Document Permissions from the base Document will be shown in blue and will be read-only.

After Document Permission is created, the icon will be added under the **Methods** tab .

*Velocity Studio navigation pane and Document Permission Properties.*

The screenshot shows the AVM Metadata interface. On the left, the navigation tree displays a sampleNS (SampleNS) node with Data Dictionaries, Documents, and UserInterface. Under UserInterface, there is a DocumentPerm method. The main workspace has tabs for General, Variables, Methods, and Mapping. The Methods tab is active, showing the configuration for DocumentPerm. The 'Name' field is set to 'DocumentPerm'. The 'Script' field contains the following JavaScript code:

```

function DocumentPerm() { // Metadata type method. Can be called by scripts.
}

```

The 'Return' field is set to 'com.conceptwave.system.Void'. Below the workspace is a 'Problems' tab showing logs and a 'Console' tab showing system messages.

## Positive and Negative Permissions

Permissions can be of 2 types: Positive or Negative

- A positive permission is that which the privilege is selected/checked. When multiple privileges are checked, the permission uses the **OR** operator to validate.
- A negative permission is that which the privilege is deselected/unchecked. When multiple privileges are unchecked, the permission uses the **AND** operator to validate.

This is particularly important when dealing with the system default *Everyone* permission.

## Assign Object Permissions

**Document Permissions** are assigned immediate when the permission is created in the **method** tab of the Document or Object.

**Global Permissions** are assigned by creating a Document permission with the **reference** property set to Global permission. A single Global permission can be assigned to many Document permission.

Once a Permission is created and assigned the permission can be call anytime like other scripts.

*Assignment of Global Permission (reference checked).*

The screenshot shows the AVM Metadata interface. The navigation tree on the left shows a CRM node with Business Processes, Data Dictionaries, and Documents. Under Documents, there is a CustomerInformation node with addPerm and attachPerm methods. The main workspace has tabs for General, Variables, Methods, and Mapping. The Mapping tab is active, showing the configuration for attachPerm. The 'Name' field is set to 'attachPerm'. The 'Reference' checkbox is checked, and the 'Reference' field is set to 'CRM.GlobalPermission'.

Assignment of Document Permission (reference unchecked).

Name	Type

Leaf Permissions can be assigned to any field of a Document. To assign a permission to a variable:

1. Select the Document instance using the **Navigation** pane of Velocity Studio.
2. Open the **Variables** tab on the document.
3. Select the field **Name** in the table in the top pane of form. This will refresh the lower half of page.
4. Click the **General** tab in the field properties section on the bottom half of page.
5. Click the **Value** column for the appropriate element property row of the table. The available permissions that can be assigned will be displayed along side fixed values of TRUE and FALSE.
6. Select the Document **Permission** from the list.

Assignment of a Leaf (variable) Permission.

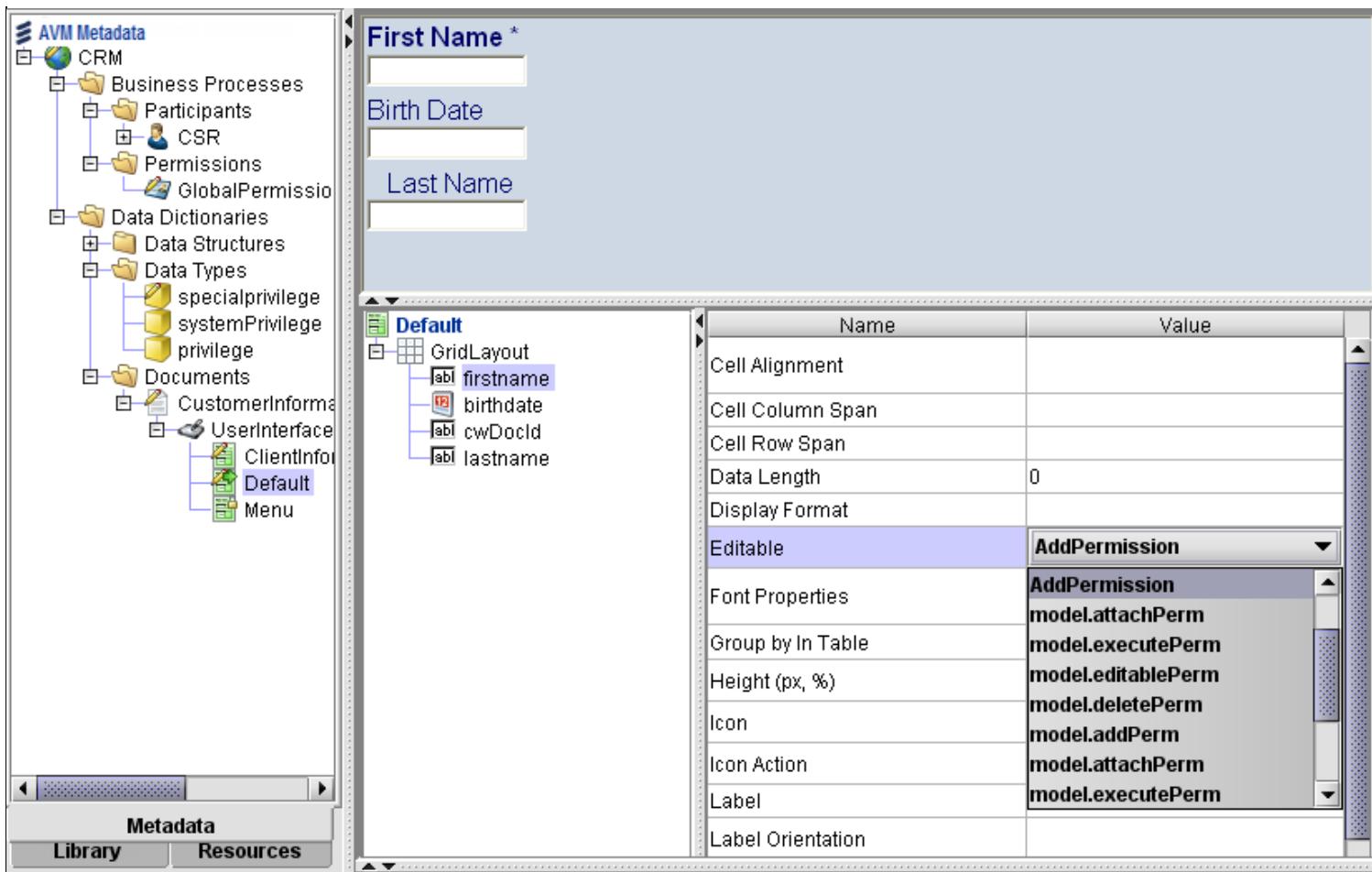
Name	Type	Methods	Vis	Opt	Edit
firstname (First Name)	String	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
lastname (Last Name)	String	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
birthdate (Birth Date)	Date	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
creditDecision (Credit Score)	String	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
cwDocId (System document ID)	String, 16	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

The icon in columns vis, opt, and edit are used to quickly indicate if a variable has a permission assigned to it. A checkmark means that TRUE has been assigned and "X" means that FALSE has been assigned to the element property.

UI Permissions can be assigned to any field of a Document form. To assign a permission to a UI field on particular form:

1. Select the UI Document Form instance using the **Navigation** pane of Velocity Studio.
2. Select the **Variable** on the document **Navigation** pane of the Object.
3. Click the **Value** column for the element property row **visible**, or **editable**, or **optional** of the table. The available permissions that can be assigned will be displayed along side fixed values of TRUE and FALSE.
4. Select the Document **Permission** from the list.

Assignment of a UI (variable) Permission.



The permissions available are the system default permission and any extended **UserInterface** Permissions.

**Note:** You can also use a Boolean variable as a permission method.

## Extend Privileges

More privileges (**Privileges**) can be added to the existing (if necessary) system Privileges by extending the **Privilege** class. This can be accomplished by adding a new **Data Type** Object to the metadata, which must be an enumeration. After the Data type object that extends the **Privilege** exists in the Metadata the new privilege will be shown in the **Privileges and Participants** section of the **Permissions** general tab. The following steps can be used to extend the existing privileges:

1. Right-click the **Namespace** folder in the **Navigation** pane of Velocity Studio.
2. Select **New ...** from the list to open the New Metadata Object dialog.
3. Double-click the **Data Dictionaries** folder in the **Navigation** pane, or click the **+** icon next to the folder to expand tree and show the **Date Type** folder.
4. Select the **Data Type** folder, which enables the **Next** button at the bottom of the popup.
5. Click the **Next** button to continue.
6. Enter the **Name** of the Data Type in the **Name:** field. The **Finish** button is only enabled after a valid name is entered.
7. To extend your data type, click the **Extends** field's drop-down menu and select a class from which to extend.
8. Click the **Finish** button to add Privilege to the metadata.

The default system Privileges can be changed if the `cwf.systemPrivileges` class is extended using the same technique.

After a Privilege is created, the icon appears under the **Data Dictionaries > Data Type** folder (see the figure that follows).

**AVM Metadata**

TestNamespace (Test namespace)

- Data Dictionaries
- Data Types
  - privilege
  - specialPrivilege
- Finders

**General Methods Enumeration**

Name: privilege  Private  Restricted  Deprecated  Final

Label: privilege

Description:

Help:

Extends: cwf.privilege (Privilege)   Nullable  Encrypt

Overrides: <NONE>

Length: 16(<Inherited>)

Default value:

XML name:  Pattern:

Text format:  Text length:

Element properties: <NONE>

Name	Value

Metadata Library Resources

## Permissions: Changes and Migration

---

The Permissions in Velocity Studio have significantly changed since 5.0. This article discusses how to migrate permission object from release 4.2.

Wizard is used to assist developer in creation of **Permission Object**

**Permission** (operations) are created directly instead of a whole **Permission Set**.

Global Permissions have been relocated under the **Navigation** pane **Business Processes -> Permissions**

All Permissions are now script methods that can be invoke any script. Developers can now extend the permission operations beyond the system default permission methods of: addPerm, deletePerm, attachPerm, editablePerm, and executePerm.

Permission Set operation mapping to Permission Methods

4.2 Permission	5.x Permission Method
Add	addPerm
Delete	deletePerm
Attach	attachPerm
Update	editablePerm
Execute	executePerm
Optional	optionalPerm
View	visiblePerm

Custom Permissions can not be created. Document Permission can perform the same function as Custom Permissions in previous release. All Custom Permissions will be migrated to Document permissions as extension to default Permissions

Global **Permission Set** can no longer be assigned directly to Object. Global permissions are assigned using Object Permission **Methods**. For each operation in a Classic Global **Permission Set**, there will be a Global Permission created. These Global Permissions will be implemented as a local Object Permission in the **Methods** tab which will use a reference property to link to the Global Permission.

**Permission Set** operations are not created automatically. Only create a permission **Methods** that change from the default - grant all access.

The general tab of the Permission now contains the properties of the permission (no additional tabs). Permissions are now methods under the **Methods** tab and no longer have a permissions tab.

Permissions now has a two new properties for input para maters and return value.

Format codes and field level scripts have been replace by field permissions.

The Metadata is used to manage privileges, not managed by administration application any more.

*Velocity Studio:Before and After Picture of Permission.*

**AVM Metadata**

**Customer Extension**

- Data Dictionaries
- External Services
- Business Processes
- Orders and Finders
- User Interface
- Security
- Permissions

  - View - Active Cus
  - View - CLIC
  - View - In Popup**
  - View - Revise Pro
  - View - Show Data
  - View - Show Data

**General Permissions**

Permission	Has Script
<input checked="" type="checkbox"/> View	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> Add	<input type="checkbox"/>
<input checked="" type="checkbox"/> Delete	<input type="checkbox"/>
<input checked="" type="checkbox"/> Update	<input type="checkbox"/>
<input checked="" type="checkbox"/> Attach	<input type="checkbox"/>
<input checked="" type="checkbox"/> Execute	<input type="checkbox"/>
<input checked="" type="checkbox"/> Optional	<input type="checkbox"/>

**Script:**

```
function cwOnPermission($psCo)
var permissionObject = this;
if (null != this.owner) return false;
```

**States:**

- Archived
- Cancelled
- Completed
- Contract
- Disconnected
- Expired
- In Provisioning
- In Service
- New
- Order
- Proposal
- Quotation
- Ready
- Return to Sender
- Split from Order
- Suspended

**AVM Metadata**

**CustomerInformation**

- Business Processes
  - Participants
  - Permissions
  - GlobalPermissionSet
- Data Dictionaries
  - Data Structures
  - privilege
    - specialprivilege
    - systemPrivilege
- Documents
  - CustomerInformation
    - Userinterface
    - ClientInformation
    - Default
    - Menu

**General Variables Methods Mapping**

**General**

**Name:** addPerm

**Description:**

**Parameters:** Name

**Script:**

```
function addPerm() { // Metadata type met
```

**Return:** com.conceptwave.system.Boolean

**Privileges:**

- Privileges and participant operations
- Add Shared Favorites
- Administration App
- ConceptWave API Access
- CW Developer
- CWT-CU - Add Customer
- CWT-CU - Address Management /
- CWT-CU - Customer Management
- CWT-CU - Delete Customer
- CWT-CU - Manage Carriers
- CWT-CU - Manage Customers
- CWT-CU - Manage Partners
- CWT-CU - Manage Suppliers
- CWT-CU - Party Management Admin
- CWT-PC - Catalog Management A
- CWT-PC - Manage Components

## External Services

---

The framework provides a metadata-driven mechanism for defining interfaces to external services. The product uses WSDL (Web Services Description Language), an XML format for describing Web services.

WSDL defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the re-use of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and interfaces (or port types), which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable binding. A collection of ports defines a service.

### WSDL Definitions

The WSDL elements can be divided into two types of definitions: Abstract Definitions and Concrete Definitions. The abstract elements are platform, protocol and language independent. The concrete elements specify the physical details required to interact with the external service. This allows the defining of a set of services that provide the same service (implement the same interface) but reside in different locations, or are implemented on different technologies (for alternate route, backup service, etc).

WSDL uses the following elements in the definition of a network service:

#### Abstract Definitions

- *Data Type*: Machine and language independent XSD-based type definitions.
- *Message*: An abstract, typed definition of the data being communicated (input and output messages). Messages consists of a collection of typed data items and Documents and Data Structures used to model WSDL messages.
- *Operation*: An abstract description of an action supported by the service. Each Interface Operation has a name, and describes input and output parameters (messages).
- *Interface*: An abstract set of operations supported by one or more endpoints (ports).

#### Concrete Definitions

- *Binding*: A concrete protocol and data format specification for a particular Interface. The binding information includes such details as protocol, serialization and encoding on the wire. These details are specific to the service provider type (that is, EJB, SOAP, COM, etc).
- *Port*: A single endpoint defined as a combination of a binding and a network address (location).
- *Service*: A collection of related ports.

Velocity Studio uses an abstract definition (Interface and Operation) when it needs to specify an external service invocation. The framework will automatically select the appropriate port where the actual service is located. The concrete port selection is determined by service availability (refer to calendars and error status described below).

### Defining External Service Elements

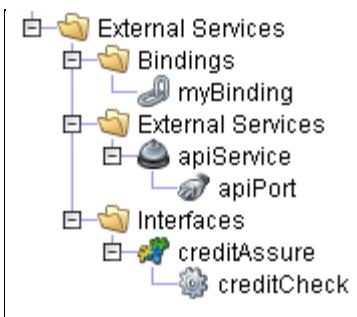
The External Service elements can be defined under the **ExternalServices** sub-folders of any namespace in the **Navigation** pane of Velocity Studio. The elements that can be created include Interfaces, Bindings and Services.

In addition, you may be required to define [Data Types](#), [Documents](#), and [Data Structures](#) that are used as messages in Interface Operations.

Refer to [Type Mapping with External Services](#) for information on mapping the native base Data Types to different technologies.

External Service leverages [built-in service providers](#) in the framework, such as SOAP, HTTP, JMS, etc. There are specific properties in External Service metadata such as Interface and Port attributes for each built-in service provider. Please see the corresponding [documentation](#) for details.

After an External Service element is created, the icon will be added under the appropriate folder.



After External Service elements are created and properly defined, there are more settings of the external service to be configured in [Configuration application's Service page](#). These settings are instance-specific and thus to be configured in Configuration application only, and not available to be configured in the metadata / Velocity Studio.

To export all External Services in a namespace, refer to [Export WSDL](#) for details.

The CWPRODUCTPROPERTIES table contains the CW-NAMESPACE-AWARE data value, which allows you to turn namespace processing on or off for XML 1.0 support. By default, this value is true. To turn off this support, set CW-NAMESPACE-AWARE to 1.

## External Service-Related Classes

The following classes are available for external services:

- CwfMetadataAlert
- CwfMetadataPort
- CwfMetadataExternalService
- CwfMetadataInterfaceBinding

For more information, see the JavaScript documentation.

## Defining an Interface

An Interface represents a collection of abstract operations.

To create a new Interface:

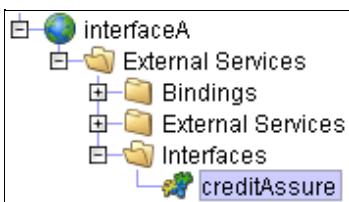
In the **Metadata** tab of **Navigation** pane, either:

1. Right-click a Namespace. Select **New ...**
2. **New Metadata Object** wizard appears as pop-up. Expand **External Services**, and select **Interface**, and then click **Next**.
3. Step two of wizard appears. Type in the name of Interface (must conform to JavaScript naming conventions), and populate other fields as desired. Then click **Finish**.

OR

1. Right-click the **Interfaces** folder or the **External Services** folder in a Namespace, if it is present. Select **New Interface**
2. Follow step3 from above.

After the Interface is created, the icon will be added under the **Interfaces** folder.



### Interface General Properties

The general Interface properties are defined on the **General** tab. The table below describes the fields.

General		Operations
<b>Name:</b>	creditAssure	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated
<b>Label:</b>	creditAssure	
<b>Description:</b>		
<b>Privileges:</b>		<input checked="" type="checkbox"/>
<b>Revision:</b>	3	

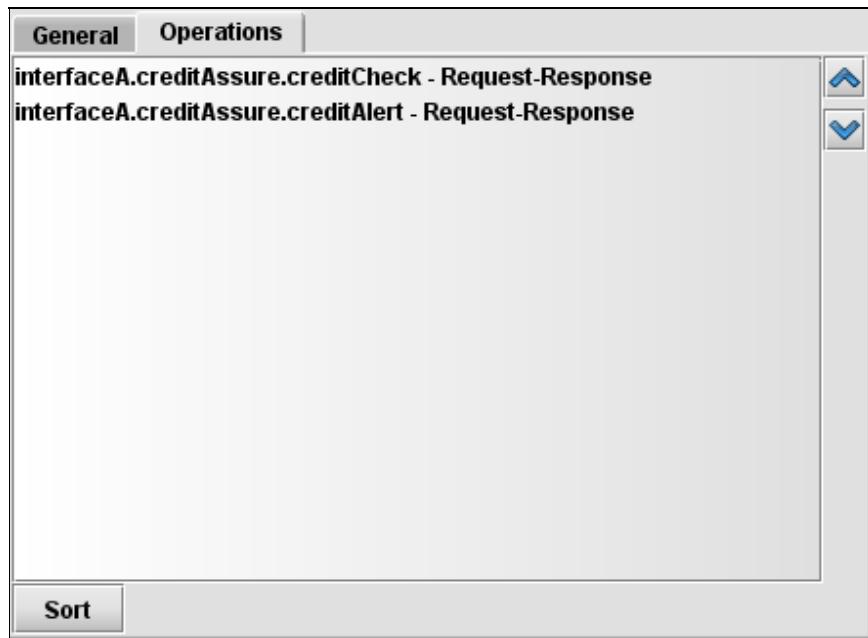
Field	Description
<b>Name</b>	Mandatory. Name of the Interface within the namespace.
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Label</b>	Mandatory. Label to identify the Interface.
<b>Description</b>	Description of the Interface for documentation purposes.
<b>Privileges</b>	When the button is clicked at the end of the field, the <b>Select Privileges</b> dialog opens, allowing the user to select a combination of user privileges.

	A Privilege is a user right which is assigned to a user and specifies allowable actions in the system. Privileges in an Interface definition specify who can use this Interface and the Interface Operations. If privileges are not defined, any user can use the Interface. Currently, privileges are enforced only for API users (external systems using services of the framework).
<b>Revision</b>	Read-only. Revision number of the Interface; increments when any of its Interface Operations is added or removed. At the moment, this is for informational purposes only -- previous revisions of the Interface are not stored, and there is no runtime functionality that leverages this revision number.

## Interface Operations

The **Operations** tab contains a **Children list** which lists all the Interface Operations. The order of the Operations is dependant on the order in the list.

The arrow buttons located at the side of the list are used to move the selected element in the list to the desired location. The **Sort** button is used to sort the elements in alphabetical order.



## Defining an Interface Operation

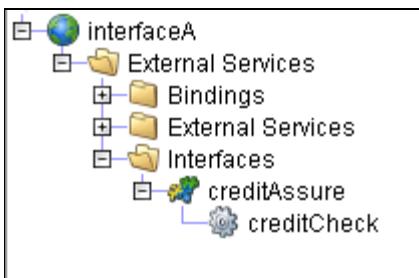
An Interface Operation is an abstract description of an action supported by the service. It defines messages that are exchanged between the service provider and AVM runtime. The Interface Operation name is usually served as a function or request name that is invoked on the service provider.

An Interface Operation can be added directly under an Interface element.

To add an Interface Operation, do the following:

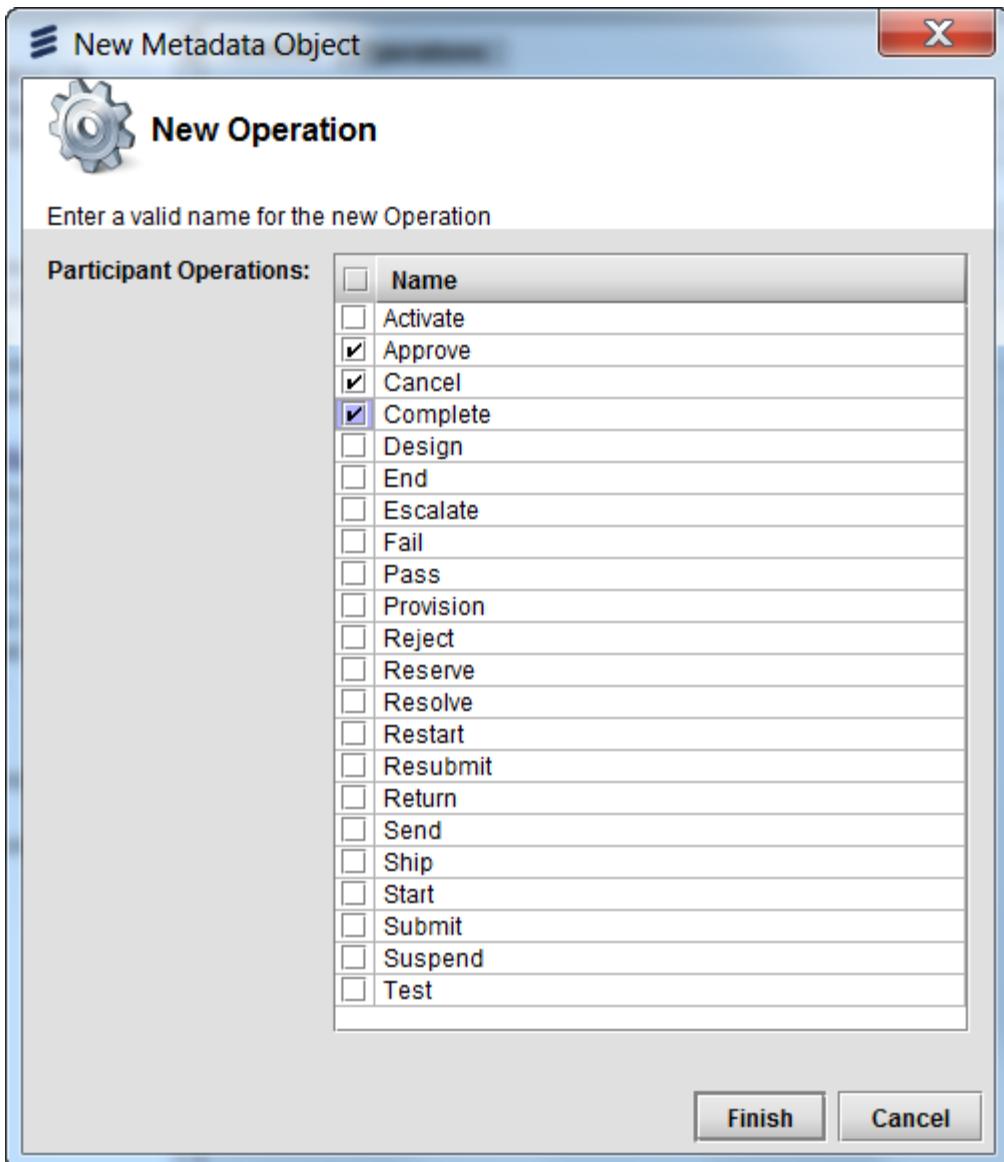
1. Right-click the **Interface** node in the **Navigation** pane of Velocity Studio.
2. Select the **New Operation** command. The **New Operation** wizard appears as pop-up.
3. Type in the name of Interface Operation (must conform to JavaScript naming conventions), and populate other fields as desired.
4. Click the **Finish** button.

After the Interface Operation is created, the icon appears under the Interface icon.

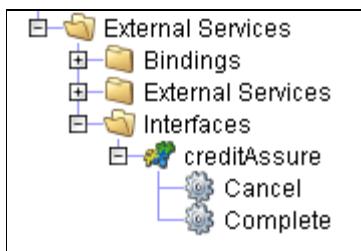


## Defining Interface Operations for (manual) Participant

As a shortcut for creating [manual participant](#) response choices, multiple common Interface Operations can be created under an Interface using the **New Participant Operations** in its right-click pop-up menu. Upon clicking the command, a pop-up appears to allow selecting from a list of common responses in manual participant operations:



Select the items from the list and then click the **Finish** button. For each selected item, an Interface Operation of **Notification** type is created based on the item name.



#### Interface Operation General Properties

You can define general Interface Operation properties on the **General** tab. The following table describes the fields.

Name:	creditCheck
Label:	creditCheck
Description:	<input type="text"/>
Input:	<input type="button" value="interfaceA.creditRequest"/> <input type="button" value="New..."/> <input type="button" value="Edit..."/> <input type="button" value="Delete..."/>
Output:	<input type="button" value="interfaceA.creditResponse"/> <input type="button" value="New..."/> <input type="button" value="Edit..."/> <input type="button" value="Delete..."/>
Fault:	<input type="button" value="&lt;NONE&gt;"/> <input type="button" value="New..."/> <input type="button" value="Edit..."/> <input type="button" value="Delete..."/>
Operation type:	<input type="button" value="Request-Response"/> <input type="button" value="New..."/> <input type="button" value="Edit..."/>
Privileges:	<input type="checkbox"/>

Description of fields in General tab of the Interface Operation properties.

Fields	Description		
Name	Mandatory. Name of the Interface Operation within the namespace.		
Label	Mandatory. Label to identify the Interface Operation.		
Description	Description of the Interface Operation for documentation purposes.		
Input	Message or input parameters that are sent to the service provider.		
Output	Message or output parameters that the service provider responds with.		
Fault	Message that can be sent by the service provider instead of an <b>Output</b> message, when the service provider wants to respond with a fault (error) condition. Fault messages are supported only by service providers that are capable of sending XML messages (such as SOAP or XML over HTTP).		
Operation type	The type of message exchange that this Interface Operation supports. The type can be one of the following four options:		
Operation Type	Icon	Description	
One-way		The service receives an input message, but does not generate any output message.	
Request-response		The service receives an input message and sends an output message.	
Solicit-response		Opposite of <i>request-response</i> . The service sends an output message and receives an input message. <b>Note:</b> This type appears for completeness to the specification, but is currently not implemented in the framework. This type of operation is normally used by subscription-based services.	
Notification		Opposite to <i>one-way</i> . The service sends an output message. This type of operation would normally be used for event listeners. However, it is possible to create an interface call with a notification type.	

#### Notes:

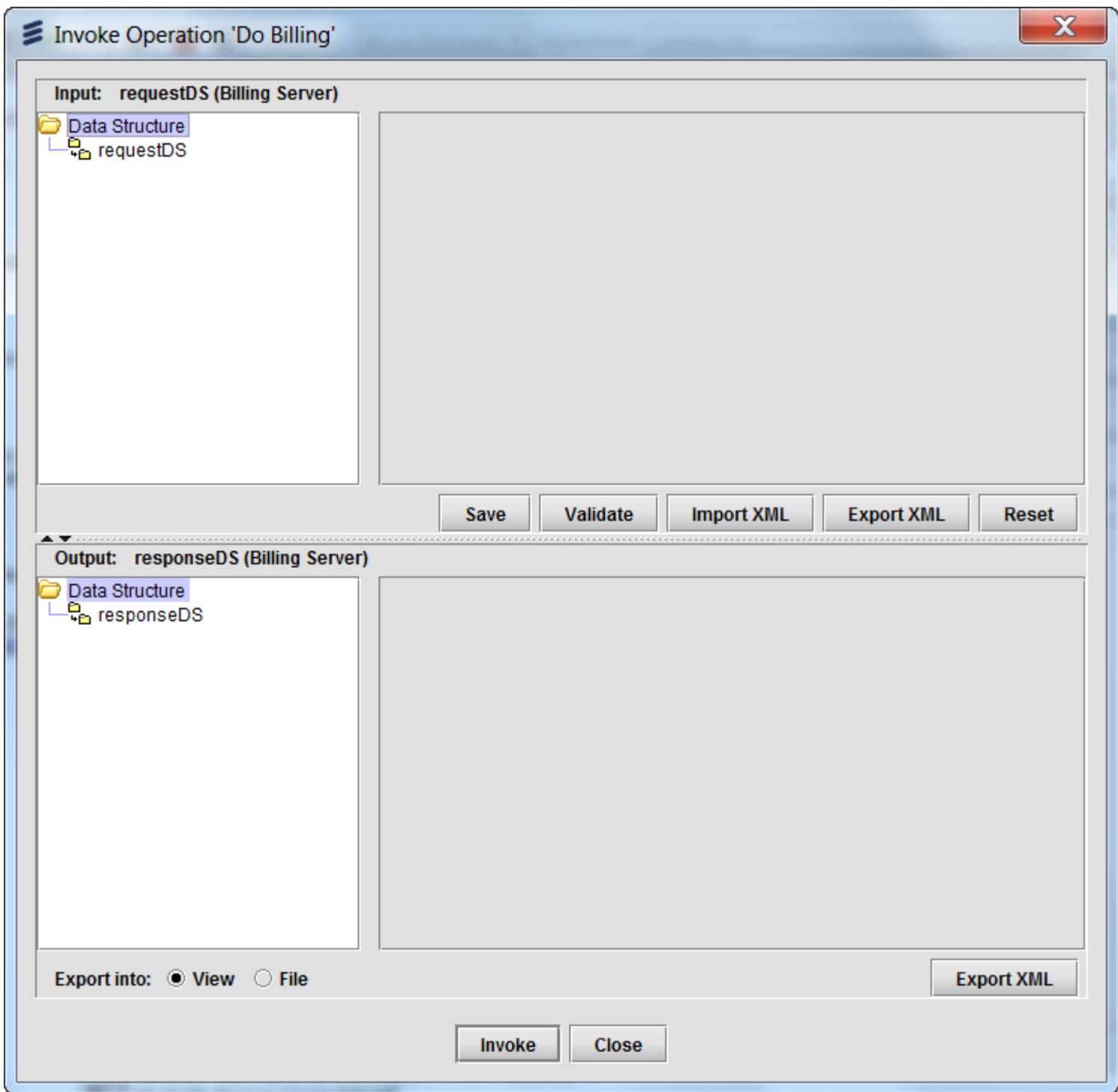
- The Operation names are defined from the viewpoint of the Client (for example, One-Way is an outgoing message, Notification is an incoming message, Request-Response is a send-wait-receive message).
- It is recommended that you use ANON\_ROBUST\_OUT\_ONLY\_OP to create the SOAP service client when the

	interface operation is one-way, but has fault defined in the metadata.
<b>Privileges</b>	<p>When the button is clicked at the end of the field, the <b>Select Privileges</b> dialog opens, allowing the user to select a combination of user privileges.</p> <p>Privileges in an Interface Operation definition specify who can use this Operation. This definition overrides privileges defined in the Interface. If privileges are not defined, the privileges in the Interface definition are used instead. Currently, privileges are enforced only for API users (external systems using the framework's services).</p>

## Testing an Interface Operation

An individual Interface Operation with a *Request-Response* or *One-Way Operation type* has an **Invoke Operation** item in its right-click pop-up menu in addition to the common right-click menu items. When selected, this menu item opens the **Invoke Operation** dialog that allows the Interface Operation to be invoked with specified criteria. This option can be used to test the Interface Operation.

The following example allows you to fill in the **customerName** field in the Input section. Click the **Invoke** button to see the results in the Output section. In this example, the **successFlag** field contains the correct input data with a confirmation message.



**Notes:**

- Refer to section on Finder Operations for a description of a similar dialog, the [Invoke Finder](#) dialog.
- The `Global.invokeInterfaceUsePE()` can be used to invoke the callback script in interface operations. Refer to [JavaScript documentation](#) for more information.
- Applications can access certain information defined in participant and participant operations from script. See the `MetadataParticipant` and `MetadataParticipantOperation` classes in the JavaScript documentation for details.

## Defining a Binding

A Binding provides concrete protocol and data format specifications for a particular Interface.

To create a new Binding, do one of the following:

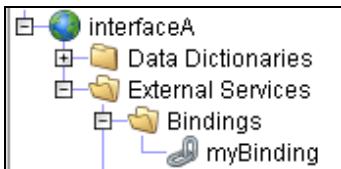
In the **Metadata** tab of **Navigation** pane, either:

1. Right-click a Namespace and select **New ...**
2. **New Metadata Object** wizard appears as a popup. Expand **External Services**, select **Binding**, and then click the **Next** button.
3. Step two of the wizard appears. Enter the name of the Binding (must conform to JavaScript naming conventions), and populate other fields, as necessary. Then, click the **Finish** button.

OR

1. Right-click the **Bindings** folder or the **External Services** folder in a Namespace, if it is present. Select **New Binding**.
2. Follow step 3 from the previous procedure.

After the Binding is created, the icon appears under the **Bindings** folder.



### Binding General Properties

The general Binding properties are defined on the **General** tab (see figure below). The table describes the fields.

*General tab of the Binding properties.*

<b>Name:</b>	myBinding	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated				
<b>Label:</b>	myBinding					
<b>Description:</b>	<input type="text"/>					
<b>Interface:</b>	interfaceA.creditAssure	<input style="width: 20px; height: 20px; vertical-align: middle;" type="button" value="..."/>				
<b>Provider:</b>	cwf.soapprov (SOAP)	<input style="width: 20px; height: 20px; vertical-align: middle;" type="button" value="..."/>				
<b>Interface elements:</b>						
<b>Element definitions:</b>						
<table border="1"><thead><tr><th>Attribute</th><th>Value</th></tr></thead><tbody><tr><td colspan="2"> </td></tr></tbody></table>			Attribute	Value		
Attribute	Value					
<input type="button" value="Change Values"/>						

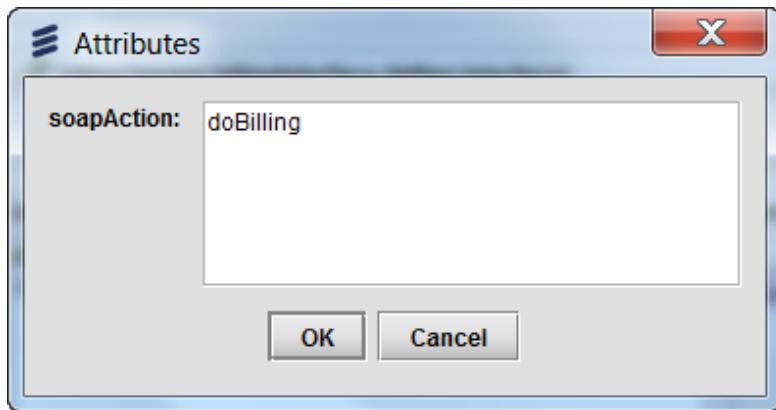
[Description of fields in the General tab of the Binding properties.](#)

Field	Description
-------	-------------

<b>Name</b>	Mandatory. Name of the Binding within the namespace.
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Label</b>	Mandatory. Label to identify the Binding.
<b>Description</b>	Description of the Binding for documentation purposes.
<b>Interface</b>	The Interface that the Binding is for.
<b>Provider</b>	The service provider that will interact with the service. The service provider selection dictates what kind of Binding details are required. The <b>Provider</b> combo box includes both <a href="#">built-in service providers</a> .
<b>Interface elements</b>	A tree showing the Interface that the Binding is for and its related Operations and message elements. The Interface Operation contain the following message elements if they were defined on the <b>General</b> tab: <ul style="list-style-type: none"> <li>• Input</li> <li>• Output</li> <li>• Fault</li> </ul>
<b>Element definitions</b>	When an Interface, Operation or message element is selected in the <b>Interface element</b> tree, the attributes for the element appear in the table where values can be provided.

All necessary information required for successful interaction with the service must be defined by the Binding details. Binding details are provided by selecting one of the Interface Operation elements in the **Interface elements** tree, and providing values for the element definitions in the **Element definitions** table.

When an attribute is selected in the **Element definitions** table, click the **Change Values** button to add or update the attribute value. The **Attributes** dialog opens showing the appropriate fields depending on the attribute type.



**Note:** Binding details that are required are service provider specific (the Binding details are specified by the service provider class). Refer to the corresponding section in [Built-in Service Providers](#) for specific information on Binding attributes and the values that each service provider supports.

There are more Binding settings of the external service to be configured in [Configuration application's Service page](#). These settings are instance-specific and thus to be configured in the Configuration application only, and not available to be configured in the metadata within Velocity Studio.

## Defining a Service

A Service represents a collection of related Ports.

To create a new Service, do one of the following:

In the **Metadata** tab of **Navigation** pane, either:

1. Right-click a Namespace and select **New ...**
2. **New Metadata Object** wizard appears as a popup. Expand **External Services**, select **External Service**, and then click the **Next** button.
3. Step two of the wizard appears. Enter the name of Service (must conform to JavaScript naming conventions), and populate other fields, as necessary. Then, click the **Finish** button.

OR

1. Right-click the **External Services** folder in a Namespace, if it is present. Select **New External Service**.
2. Follow step 3 from the previous procedure.

After the Service is created, the icon will be added under the **External Services** folder.



### Service General Properties

The general Service properties are defined on the **General** tab (see figure below). The table below describes the fields.

*General tab of the Service properties.*

<b>Name:</b>	apiService	<input type="checkbox"/> Private	<input type="checkbox"/> Restricted	<input type="checkbox"/> Deprecated
<b>Label:</b>	apiService			
<b>Description:</b>	<input type="button" value="Edit"/>			
<b>WSDL Type:</b>	WSDL 1.1	<input type="button" value="▼"/>		

Description of fields in the General tab of the Service properties.

Field	Description
<b>Name</b>	Mandatory. Name of the Service within the namespace.
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Label</b>	Mandatory. Label to identify the Service.
<b>Description</b>	Description of the Service for documentation purposes.

<b>WSDL Type</b>	Choose between <b>WSDL 1.1</b> and <b>WSDL 2.0</b> . This selection is only relevant for SOAP bindings -- that is, when any Port of this service has a Binding that uses the SOAP service provider.
------------------	---

## Defining a Port

A port represents a single endpoint defined as a combination of a Binding and a network address (location). A port can be added directly under an External Service element.

To create a new port, do the following:

1. Right-click the **External Services** folder in a Namespace and select **New Port**.
2. The New Port dialog appears. Enter the **Name** of the port (must conform to JavaScript naming conventions), select the **Binding**, and populate other fields, as necessary.
3. Click the **Finish** button.

After you have created your port, its icon appears in the **External Services** folder, under your External Service element.



### Port General Properties

The port's **General** tab defines general port properties.

Name:	apiPort
Label:	apiPort
Description:	(Text input field with a pencil icon)
<input type="checkbox"/> Listener <input type="checkbox"/> Message on fault	
Calendar:	<NONE> (dropdown with globe icon)
Binding:	testNamespace.apiBinding (dropdown with globe icon)

The following table describes the property fields in the port's General tab:

Field	Description
<b>Name</b>	Mandatory. Name of the port within the namespace.
<b>Label</b>	Mandatory. Label to identify the port.
<b>Description</b>	Description of the port for documentation purposes.
<b>Listener</b>	If checked, this port is a listener port. This option will be enabled only when the service provider supports listening. Selection of this option will make the <b>Backup port</b> field invisible in the User Interface and their settings will be ignored by the runtime system. Selection of this option will also replace the calendar field with the Script field.
When you define a port as a listener, you can create cwOnApixxxError methods to handle exceptions for error processing:	
<ul style="list-style-type: none"><li>• If your API script on a listener port has a try-catch and does not have a rethrow exception, the predefined onxxxerror does not get invoked.</li><li>• There is no onxxxerror method that is a catch-all.</li><li>• If you define a new method, it does not get processed. Only the following methods are processed:<ul style="list-style-type: none"><li>◦ cwOnApiConnectionError</li><li>◦ cwOnApiBusinessError</li><li>◦ cwOnApiExternalError</li><li>◦ cwOnApiSecurityError</li><li>◦ cwOnApiInternalError</li><li>◦ cwOnApiUnknownError</li><li>◦ cwCategorizeApiError</li></ul></li></ul>	
<b>Message</b>	This property appears under all these conditions:

<b>on fault</b>	<ul style="list-style-type: none"> <li>The <b>Listener</b> property is unchecked</li> <li>The <b>Binding</b> field is not empty</li> <li>The binding uses either the SOAP or HTTP service provider</li> </ul> <p>If an <i>AxisFault</i> is received from the SOAP service, the framework will try to populate its data into the data structure specified in <b>Fault</b> of the interface operation definition. When unchecked, an <i>AxisFault</i> will throw a CwfException and the error stack trace will be in the Event Log (the stack trace is limited to 1024 characters).</p> <p>When <b>Message on fault</b> is enabled on a SOAP external interface, if an operation is at fault, the Fault data structure defined returns a fault message.</p>
<b>Export schema inside wsdl</b>	This property appears only when you have selected the <b>Listener</b> property. By default, the <b>Export schema inside wsdl</b> property is unchecked. When selected, this property exposes a flat WSDL (that is, with embedded schemas) for a SOAP interface. When you access the WSDL from the Web, a list of schemas displays.
<b>Calendar</b>	Specifies an optional calendar that is used to determine service availability. If no calendar is specified in the <b>Calendar</b> field, the service is considered available at all times. When a calendar is specified in the <b>Calendar</b> field, the service is considered available only during the <i>working</i> slots of the calendar. When the framework needs to perform a particular Interface Operation, it selects the first port that implements the Interface and that is currently available.  <b>Note:</b> <a href="#">Calendars</a> are defined and managed in the System Administration application, and the enumeration of these selections in Velocity Studio are defined by the <i>cwf.allCalendars</i> Data Type.
<b>Script</b>	Appears when listener is checked. Allows for specification of a script to be used to process the request. Provided as an alternative to specifying a separate Javascript port and binding. Two arguments are passed to the script: Argument 1: input: the data received by the listener Argument 2: operation: the name of the interface operation handling the request.
<b>Binding</b>	Specifies the port binding that determines the Service Provider and all concrete details necessary to interact with the service.
<b>URL Mapping</b>	<p>This field appears when you select the <b>Listener</b> field. This field can only be set for HTTP listener ports and is validated the same way as a user interface application URL mapping. The value specified in this field represents a URL path that is to be mapped to the port (listener). This value replaces the current hard-coded interface name in the URL.</p> <p>The URL path can contain multiple path items. However, it must begin with / (for example, /portA). The URL path supports a wildcard (*). However, the wildcard character must be at the end and is only supported as the full path element. As an example, /a/b/* is valid, but /a/b* is not. If a wildcard is not specified, the request's full URL path must match the value.</p>

There are more port settings of the external service to be configured in [Configuration application's Service page](#). These settings are instance-specific and thus to be configured in Configuration application only, and not available to be configured in the metadata or Velocity Studio. Furthermore, some of these settings in the Configuration application, such as *Address attributes*, are service provider-specific (the port properties are as per the service provider class). Refer to the corresponding section in [Built-in Service Providers](#) for specific information on port attributes and the values that each service provider supports.

## Bind Multiple Ports to the Same Interface Using the Same Binding

In External Services, you can create multiple ports that are bound to the same interface, to easily copy and create backup ports.



This feature is useful when you want to have backup ports.

## Service Robustness

### Handling Service Invocation Errors

Every port has a *retry count* attribute that determines how many times the framework will try to use this port if service invocation has failed. The framework maintains an error count for consecutive failed invocations. Every successful service invocation resets this error counter. When the counter value passes the retry count attribute value (equal to retry count + 1), the port is marked as in error status (the service on this port is unavailable).

The framework will stop using ports in error status if a backup port that implements the same interface exists. A backup port is a port that has a *backup* attribute set to *true*.

If the framework switches to a backup port, an attempt will be made to recover the original (production) port according to the *recover after* minutes setting. *Recover after* is an attribute of the port. This allows the framework to automatically switch back to the proper port when the service on that port becomes available.

### Determining Service and Port Availability

The framework automatically selects the appropriate port for service invocation and uses the following algorithm:

- Finds a port that is not in error status (see above), and is currently available based on the port calendar definition. If such a port is available, it will use this port.
- Finds a port that is in error status (see above) and is currently available based on the port calendar definition.
- If this port's *recover after* time has passed (number of minutes since last use), it will use this port.
- If this port's *recover after* time has not passed (number of minutes since last use) and there is a backup port available, use the backup port.
- Otherwise use this port.
- If none of the above is found, return no port.

If the service invocation request was initiated from the OMS UI, and no available port was found, the framework will present an error to the user.

The process management framework can automatically delay a service request (when no available port is found) for processes that communicate with external services.

## Type Mapping with External Services

XSD Type	XML	Base Type	Base Type Restriction				Java	COM	CORBA	Oracle
			Length	Min	Max	Pattern				
string		String	inherited	NA	NA		String	VT_BSTR	tk_string	VARCHAR2 (n) - see 1
byte		Decimal	inherited	-128	127	[^-]? [0-9]+	Byte	VT_I1	tk_char	NUMERIC (3)
unsignedByte		Integer	inherited	0	255	[^-]? [0-9]+	Short	VT_UI1	tk_short	NUMERIC (3)
integer		Integer	inherited	inherited	inherited	[^-]? [0-9]+	BigInteger	VT_BSTR	tk_string	NUMERIC (n) - see 2
positiveInteger		Integer	inherited	1	inherited	[^-]? [0-9]+	BigInteger	VT_BSTR	tk_string	NUMERIC (n) - see 2
negativeInteger		Integer	inherited	inherited	-1	[^-]? [0-9]+	BigInteger	VT_BSTR	tk_string	NUMERIC (n) - see 2
nonNegativeInteger		Integer	inherited	0	inherited	[^-]? [0-9]+	BigInteger	VT_BSTR	tk_string	NUMERIC (n) - see 2
nonPositiveInteger		Integer	inherited	inherited	0	[^-]? [0-9]+	BigInteger	VT_BSTR	tk_string	NUMERIC (n) - see 2
int		Integer	inherited	-2147483648	2147483647	[^-]? [0-9]+	Integer	VT_I4	tk_long	NUMERIC (10)
unsignedInt		Integer	inherited	0	4294967295	[^-]? [0-9]+	Long	VT_UI4	tk_ulong	NUMERIC (10)
long		Integer	inherited	- 9223372036854775808	9223372036854775807	[^-]? [0-9]+	Long	VT_I8	tk_longlong	NUMERIC (20)
unsignedLong		Integer	inherited	0	18446744073709551615	[^-]? [0-9]+	BigInteger	VT_UI8	tk_ulonglong	NUMERIC (20)
short		Integer	inherited	-32768	32767	[^-]? [0-9]+	Short	VT_I2	tk_short	NUMERIC (5)
unsignedShort		Integer	inherited	0	65535	[^-]? [0-9]+	Integer	VT_UI2	tk_ushort	NUMERIC (5)
decimal		Decimal	inherited	inherited	inherited		BigDecimal	VT_BSTR	tk_fixed	NUMERIC (n,s) - see 2
float		Decimal	inherited	inherited	inherited		Float	VT_R4	tk_float	FLOAT
double		Decimal	inherited	inherited	inherited		Double	VT_R8	tk_double	NUMERIC
boolean	0,false,1,true	Boolean	NA	NA	NA		Boolean	VT_BOOL	tk_boolean	NUMERIC (1)
time	hh:mm:ss	Boolean	NA	NA	NA		Date	VT_DATE	tk_string	DATE
dateTime	CCYY-MM-DDThh:mm:ss	DateTime	NA	NA	NA		Date	VT_DATE	tk_string	DATE
duration		String	inherited	NA	NA		String	VT_BSTR	tk_string	DATE
date	CCYY-MM-DD	Date	NA	NA	NA		Date	VT_DATE	tk_string	DATE
gMonth	--MM--	DateTime	NA	NA	NA		Date	VT_DATE	tk_string	DATE
gYear	CCYY	DateTime	NA	NA	NA		Date	VT_DATE	tk_string	DATE
gYearMonth	CCYY-MM	DateTime	NA	NA	NA		Date	VT_DATE	tk_string	DATE
gDay	--DD	DateTime	NA	NA	NA		Date	VT_DATE	tk_string	DATE
gMonthDay	--MM-DD	DateTime	NA	NA	NA		Date	VT_DATE	tk_string	DATE
anyUri		String	inherited	NA	NA		String	VT_BSTR	tk_string	VARCHAR2 (n) - see 1

**Note:**

1. n <= 4000
2. n <= 38

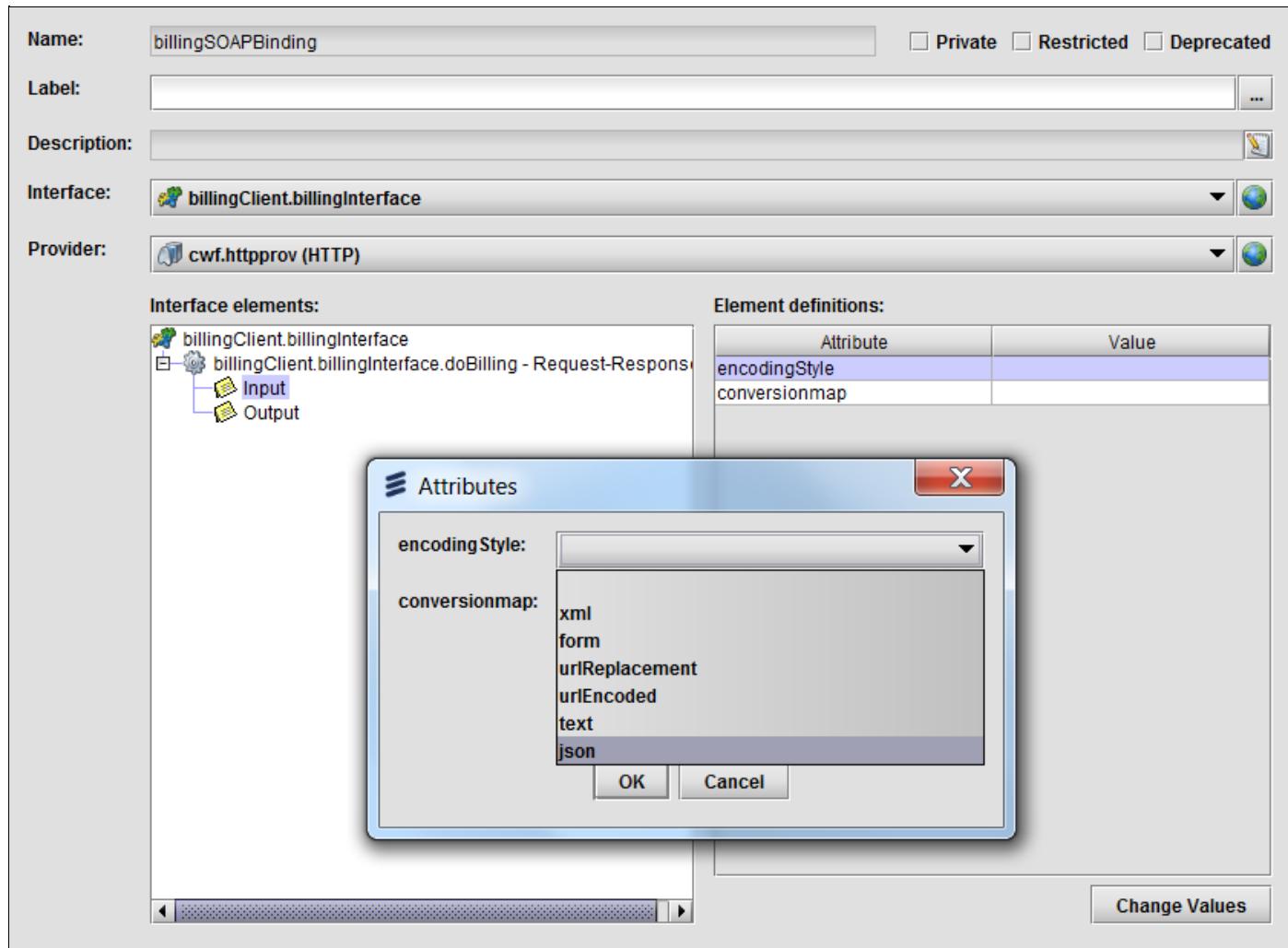
## JSON and REST Support

**JavaScript Object Notation (JSON)** is a lightweight data serialization format based on a subset of JavaScript. A Representational State Transfer (REST) interface is one that conforms to the constraints and characteristics of the [REST architectural style](#) for distributed hypermedia systems.

The following support is available:

- JSON for both the HTTP service provider and HTTP listener
- REST for the HTTP service provider

You can select **JSON** from the **Encoding Style** field for HTTP messages:



The **Encoding Style** field specifies how message data is formatted when performing a request. The system performs an automatic conversion between JSON and the metadata object. As a result, the metadata object's attributes must match the property names of the JSON object.

For example, suppose your JSON object is the following:

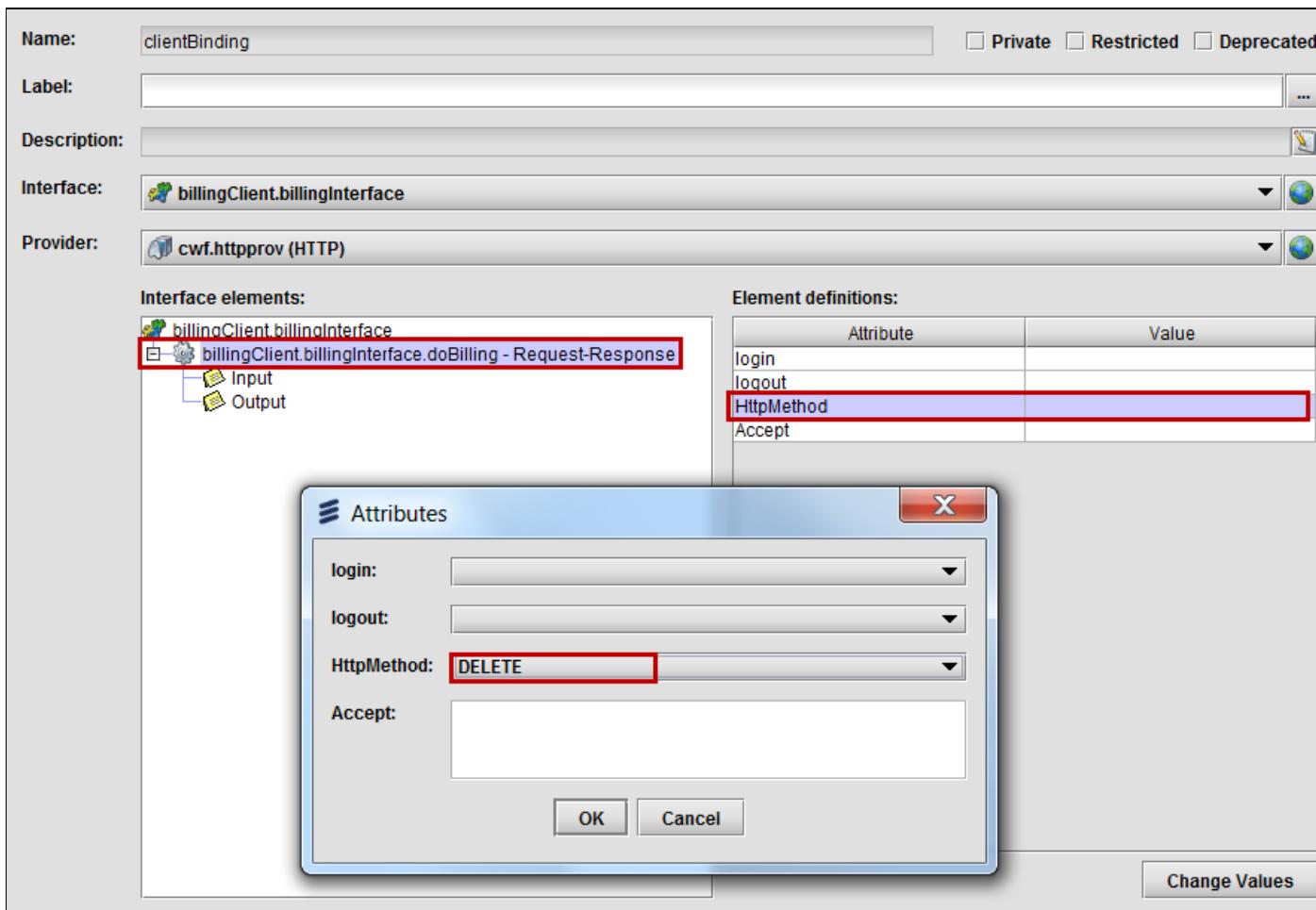
```
{  
    "loginName" : "email/ phone number",  
    "password" : "",  
    ...  
}
```

To use the auto-conversion, the BusinessPartner metadata document needs to have leaves named *loginName* and *password*.

The **HTTP Method** attribute appears in the HTTP Service Provider operation level, and has the following options to send out requests:

- **POST** (create)
- **GET** (read)
- **PUT** (update or create)

- **DELETE** (delete)
- **HEAD** (to read header information only)



This attribute is used as HttpURLConnection's property to send out a request, such as the following:

```
connection.setRequestMethod( "POST" );
```

If there is no **HTTP Method** attribute specified, use the following:

- Use **POST** for the following **Encoding Style** values:
  - XML
  - Form
  - Text
  - Json
- Use **GET** for the following **Encoding Style** values:
  - UrlReplacement
  - UrlEncoded

## APIs

The following APIs are available:

- Global.getItemsFromJson(), which converts a JSON string into a list of data objects.
- These methods in DataObject (implemented in both DataStructure and DataDocument):
  - fromJson, which performs a conversion from a JSON object to a data object.
  - toJson, which converts a dataObject to a JSON string.
- DataStructure.toText(), which must be overridden.
- DataStructure.fromText(), which must be overridden.

Velocity Studio exposes these methods, which allows you to override them:

API Method	Velocity Studio Method
fromJson	cwFromJson

toJson	cwToJson
toText	cwToText
fromText	cwFromText

#### Notes:

- toJson can have the excludeHeader flag, which is false by default. You can set it to true using toJson(true) to exclude either the datastructure or document name in the JSON string.
- toJson ignores elements that have the **Exclude** property set.
- XML rules that you define at the object level (for example, document, datastructure, or order) are invalid for the toJson method.
- You can exclude the header from a JSON string by using the fromJson method's excludeHeader parameter, which takes a Boolean value. This method uses the metadata setting if this parameter is missing. The method first does matching with the XML name for backward compatibility, then the JSON name, and then the local name. This method throws an error when the JSON header is expected, but the given string does not contain a header.
- To detect whether a data structure has been changed after a fromJson(call), use the isLeafSet("name") method.
- You can use the cleanLeafSetFlag("name") and cleanLeavesSetFlag() methods to clean the SET flag on a variable and on all variables, respectively.

#### Placeholders in the URL to Support REST Service

The URL for the httpservice location has placeholders, such as `http://url/api/v3/clients/{client_id}/{client_variant_id}/businessPartners`. The `client_id` and `client_variant_id` placeholders are the leaf names in either the document or data structure.

At runtime, these placeholders are populated with the values from the document or data structure to get the actual URL location for the interface service.

For example, if the incoming document has `client_id` containing a value=2012, and `client_variant_id` with a value=0102, the URL becomes `http://url/api/v3/clients/2012/0102/businessPartners`.

#### MessageLog Finder

To support displaying JSON or text in sent or receive receive data, only XML is allowed. You can also resend a JSON text.

#### JSON Date and Time Format

Support is available for a flexible JSON date format using the **JSON Format** field, which includes support for the /Date(milliseconds) format and any valid Java date, such as `yyyy-mm-dd`.

General Methods Enumeration

Name:	TestDataType	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated <input type="checkbox"/> Final				
Label:	TestDataType	...				
Description:						
Help:						
Extends:	com.conceptwave.system.DateTime	<input checked="" type="checkbox"/> Nullable <input type="checkbox"/> Encrypt				
Overrides:	<NONE>					
Length:	<Unlimited>(<Inherited>)					
Default value:						
XML name:	Pattern:					
Text format:	Text length:	<b>Json format: /Date</b>				
Element properties:	<NONE>					
<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td colspan="2"> </td> </tr> </tbody> </table>			Name	Value		
Name	Value					

### Parsing from JSON

Parsing from JSON supports `/Date()`. If no format is specified, assuming and it is a datetime type, the ISO-8601 standard is assumed, and either `YYYY-MM-DDTHH:mm:ss.SSSZ` or `yyyy-MM-dd'THH:mm:ss.SSS` format is assumed. You can choose whether the time should be in UTC (for example, `2012-06-01T03:04:05.123Z`) or in a specific timezone (for example, `2012-06-01T03:04:05.123+0200`). For a date type, the ISO-8601 standard is assumed and takes the `yyyy-MM-dd` format.

### Parsing to JSON

If no format is specified, the GMT or UTC time in is converted in ISO-8601 format `YYYY-MM-DDTHH:mm:ss.SSSZ` (for example, `2012-06-01T03:04:05.123Z`). If the format is `/Date`, it would be converted into `/Date(milliseconds)`. Otherwise, the system performs regular Java date formatting.

If there is no JSON format specified, and it is a datetime type, ISO-8601 standard format, `yyyy-MM-ddTHH:mm:ss.SSSZ` or `yyyy-MM-dd'THH:mm:ss.SSS`, is assumed. For a date type, the ISO-8601 standard format, `yyyy-MM-dd`, is assumed. For example, if you have a date string, `2012-02-02`, and its type is Date and no JSON format specified, parsing occurs to get the value.

### Support to Get Request URL in Listeners

Support is available for REST over HTTP, which allows you to get the request URL (parameter or value) in listeners.

The existing `UserProfile.getRequestParamNames()` and `UserProfile.getRequestParam()` methods can be used in HTTP listener scripts to access URL parameters. This feature is for URL parameters that can exist in a POST, GET, and so on.

**Note:** This feature is not for HTTP headers.

### Access to Response BODY and HTTP Header

When an error message appears in the BODY of the HTTP response, you have access to this error message when using HTTP and REST service providers.

To access this error message, verify the following:

- Select the **Message on fault** option for this HTTP port.
- Ensure that the fault data element for the specified operation matches the incoming response body.

An array containing the fault data object is returned as a result of this interface call. The response body is also logged as received data in the message log.

Additionally, the response header information is logged as received properties in the message log, which is also available using `UserProfile.getHttpResponseHeaderMap()`.

### Support for REST HTTP Verbs

The `UserProfile.getHttpMethod()` allows the access to the REST http verb used in the request.

The `UserProfile.getRequestPath()` can be used to work with the URL paths. HTTP Listener URL paths contain namespace, interface and the operation information. URL information beyond the standard HTTP Listener URL pattern is considered URL path and can be used to develop resource based REST services.

### URL Path Support for REST

The `getRequestPath()` method enhances the REST functionality by allowing you to work with URL paths. Currently, HTTP listener URL paths contain namespace, interface, and operation information:

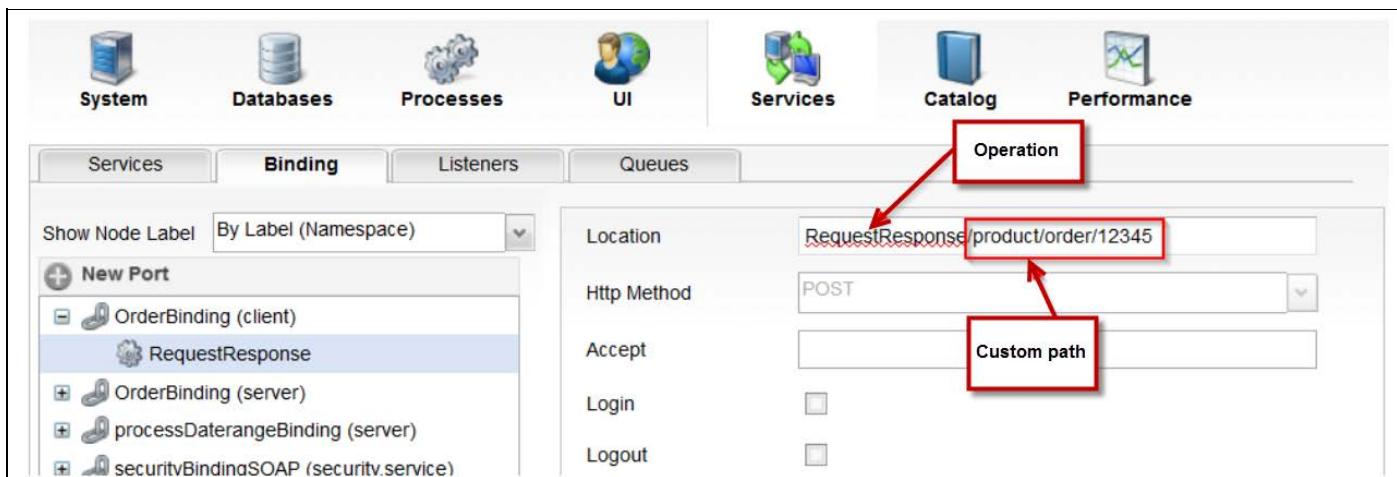
```
http://host:port/cwf/api/<namespaceName>-<interfaceName>/<operationName>
```

URL information beyond the standard HTTP listener URL pattern is considered to be a URL path and can be used to develop resource-based REST services. The following is a sample URL path:

```
http://www.server.com/cwf/api/server-OrderInterface/ResponseRequest/resource_category/resource/resource_id
```

You can access a custom URL path by completing these steps:

1. In the Configuration application, click **Services > Binding** and click the operation that you want.
2. In the operation's **Location** field, add a custom path.



The internal URL parser considers any path element after the operation (that is, first element) to be customized. As a result, it is made available in the script.

3. In the listener script, use the `UserProfile.getRequestPath()` method to get an array of strings that represents the URL path:

```
Global.logDebug("Request URI "+UserProfile.getRequestPath());
```

**Note:** If you do not specify an operation in the URL path, it defaults to the first interface operation.

## User Interface Contributor

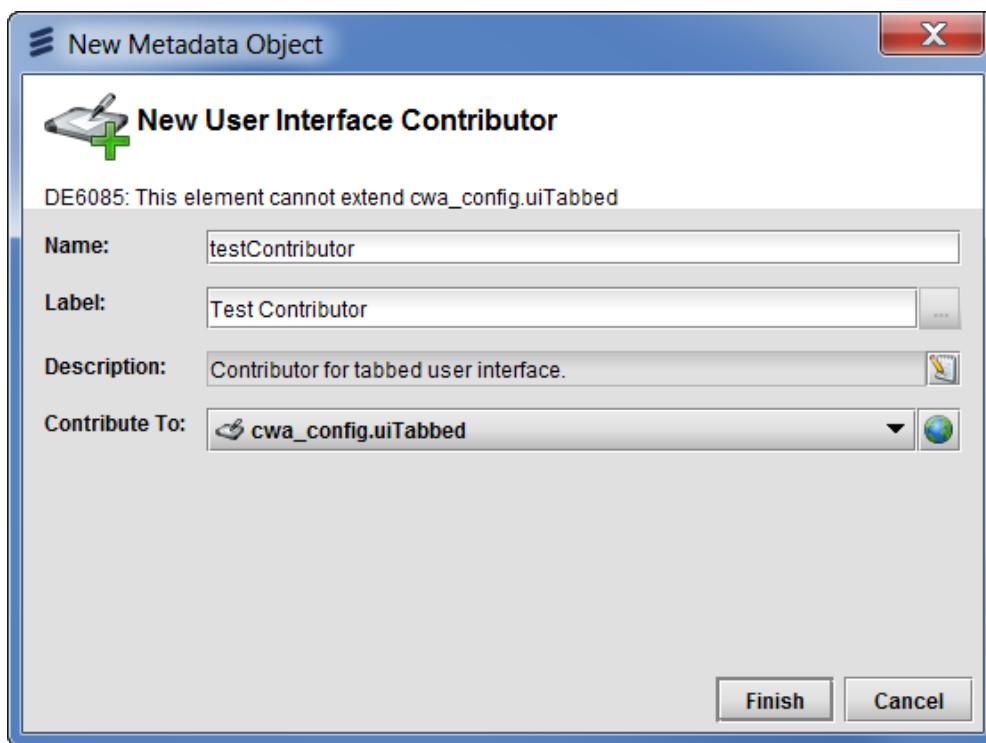
User Interface (UI) Contributor is a metadata object that can be used to add new information to user interface forms from multiple sources. The UI Contributor has the ability to pick the latest inherited version of forms. With the UI Contributor, changes to user interface forms are same as changes to overridden forms. The UI Contributor is another type of User Interface Controller.

### Create a UI Contributor

To create a new UI Contributor, do one of the following options:

In the **Metadata** tab of **Navigation** pane, either:

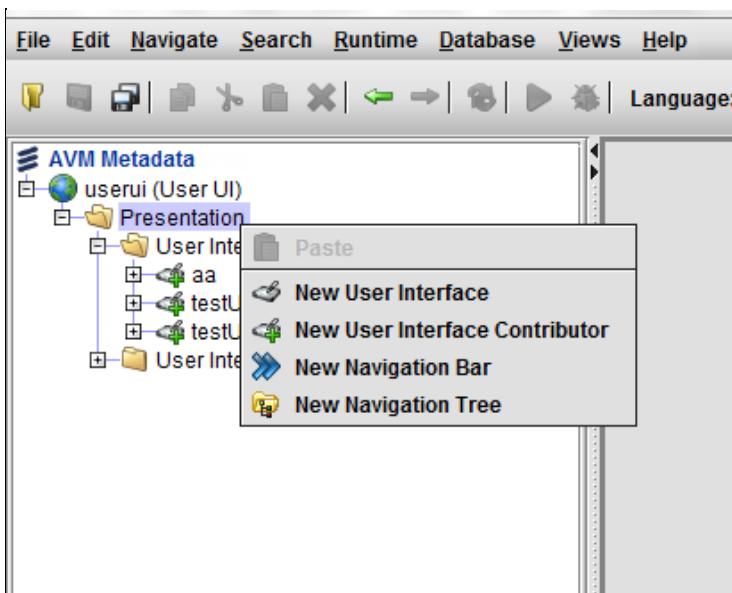
1. Right-click a **Namespace** and select **New**.
2. **New Metadata Object** wizard appears as a popup. Expand **Presentation**, select **User Interface Contributor**, and then click the **Next** button.
3. Step two of the wizard appears. Enter the **Name** of the **UI Contributor**. Enter the **Label** for the **UI Contributor**.
4. Select the UI Controller for the **Contribute To** field; UI Contributor must contribute to a valid UI Controller.
5. Click the **Finish** button.



OR

1. Right-click the **Presentation** folder in a Namespace, if it is present. Select **New User Interface Contributor**.
2. Follow step 3 from the previous procedure.

**Note:** The selected UI Controller in the **Contribute To** field is referred as the **Contributee**, whether it is a top-level controller or a non-top level controller.



## Usage Scope of UI Contributor

Once the UI Contributor is created, it can:

- Override the **Menu** Form.
- Override or extend **Menu Elements**.
- Add new **Menu Elements**.
- Add new local variables and local methods.

## Limitations of UI Contributor

The usage of the UI Contributor is limited; consider the following points, when working with UI Contributor:

- A UI Contributor cannot add new forms to the Contributee.
- A UI Contributor can only contribute to the **Menu** Form.
- A UI Contributor cannot override, extend, or add any of the base elements other than **Menu Elements**.
- A UI Contributor cannot override the Contributee's original methods.
- At design time, a controller **A** extending controller **B** does not have access to any methods or variables that comes from contributors of controller **B**.
- Local variables and methods of contributors can be used only by the UI elements added by the Contributors and cannot be expected to be usable for other purposes.

## Conflicts

Contributors of the same Contributee cannot have methods or variables with the same name among themselves. If conflicting contributors exist in a project, either resolve conflicts by renaming UI elements, variables, methods, or disable a contributor.

To disable a contributor, follow these steps:

1. Click on the metadata header and select the **Contributors** tab.
2. From the list, either check or uncheck the **Enabled** field to select which contributor to use or not.

The screenshot shows the AVM Metadata interface with the 'Contributors' tab selected. On the left, the tree view shows 'Test (Test UI Contributor)' with its structure. On the right, the 'Contributors' table lists four entries:

Enabled	Contributor	Contributee
<input checked="" type="checkbox"/>	Test.customer	ui_common.favoriteResult
<input type="checkbox"/>	Test.customerUIContributor	ui_common.favoriteResult
<input checked="" type="checkbox"/>	Test.testUIContributor	ui_common.favoriteResult
<input checked="" type="checkbox"/>	Test.testUIContributor1	ui_common.favoriteResult

## Design-Time Validation Errors

When working with the UI Contributors, there are couple of points that can be considered to avoid the validation errors:

- If different UI Contributors have variables or methods with same name and they contribute to the same UI Controller or UI, then it results in a design time validation error.

Object	Description
	Errors(11 items)
	DE6106: This contributor has method or variable "{0}" that conflicts with another contributor of the same UI Controller
	DE6106: This contributor has method or variable "clickSave" that conflicts with another contributor of the same UI Controller
	DE6106: This contributor has method or variable "{0}" under Overlay "[1]" that conflicts with another contributor of the same UI Controller
	DE6107: This contributor has element "{0}" unresolved.
	DE6106: This contributor has method or variable "clickSave" that conflicts with another contributor of the same UI Controller Rename the method or variable, or disable this contributor or the other contributor(s) in the metadata header
	Warnings(28 items)
	DW6035: "{0}" is a reserved java key word.

- If different UI Contributors modify the same menu element, or create menu elements with the same name and they contribute to the same UI Controller or UI, then it results in a design time validation error.

Object	Description
	Errors(11 items)
	DE6106: This contributor has method or variable "{0}" that conflicts with another contributor of the same UI Controller
	DE6107: This contributor has element "{0}" under Overlay "[1]" that conflicts with another contributor of the same UI Controller
	DE6107: This contributor has element "contactCustomer" under Overlay "Menu" that conflicts with another contributor of the same UI Controller
	DE6107: This contributor has element "contactCustomer" under Overlay "Menu" that conflicts with another contributor of the same UI Controller
	DE9200: "{0}" Unresolved.
	Warnings(28 items)
	DW6035: "{0}" is a reserved java key word.
	DW6036: "{0}" is a reserved java script key word.
	DE6107: This contributor has element "contactCustomer" under Overlay "Menu" that conflicts with another contributor of the same UI Controller Rename the method or variable, or disable this contributor or the other contributor(s) in the metadata header

## User Interface Contributor General Properties

The general properties of User Interface (UI) Contributor are defined on the **General** tab. The following image displays the general tab of the UI Contributor.

The screenshot shows the 'General' tab of a UI Contributor configuration. The 'Name' field contains 'testUIContributor'. The 'Label' field contains 'Test UI Contributor'. The 'Description' field is empty. The 'Contribute To' dropdown menu is open, showing 'ui\_common.favoriteResult (Custom Views)' as the selected item. Checkboxes for 'Private', 'Restricted', and 'Deprecated' are present but not checked. There are also standard UI controls like a toolbar icon and a help button.

The following table describes the fields for **General** tab of user interface contributor:

Field	Description
<b>Name</b>	Mandatory. Name of the metadata object (used in scripts; must conform to JavaScript naming conventions).
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Label</b>	Mandatory. Display label for the UI Contributor.
<b>Description</b>	Description of the metadata object for documentation.
<b>Contribute To</b>	Allows to select the User Interface or UI Controller for UI Contributor.

## User Interface Contributor Variable Properties

The **Variables** tab displays default (also known as base), overridden, and new variables for the User Interface (UI) Contributor. For example in the following image, variables highlighted in color blue are default, and the variables highlighted in color black are overridden. The default variables depends on which UI Controller is selected in the **General** tab of the UI Contributor in the **Contribute To** field and which default user interface is extended by the selected UI controller. More variables can be added when you select the default variable from the list and click button.

Name	Type	Methods	Vis	Opt	Edit
cwShowHover	com.conceptwave.system.Boolean(Boolean)	<input checked="" type="checkbox"/>			
confirmObject	com.conceptwave.system.Object(Object)	<input checked="" type="checkbox"/>			
model	com.conceptwave.system.Model(Model)	<input checked="" type="checkbox"/>			
chgPwdDoc	cwf_up.chgPwdDoc(Document)	<input type="checkbox"/>			
access	config.data.access(Integer)	<input checked="" type="checkbox"/>			
YEAR	cwf_up.YEAR(Decimal, 4)	<input checked="" type="checkbox"/>			
model1	com.conceptwave.system.Model(Model)	<input type="checkbox"/>			
model2	com.conceptwave.system.Model(Model)	<input type="checkbox"/>			

General		Methods					
Name:	YEAR	<input type="checkbox"/> Constant	<input type="checkbox"/> Audit				
Description:	<input type="text"/>						
Data type:	<input type="button" value="cwf_up.YEAR"/> <span style="font-size: small;">▼</span> <span style="font-size: small;">🌐</span>						
Type error code:	<input type="text"/>	Mandatory error code:	<input type="text"/>				
Encrypt:	<input type="button" value="No"/> <span style="font-size: small;">▼</span>						
Element properties:	<input type="button" value="Inherited"/> <span style="font-size: small;">▼</span>						
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> </tr> </tbody> </table>				Name	Value		
Name	Value						

The UI Contributor variable details appear in the **Detail** pane when a variable entry is selected. The UI Contributor variable properties are listed in the [property table](#). Some of the properties appear in the columns of the **Variables** table.

- The **Name** column displays the name of the variables either system generated, default variables or the overridden.
- The **Type** column is populated with the primitive data type of the variable's selected data type (for example, String), its actual data type (for example, com.conceptwave.system.String), and its length, if it is specified.
- The **Methods** check box is checked whenever the UI Contributor variable contains one or more methods.

The **Vis**, **Opt** and **Edit** fields denote the **Visible**, **Optional**, and **Editable** element properties, respectively. A column of action icons exists at the right of the table. The following table describes their functionalities:

Icon	Description
	Move the selected element up one row in the table.
	Move the selected element down one row in the table.
	Add an element to the table.
	Remove or delete the selected elements from the table.
	Sort the elements in the table in alphabetical order.
	Highlight the selected elements in another color. The <b>Highlighting Settings</b> Dialog appears which has the same functionality as the <b>Highlight</b> command in right-click pop-up menu found in Navigation Pane.
	Cut the selected element.
	Make a duplicate copy of the selected elements in the table. The duplicated elements have all the same properties as the selected elements, but with the name appended with 1 (for example in the image, when a copy and paste action is done on <i>model</i> variable the name appears as <i>model1</i> , <i>model2</i> ).
	Paste a selected element that is either cut or copied.

**Note:** With the **Cut**, **Copy**, and **Paste** icons, you can move or copy object parts to other objects. For example, select variables of a UI Contributor then either copy-paste or cut-paste to another similar UI Contributor, or any other related metadata objects that has **Variables** tab, such as User Interface or Documents.

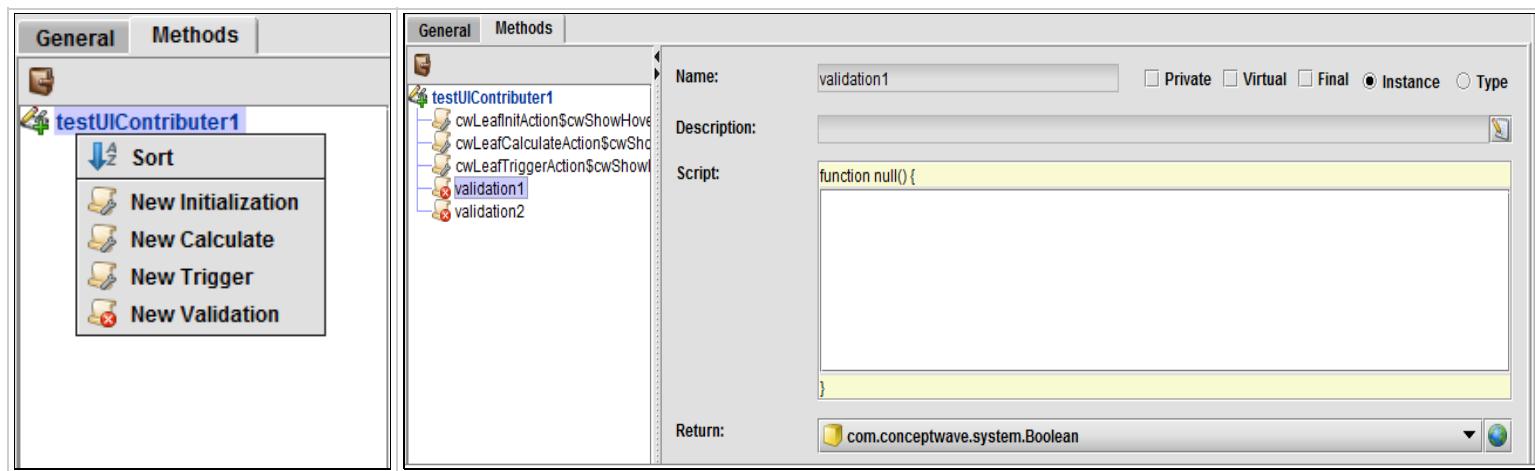
The following system variables are generated for the *com.conceptwave.system.UserInterface (User Interface)*:

Variable	Return Type	Description
<b>cwShowHover</b>	Boolean	Instructs the web browser to show hover data when the method is set to true for table's hover method.
<b>confirmObject</b>	Object	Name of the variable (used in scripts; must conform to JavaScript naming conventions).
<b>model</b>	Model	The object representing the data for this user interface contributor (that is, Document, Finder or Order).

When you select a variable in the list, all properties display on the **General** tab, while corresponding method information (if any) displays in the **Method** tab. The following table describes each property of a variable, which is presented in **General** tab and **Methods** tab.

Field	Description
<b>Name</b>	This value is automatically set to the data type's original name. It may be overwritten when necessary.
<b>Array</b>	This field specifies that this is an array of this variable.
<b>Constant</b>	If checked, the property <b>Constant value</b> appears. It becomes a constant variable that has a fixed value specified in the constant value.
<b>Audit</b>	If checked, the runtime system logs the variable update history in the database if the <b>Audit Trail</b> checkbox for the UI Contributor variable is checked in the <b>General</b> tab and the configuration application has specified that audit is allowed. Any changes made to the variable are recorded into the system table CWFIELDAUDITTRAIL in the database.  <b>Note:</b> Translation fields are audited. The old and new values in the CWFIELDAUDITTRAIL table indicate the language of the change, and takes the format <language code><translation text> (for example, en-xx HELLO).
<b>Description</b>	Description of the variable for documentation.
<b>Data Type</b>	The data type of the Variable.
<b>Type Error Code</b>	Specifies an error message code that is in the application resources. If set, the specified error message is used instead of the default system message when the field contains an invalid value.
<b>Mandatory Error Code</b>	Specifies an error message code that is in the application resources. If set, the specified error message is used instead of the default system message when the field is mandatory and it has no value.
<b>Encrypt</b>	If <b>Yes</b> is selected from the list then the variable is encrypted when saving the UI Contributor and decrypted when loading the UI Contributor.
<b>Element Properties</b>	Lists the default element properties of a variable that depends on the inherited data type's element properties.

The **Methods** tab is a [script management screen](#) in Velocity Studio, which is composed of the **Method List** pane at the left which lists all scripts of the UI Contributor Variable, and **Details** pane which displays the details of a script when selected at the **Method List** pane. To create a new method, or to override an existing method of its base UI Contributor, right-click the UI Contributor in the **Method List** pane. A pop-up menu appears.



**Note:** These methods are created at the UI Contributor variable level, impacting only to the UI Contributor variable.

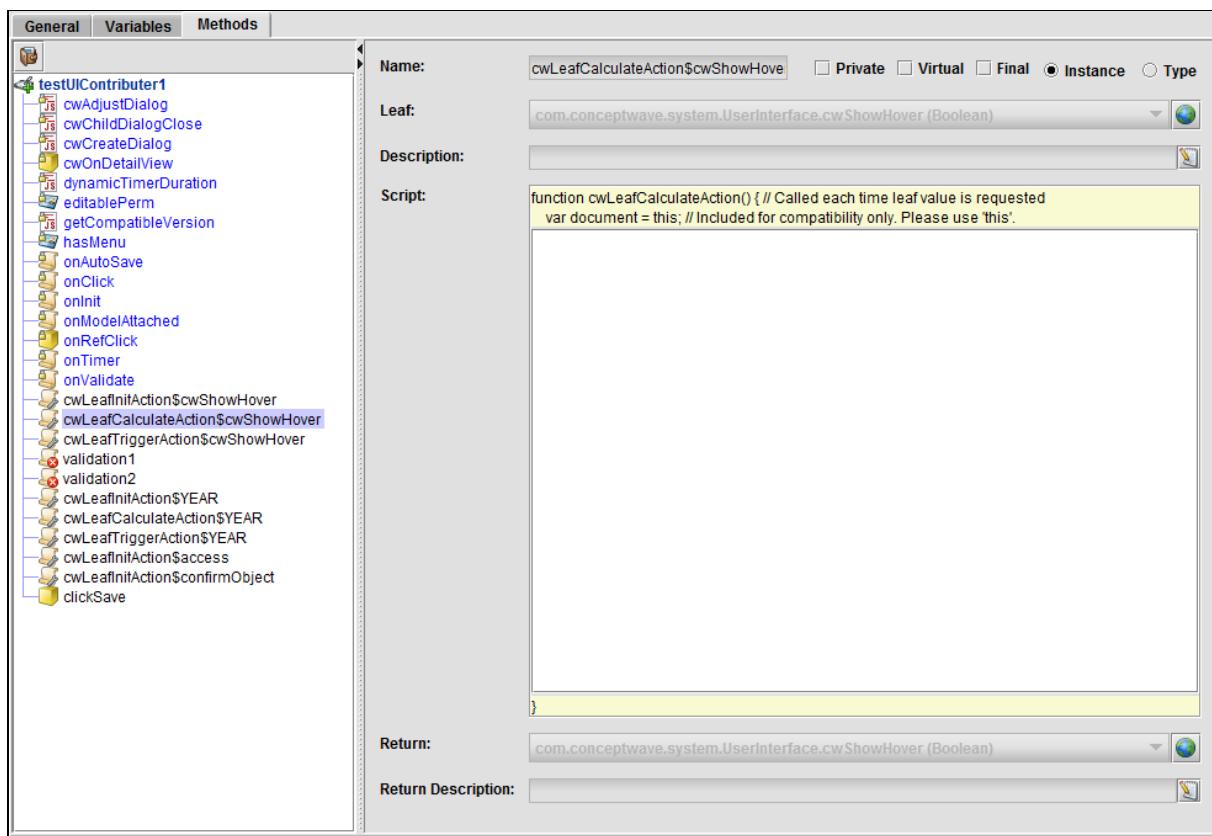
The following table lists the type of methods that can be added or overridden at the UI Contributor variable level:

Method Type	Script Name	Description
Initialization	<code>cwLeafInitAction(document)</code>	This script is to set initial values for the UI Contributor variable. It is triggered when the UI Contributor is created. This method is not executed when the UI Contributor is read from the database or an external system interface.
Calculate	<code>cwLeafCalculateAction()</code>	These variables are called each time variable (also known as leaf) value is requested. UI Contributor variable calculate script is to compute a value for the calculated UI Contributor variable. A calculated UI Contributor variable is a variable that is not stored in the database or sourced from external sources, but their values computed based on values of other UI Contributor variables. Variables with calculate method are read-only fields in the UI at runtime.  <b>Note:</b> Because the value of the calculated field is computed every time the field is accessed, it may be accessed many times for different purposes by AVM at runtime. Unwise and excessive use of calculated fields may introduce serious performance problems. It is recommended to limit the method to simple calculations based on UI Contributor variable values, and not invoke database access operations or external interface calls.
Trigger	<code>cwLeafTriggerAction(\$loadTime)</code>	This script is called when document trigger leaf is changed. The <code>\$loadTime</code> is not used currently and its value is always <code>false</code> . UI Contributor variable Trigger script is executed when the content of a UI Contributor form field is changed (and loses focus) at runtime. Trigger execution may come automatically from the UI or can come from a user script at runtime. When a trigger execution request is received, all triggers for variables that have been changed are executed.  <b>Note:</b> As a trigger method is triggered at UI, a server round trip is required to send updates to the client; this is done in AJAX, with only changed variables updated to browser with no page refresh. Regardless, trigger methods are used sparingly to avoid incurring this overhead.
Validation	<code>cwOnDocValidRule(document)</code>	This script contains validation rule and is used in generated script for UI Contributor variable. It is the validation script that determines the validity of runtime data values of the UI Contributor. It is triggered with UI Contributor's <code>onvalidate</code> method, and by default, the generated UI Contributor form's save menu button triggers such validation.

The variables that are overridden or initialized, they appear in the **Method List** pane appear with a check mark, and may no longer be available from the right-click context pop-up menu. For Method Type **Validation** you can create multiple methods (for example, Validation 1, Validation 2...), each identifiable by unique method name. The overridden variables in the method tab are accessible in the **Method** tab of the UI Contributor. For each new initialization of the variable method name is a unique. The method created in **Method List** pane can be right-clicked, with various commands available, such as **Delete**, **Highlight**, etc. For more information about **Variables**, see [script management UI](#) or [scripting](#).

## User Interface Contributor Methods Properties

The **Methods** tab contains default (also known as base), overridden, and newly created methods for the User Interface (UI) Contributor. For example in the following image, methods highlighted in color blue are default or base methods, and the methods highlighted in color black are overridden or new methods. The default variables depends on which UI Controller is selected in the **General** tab of the UI Contributor in the **Contribute To** field and which default user interface is extended by the selected UI controller.



The Methods tab is a [script management screen](#) in Velocity Studio, which is composed of the **Method List** pane on the left that lists all scripts related to the UI Contributor, and **Details** pane that displays the details of a script when selected at the **Method List** pane. The fields on the **Detail** pane of the methods tab depends on the method selected in the **Method List** pane. The following table describes the common fields for the **Detail** pane of methods tab:

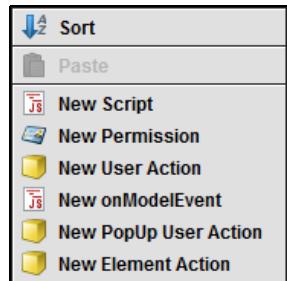
Field	Description
<b>Name</b>	This field displays the name of the method.
<b>Private</b>	If checked, the method is hidden from display when it is packaged in as Product Metadata (that is, library).
<b>Virtual</b>	If checked, it provides the ability to create methods in template metadata that does not have implementation, but can be used in template script and be bound to UI elements. The implementation is done in user metadata (see <a href="#">Virtual Field</a> for more information).
<b>Final</b>	Methods can be marked as final. Any method that has the <b>Final</b> property set to true can neither be extended, nor overridden.
<b>Instance</b>	If checked, the method is accessible by instance objects.
<b>Type</b>	If checked, the method is accessible without an instance object.
<b>Leaf</b>	This field provides the information about the base method for the selected method in the list of methods.
<b>Description</b>	Description of the method for documentation.
<b>Script</b>	Validation JavaScript script. The script can be one of the following: <ul style="list-style-type: none"> <li>An expression that evaluates to true to indicate that the text in <i>Message</i> field or the message corresponding to the specified <i>Message Code</i> is to be displayed. Otherwise it must evaluate to false.</li> <li>A script that returns true to indicate that the text in <i>Message</i> field or the message corresponding to the specified <i>Message Code</i> specified is to be displayed. Otherwise it must return false.</li> <li>A script that returns a resource identifier string or list of identifiers separated by semicolons to indicate the resource message(s) to be displayed. For example: AW0155 or AI0155, AE0156. It is recommended that the first character must be A. The second character indicates the message type: I - information, W - warning, and E - error.</li> <li>A JavaScript script returning text (that is not a resource identifier) that can be displayed as a rule message.</li> </ul>
<b>Return type</b>	The Object type to be returned by the script.
<b>Return Description</b>	Description of the return type for documentation.

To show all base methods, click the **Show base methods** ( button). Click the button again to hide all base methods. The default or base methods, whether it is a script, permission or a user method, are not accessible to the UI Contributor. These methods cannot be overridden or no changes can be done to the scripts or permissions.

If you have a long list of methods in the left pane, you can press the first letter of the method you are looking for to jump to the list of methods beginning with that letter. In this example, pressing the **E** key highlights the first instance of a method that begins with the letter E. The details of the method are displayed in the **Detail** pane.

## Create a New Method

To create a new method, right-click the UI Contributor in the **Method List** pane. The following image describes the type of methods that can be selected from the pop-up menu:



There are three types of methods that can be created:

Method Type	Description
Script	Generic user-defined JavaScript function that can be defined in UI Contributor and explicitly invoked by other methods or form elements.
Permission	Permission method that can be associated to properties of form elements to determine participant's access of the form element such as <i>Visible</i> , <i>Optional</i> and <i>Editable</i> . See section <a href="#">Permissions</a> for details.
User Action	User Action Method that implements the response of user actions, such as clicking a button or an image, by returning an object presented in a specified form to the user. See section <a href="#">User Action Method</a> for details.

You can create multiple methods of the same method type by using unique method names (for example, `Permission1`, `Permission2`). A newly added method in the UI controller, the one that is selected in the **Contribute To** field, in the **General** tab of the UI Contributor, comes in the list of methods in the UI Controller's **Method** tab.

The methods created in **Method List** pane can be right-clicked, with various commands available, such as **Delete**, **Show Usage**, etc. For more information see, [Scripting Interface](#).

## Built-in Service Providers

---

The Application Virtual Machine (AVM) is capable of performing interactions with external services through service providers. A service provider is implemented as a Java class, and supports one specific technology (for example, EJB or SOAP). The service provider exposes all the concrete details required to interact with external services. The framework provides built-in support for the following Service Provider types:

- [DBSP](#)
- [FTP](#)
- [HTTP](#)
- [JAVA](#)
- [JMS](#) (supports asynchronous mode - can be used to implement API listener)
- [JS](#) (implements external service with JavaScript functions)
- [LDAP](#)
- [MQ](#)
- [SMTP](#)
- [SOAP](#)
- [SOCKET](#)
- [TIBCO](#) (supports asynchronous mode - can be used to implement API listener)

All built-in service providers, with the exception of JMS and TIBCO, interact with external services synchronously.

Additionally, some built-in service providers [support listener functionality](#).

## Listener Support

---

Some of the built-in service providers support listener functionality. This functionality allows your application to provide services to external systems, such as receiving and processing incoming requests. The following service providers support listener functionality:

- [JMS](#)
- [MQ](#)
- [SOAP](#)
- [TIBCO](#)

In general, the following steps create a listener. See the articles in this section for specific listener setups for specific service providers.

1. Create an [Interface](#) with one Operation of type *one-way* or *request-response* (Select **One-Way** or **Request-Response** in the **Operation type** combo box).
2. Create a [Binding](#) that binds this Interface to one of the service providers listed above (Select the Interface in the **Interface** combo box and the service provider in the **Provider** combo box).
3. Create a Listener [Service Port](#) for this Binding (Check the **Listener** check box of the Port properties).
4. In the Configuration application, this listener Port can be found and configured on the [Listeners](#) horizontal tab of the **Services** vertical tab, as well as in [Services](#) horizontal tab.
5. To pass the message to JavaScript for processing, create a Binding that binds the same Interface to the JS (JavaScript) service provider (Select the Interface in the **Interface** combo box and the **JS Service Provider** in the **Provider** combo box).  
Then, implement the Interface Operation script functions in the [JavaScript Binding](#) (Select the **Operation**  element in the **Interface element** tree of the Binding properties, and define the **Script** attribute value in the **Element definitions** table).  
Lastly, create a Service Port for this JS Binding (Select the JS Binding in the **Binding** combo box)

**Note:** For JMS and MQ listeners, the port settings support multiple operations (per queue) under one port. However, more than one message queue port cannot be active for the same operation.

## DBSP Service Provider

The DBSP Service Provider allows invocation of Oracle stored procedures and functions.

### DBSP Attributes in Velocity Studio

The following table describes the attributes that are exposed in Velocity Studio specifically for the DBSP Service Provider.

These attributes are found at **Binding** metadata object in Velocity Studio. When an Interface, Operation or message element is selected in the **Interface element** tree, the attributes for the element appears in the **Element definitions** table where values can be provided.

[Attributes in Velocity Studio for the DBSP Service Provider.](#)

Attribute	Description	Element						
<b>Procedure</b>	Stored procedure name.	Operation 						
<b>Function</b>	Combo box. Defined if the stored procedure is a function that returns a value.	<p>Operation </p> <p><b>DBSP Attributes in Configuration Application</b></p> <p>The following table describes the attributes that are exposed in Configuration application specifically for the DBSP Service Provider.</p> <p>These attributes are found in <b>Services</b> tab in Configuration application. When clicking a Port, Binding, Operation or message node in the <b>Service Port tree</b>, the Configuration attributes for the node appears at the <b>Configuration</b> section at the right where values can be provided.</p> <p><u><a href="#">Attributes in Configuration application for the DBSP Service Provider.</a></u></p> <table border="1"><thead><tr><th>Attribute</th><th>Description</th><th>Element</th></tr></thead><tbody><tr><td><b>Schema</b></td><td>Schema from the Configuration application in which the stored procedure is defined.</td><td>Port </td></tr></tbody></table>	Attribute	Description	Element	<b>Schema</b>	Schema from the Configuration application in which the stored procedure is defined.	Port 
Attribute	Description	Element						
<b>Schema</b>	Schema from the Configuration application in which the stored procedure is defined.	Port 						

## FTP Service Provider

The FTP Service Provider allows the transfer of XML files to/from FTP server.

### FTP Attributes in Velocity Studio

The following table describes the attributes that are exposed in Velocity Studio specifically for the FTP Service Provider.

These attributes are found at **Binding** metadata object in Velocity Studio. When an Interface, Operation or message element is selected in the **Interface element** tree, the attributes for the element appears in the **Element definitions** table where values can be provided.

[Attributes in Velocity Studio for the FTP Service Provider.](#)

Attribute	Description	Element																		
<b>Direction</b>	The FTP operation to be performed: <i>put</i> or <i>get</i> .	<p>Operation </p> <p><b>FTP Attributes in Configuration Application</b></p> <p>The following table describes the attributes that are exposed in Configuration application specifically for the FTP Service Provider.</p> <p>These attributes are found in <b>Services</b> tab in Configuration application. When clicking a Port, Binding, Operation or message node in the <b>Service Port tree</b>, the Configuration attributes for the node appears at the <b>Configuration</b> section at the right where values can be provided.</p> <p><u><a href="#">Attributes in Configuration application for the FTP Service Provider.</a></u></p> <table border="1"><thead><tr><th>Attribute</th><th>Description</th><th>Element</th></tr></thead><tbody><tr><td><b>Location</b></td><td>Specifies the location of FTP Server.</td><td>Port </td></tr><tr><td><b>Port</b></td><td>Specifies the port to use. If not specified the default FTP port 21 is used.</td><td>Port </td></tr><tr><td><b>User</b></td><td>Specifies the user name if the FTP Server requires authentication.</td><td>Port </td></tr><tr><td><b>PasswordKey</b></td><td>Specifies the password key to use for retrieving the password. The password key/value pairs are created in the Configuration application.</td><td>Port </td></tr><tr><td><b>Location</b></td><td>File name to get or put. If the file name is not specified and the put operation is performed, the file name is the runtime ID of the data object being sent. In this case the file type is .xml.</td><td>Operation </td></tr></tbody></table>	Attribute	Description	Element	<b>Location</b>	Specifies the location of FTP Server.	Port 	<b>Port</b>	Specifies the port to use. If not specified the default FTP port 21 is used.	Port 	<b>User</b>	Specifies the user name if the FTP Server requires authentication.	Port 	<b>PasswordKey</b>	Specifies the password key to use for retrieving the password. The password key/value pairs are created in the Configuration application.	Port 	<b>Location</b>	File name to get or put. If the file name is not specified and the put operation is performed, the file name is the runtime ID of the data object being sent. In this case the file type is .xml.	Operation 
Attribute	Description	Element																		
<b>Location</b>	Specifies the location of FTP Server.	Port 																		
<b>Port</b>	Specifies the port to use. If not specified the default FTP port 21 is used.	Port 																		
<b>User</b>	Specifies the user name if the FTP Server requires authentication.	Port 																		
<b>PasswordKey</b>	Specifies the password key to use for retrieving the password. The password key/value pairs are created in the Configuration application.	Port 																		
<b>Location</b>	File name to get or put. If the file name is not specified and the put operation is performed, the file name is the runtime ID of the data object being sent. In this case the file type is .xml.	Operation 																		

## HTTP Service Provider

The HTTP service provider allows interaction with services over HTTP connection. It supports both GET (URL-encoded) and POST (form or XML-encoded) verbs. The HTTP service provider supports login and session cookies for authenticated sessions.

**Note:** Parsing valid XML with leading whitespace is supported.

### HTTP Attributes in Velocity Studio

The table that follows describes the attributes that are exposed in Velocity Studio specifically for the HTTP service provider.

These attributes are found at **Binding** metadata object in Velocity Studio. When an Interface, Operation, or message element is selected in the **Interface element tree**, the attributes for the element appears in the **Element definitions** table where values can be provided.

[Attributes in Velocity Studio for the HTTP service provider](#):

Attribute	Description	Element																		
<b>Scope</b>	Specifies how the session cookie is used. Can have one of the following values: <ul style="list-style-type: none"> <li><b>application</b>: All session cookies are shared by all user sessions. This is the default behavior.</li> <li><b>session</b>: Each session cookie is dedicated to one user session.</li> <li><b>request</b>: The session cookie is used for the duration of one user request (when invoking service provider operations from within the same JavaScript, all service invocations are considered one request).</li> <li><b>none</b>: A session cookie is not used between interface invocations.</li> </ul>	Interface 																		
<b>implementation</b>	Determines what HTTP API implementation will be used for sending out HTTP request. Takes one of two options: <i>default</i> (Sun Java) or <i>ConceptWave</i> . If left blank, the latter is used.	Interface 																		
<b>encoding</b>	Determines the <i>charset</i> attribute in the HTTP request header. If left blank, <i>charset</i> will take one of two values: <ul style="list-style-type: none"> <li>If the <i>encodingStyle</i> setting for <i>Input</i> in this HTTP binding is left blank or set as <i>xml</i>, then <i>UTF-8</i> will be used as the <i>charset</i>.</li> <li>Otherwise, <i>ISO-8859-1</i> will be used for <i>charset</i>.</li> </ul>	Interface 																		
<b>Login</b>	Specifies if this Operation is a <i>login</i> operation. Possible values are <i>yes</i> and <i>no</i> . Default value is <i>no</i> . Only one Operation can be marked as a login operation. A login operation does not have to be implicitly called; the HTTP Service Provider will invoke the login operation automatically when invoking any other Operation on that Interface if a session ID cannot be obtained. For an automatic login operation to succeed, a Document that represents the login input message should be initialized with the proper values (user ID, password, and so on) through initialization.	Operation 																		
<b>Logout</b>	Specifies if this Operation is a <i>logout</i> operation. Possible values are <i>yes</i> and <i>no</i> . Default value is <i>no</i> . Only one Operation can be marked as a logout operation. When a logout operation is called, the session ID is removed from the cache and any subsequent Operation will require performing a login operation.	Operation 																		
<b>HttpMethod</b>	This attribute specifies the HTTP method for the selected operation. The POST, GET, PUT, DELETE, and HEAD methods are available.	Operation 																		
<b>Accept</b>	This is a HTTP-protocol-standard attribute in HTTP header, which defines what can be accepted in the response from the server to which this HTTP request is sent to. For specific information refer to RFC 2616 at <a href="http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html">http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html</a> . For instance, if an XML response over HTTP is expected, it should be <i>text/xml</i> .	Operation 																		
<b>Encoding Style</b>	Specifies how message data is formatted when performing a HTTP request. Can have one of the following values: <table border="1" data-bbox="291 1318 777 1576"> <thead> <tr> <th>Value</th><th>Description</th><th>Message</th></tr> </thead> <tbody> <tr> <td><b>XML</b></td><td>Data formatted as a XML document based on a Document or Data Structure definition.</td><td>Input, Output and Fault messages.</td></tr> <tr> <td><b>Form</b></td><td>Data is encoded exactly as in urlEncoded but a POST request is performed instead of GET.</td><td>Input.</td></tr> <tr> <td><b>UriReplacement</b></td><td>Indicates that all the message data parts are encoded into the HTTP request URI using a replacement</td><td>Input.</td></tr> </tbody> </table>	Value	Description	Message	<b>XML</b>	Data formatted as a XML document based on a Document or Data Structure definition.	Input, Output and Fault messages.	<b>Form</b>	Data is encoded exactly as in urlEncoded but a POST request is performed instead of GET.	Input.	<b>UriReplacement</b>	Indicates that all the message data parts are encoded into the HTTP request URI using a replacement	Input.	Message  <b>HTTP Attributes in Configuration Application</b> <p>The table that follows describes the attributes that are exposed in Configuration application specifically for the HTTP service provider.</p> <p>These attributes are found in <b>Services</b> tab in Configuration application. When clicking a Port, Binding, Operation or message node in the <b>Service Port tree</b>, the Configuration attributes for the node appears at the <b>Configuration</b> section at the right where values can be provided.</p> <p><u><a href="#">Attributes in Configuration application for the HTTP service provider</a></u>:</p> <table border="1"> <thead> <tr> <th>Attribute</th><th>Description</th><th>Element</th></tr> </thead> <tbody> <tr> <td><b>location</b></td><td>Specifies the base URI for the Port. The value of the attribute is combined with the values of the <b>Location</b> attribute of the Operation element of the Binding properties. For external services, this would specify the metadata name of the Operation being invoked at this port.</td><td>Port </td></tr> </tbody> </table>	Attribute	Description	Element	<b>location</b>	Specifies the base URI for the Port. The value of the attribute is combined with the values of the <b>Location</b> attribute of the Operation element of the Binding properties. For external services, this would specify the metadata name of the Operation being invoked at this port.	Port 
Value	Description	Message																		
<b>XML</b>	Data formatted as a XML document based on a Document or Data Structure definition.	Input, Output and Fault messages.																		
<b>Form</b>	Data is encoded exactly as in urlEncoded but a POST request is performed instead of GET.	Input.																		
<b>UriReplacement</b>	Indicates that all the message data parts are encoded into the HTTP request URI using a replacement	Input.																		
Attribute	Description	Element																		
<b>location</b>	Specifies the base URI for the Port. The value of the attribute is combined with the values of the <b>Location</b> attribute of the Operation element of the Binding properties. For external services, this would specify the metadata name of the Operation being invoked at this port.	Port 																		

	<p>algorithm. The relative URI value of the <b>Location</b> attribute of the Binding properties is searched for a set of search patterns. The search occurs before the value of the <b>Location</b> attribute of the Binding properties is combined with the value of the Address <b>Location</b> attribute of the Port properties. There is one search pattern for each message part. The search pattern string is the name of the message part surrounded with parenthesis "(" and ")". For each match, the value of the corresponding message part is substituted for the match at the location of the match. Matches are performed before any values are replaced (replaced values do not trigger additional matches).</p> <p>Message data parts MUST NOT have repeating values.</p> <p>For example: Location = "o1/A(part1)B(part2)/(part3)" port address = "<a href="http://example.com/">http://example.com/</a>"</p> <p>If the values being passed are: part1=1, part2=2, part3=3 URL=" <a href="http://example.com/o1/A1B2/3">http://example.com/o1/A1B2/3</a> "</p>	
<b>UriEncoded</b>	Indicates that all the message data parts are encoded into the HTTP request URI using the standard URI-encoding rules (name1=value&name2=value...). The names of the parameters correspond to the names of the message data parts (Document or Data Structure elements). Each value contributed by the part is encoded using a name-value pair. This will perform an HTTP GET request. The "?" character is automatically appended as necessary.	Input.
<b>Text</b>	Data returned by the service is not processed and is returned as a text (string).	Output.

**Note:** When you have an HTTP listener, and you specify the **encodingStyle** field for both the Input and Output, the listener returns the Content Type as the response, which depends on the encoding set for the Output.

As an example, if the **Input** is text and the **Output** is json, the Content Type in the response headers is application/json.

user	Basic authentication user name.	Port
passwordKey	Basic authentication password name.	Port
<b>Note:</b> The actual password is contained in the password list referenced by this value.		
<b>reuseSession</b>	If set to 'no', a new user context is created for each request. Absence of this attribute is defaulted to yes, in which the user context will be reused. Used by listener ports only.	Port
<b>Location</b>	Specifies a relative URI for the Operation. This URI is combined with the URI specified for the Port Address <b>Location</b> attribute to form the full URI for the HTTP request. For HTTP listeners, this address takes the form, by default,	Operation
	<pre>http://host:port/cwf/api/&lt;namespaceName&gt;-&lt;interfaceName&gt;</pre>	
<b>Timeout</b>	Number of seconds to wait before the HTTP response come back. If the response does not come in that time, the underlying HTTP software generates an error.	Operation

Every time an XML message is sent or received, Velocity Studio records the send or receive request in the cwmessagelog table. With the HTTP service provider, both the request and the time of the request are logged.

**Notes:**

- The cwf.messageLog metadata object replaces the cwf\_pm.messageLog document and has no child user interface.
- You can also create an [event handler](#) using the SYSTEM\_MESSAGELOG\_STORE event, which stores the application message log and is called from the ProviderFactory Java class. This event takes the following parameters:
  - msgLogId (optional)
  - msgLogDataDoc (optional)
  - interfaceTypeNo (mandatory)
  - operationName (mandatory)
  - sendData (optional)
  - receiveData (optional)
  - date (optional)
  - separateConnection (optional)
  - propertiesArray (mandatory)
  - attemptCount (mandatory)
  - failure (mandatory)

**Use a keystore for two-way SSL handshake**

After you have set up your Java truststore and keystore, set the following system properties in your Java command line:

```
-Djavax.net.ssl.keyStore=<keystore_path>
-Djavax.net.ssl.trustStore=<truststore_path>
-Djavax.net.ssl.keyStorePassword=<keystore_password>
-Djavax.net.ssl.trustStorePassword=<truststore_password>
```

If your keystore or truststore types are not JKS, you have to include the following system properties:

```
-Djavax.net.ssl.keyStoreType=<type (for example, pkcs12)>
-Djavax.net.ssl.trustStoreType=<type (for example, pkcs12)>
```

You can use the following property to debug the SSL communication:

```
-Djavax.net.debug=ssl
```

**Serialize Message Processing by Listeners**

You can serialize message processing by listeners without requiring database access. This feature does the following:

- Supports synchronous requests
- Serializes incoming listener API requests based on a key
- Provides a mechanism for metadata to know whether a request has timed out while waiting to be processed in serial
- Allows metadata to indicate a serialization timeout
- At the system level, enforces low maximum timeouts (5 seconds) to limit the number of waiting requests

**Notes:**

- No configuration fields are available for each interface. Instead, your metadata can use configuration variables.
- A backlog or list of waiting requests is not maintained. As a result, serialization behaviour is not dependent on any such list. Your metadata can maintain a count, if necessary, and exit early if the count is high.
- Request starvation (that is, requests being blocked abnormally long) is not addressed since timeouts are enforced. It is also expected that blocking is rare.
- No mechanism for limiting the number of blocked threads is provided since maximum timeouts are expected to be low.
- Using this feature may limit the throughput of product listeners only if the mechanism is used. With SOAP and HTTP, request threads blocked by this mechanism reduce the total number of threads available for processing incoming HTTP requests by the container for any type of request (that is, UI or API).

Listeners are exposed as scriptable objects in listener error handling scripts. You can process API scripts in the scope of the listener instance and the listener instance can be

accessed in script using `this`.

The following API is a part of the Listener object, which blocks the listener thread until any other thread is processed with the same serialization key:

```
public boolean serialize(String key, int timeout)
```

The `key` parameter indicates what requests are serialized. The `timeout` parameter is in milliseconds and is likely to be the same for all related requests. Your metadata needs to get this value either from a constant or from configuration variables. See the JavaScript documentation for details about this API.

Listener API scripts may use this method at the beginning of the script. Ensure that your script checks its result to discover whether it is being processed in serial or the timeout has expired.

If a listener thread is the first to try to serialize on a key, it inserts the key into a static map. When other listener threads attempt to serialize on that key simultaneously, that key exists in the map. The threads then perform a sleep-and-check routine until either the timeout has expired or the key is no longer in the map.

At the end of processing the listener (API) script, if the thread has obtained serialization (that is, placed a key in the map), the serialization key is removed from the map, which unblocks any other thread waiting on that key. As a result, map entries are guaranteed to exist only for the duration of listener script's processing, which limits the map's size to the number of configured listener threads.

## Set up an HTTP API Listener

To set up the HTTP API listener, complete these steps:

1. Create an [\*\*Interface\*\*](#) with all the required Operations of type *one-way* or *request-response* (Select **One-Way** or **Request-Response** in the **Operation type** combo box).
2. Create a [\*\*Binding\*\*](#) that binds this Interface to the HTTP service provider (Select the Interface in the **Interface** combo box and the **HTTP Service Provider** in the **Provider** combo box).
3. Create a [\*\*Service Port\*\*](#) for this Binding and set it as listener port (Check the **Listener** check box of the Port properties).
4. Create a Binding that binds the same Interface to the JS (JavaScript) service provider (Select the Interface in the **Interface** combo box and the **JS Service Provider** in the **Provider** combo box).
5. Implement the Interface Operation script functions in the [\*\*JavaScript Binding\*\*](#) (Select the **Operation**  element in the **Interface element** tree of the Binding properties and define the **Script** attribute value in the **Element definitions** table).
6. Create a Service Port for this JS Binding by selecting the JS Binding in the **Binding** combo box.
7. Save and then run your metadata.
8. In the Configuration application, this Port can be found and configured at the [\*\*Services\*\*](#) and [\*\*Listeners\*\*](#) horizontal tab of the **Services** vertical tab.
9. In the Configuration application, ensure that the HTTP access for the API is allowed with no authentication (that is, the [\*\*HTTP API login required\*\*](#) property is unchecked).
10. The AVM sets up this service under the following URL: `http://host:port/cwf/api/<namespaceName>-<interfaceName>`, where `<namespaceName>` and `<interfaceName>` are the names of the Namespace and Interface respectively. (For example, Operation "getEquipment" in Interface "sampleInf" of the "sampleNS" Namespace may have an URL of `http://localhost:8080/cwf/api/sampleNS-sampleInf/getEquipment`.) The HTTP API listener supports the POST method.

## HTTP SMS Provider Support

When developing your application, you may have the need to implement Short Message Service (SMS) message-sending capabilities. Employing third-party SMS providers allows you to send SMS messages. One obstacle during implementation is that different SMS providers have different sets of parameters and it would be helpful to have a generic interface that can be configured to work with all SMS providers.

Using the HTTP protocol, to configure your interface to support different SMS providers, do the following:

1. In the Configuration application, under **Services > Services**, navigate to your HTTP interface.
2. Under Configuration, for the **Location** field, enter a URL containing placeholders between parentheses (that is, {}), which specifies a location such as the following:

```
| http://host:port/SMSsend?number={num_field}&user={user_field}&pass={pass_field}&message={message_field}
```

where *num\_field*, *user\_field*, *pass\_field*, and *message\_field* are fields of the input data structure. That way, the interface can be configured at runtime to adapt to different SMS providers.

3. In your HTTP binding, for the **encodingStyle** attribute, select **URLReplacement** and save your changes.

## Java™ Service Provider

The Java Service Provider allows invocation of external Java class methods. Only primitive types are supported (for example, Date, String, Number). As such, to transfer complex types, you may need to create a bridge -- a Java method that translates the complex type to the primitive types.

**Note:** The JAR file of the external Java class should be in the CLASSPATH.

### JAVA Attributes in Velocity Studio

The following table describes the attributes that are exposed in Velocity Studio specifically for the Java Service Provider.

These attributes are found at [Binding](#) metadata object in Velocity Studio. When an Interface, Operation or message element is selected in the **Interface element** tree, the attributes for the element appears in the **Element definitions** table where values can be provided.

#### [Attributes in Velocity Studio for the Java Service Provider](#)

Attribute	Description	Element									
<b>Scope</b>	Specifies how the session cookie is used. Can have one of the following values: <ul style="list-style-type: none"><li>• <b>application</b>: All session cookies are shared by all user sessions. This is the default behaviour.</li><li>• <b>session</b>: Each session cookie is dedicated to one user session.</li><li>• <b>request</b>: The session cookie is used for the duration of one user request (when invoking service provider operations from within the same JavaScript, all service invocations are considered one request).</li><li>• <b>none</b>: A session cookie is not used between interface invocations.</li></ul>	Interface 									
<b>Method</b>	The class method name.	Operation 									
<b>Constructor</b>	Specifies that this method is a constructor.	Operation 									
<h3>Java Attributes in Configuration Application</h3> <p>The following table describes the attributes that are exposed in Configuration application specifically for the Java Service Provider.</p> <p>These attributes are found in <a href="#">Services</a> tab in Configuration application. When clicking a Port, Binding, Operation or message node in the <b>Service Port tree</b>, the Configuration attributes for the node appears at the <b>Configuration</b> section at the right where values can be provided.</p> <p><a href="#">Attributes in Configuration application for the Java Service Provider</a></p>											
<table border="1"><thead><tr><th>Attribute</th><th>Description</th><th>Element</th></tr></thead><tbody><tr><td><b>Location</b></td><td>Fully qualified Java class name.</td><td>Port </td></tr><tr><td><b>cacheSize</b></td><td>The number of connections to be cached for the Service Provider. 100</td><td>Port </td></tr></tbody></table>			Attribute	Description	Element	<b>Location</b>	Fully qualified Java class name.	Port 	<b>cacheSize</b>	The number of connections to be cached for the Service Provider. 100	Port 
Attribute	Description	Element									
<b>Location</b>	Fully qualified Java class name.	Port 									
<b>cacheSize</b>	The number of connections to be cached for the Service Provider. 100	Port 									

is the default, and also capped to 100.

## JMS Service Provider

The JMS Service Provider allows interactions with other systems through JMS Server (Java Messaging Service). The JMS Service Provider supports connection queues only (no topics) for sending and receiving messages.

### Invoking JMS Operations

When a JMS interface operation is invoked, a logical connection is initiated to a specific location. When completed, the connection is cached. If it is not used again within one hour (the time is not configurable), the connection is closed. The connection is not shared between threads. So, if interface operations are invoked with the same location simultaneously by different threads, extra connections are created and remain open for one hour, if they are not used. Additionally, if an error or exception occurs during the invocation of an operation, the connection is closed and is not cached.

**Note:** The JMS Service Provider only supports Request-response and One-way interface operation types. JMS listeners are port based. That is, a port needs to be defined for each listener. Port settings for JMS listeners allow the choice of an operation meaning it can support multiple operations, however each JMS listener is tied to a specific operation. By default, the operation is the first operation in the interfaces operation list.

### JMS Listeners

JMS listeners have a single logical connection created for each number of threads that are configured. During normal operation, that connection is open for the listener's lifetime. In the configuration, at the operation level, there is a **Distributed** flag. If this flag is set to true, the connection is shut down and an attempt is made to reconnect approximately every five minutes (the time is not configurable). The time is approximate, and depends on how long the listener has been waiting for a message and how many threads are defined. If multiple listener threads are defined, the time is staggered between them by 10 seconds so that not all threads are shut down at once. If any kind of error occurs in between establishing a connection and trying to receive a message, the connection is shut down after 600 attempts (the number of attempts is not configurable) and attempts to get a new connection. If an error occurs trying to obtain a connection, the listener waits for ten seconds (time is not configurable) before it tries again. These attempts continue indefinitely.

### JMS Attributes in Velocity Studio

The table below describes the attributes that are exposed in Velocity Studio specifically for the JMS Service Provider.

These attributes are found at [Binding](#) metadata object in Velocity Studio. When an Interface, Operation or message element is selected in the **Interface element** tree, the attributes for the element appears in the **Element definitions** table where values can be provided.

#### [Attributes in Velocity Studio for the JMS Service Provider](#)

Attribute	Description	Element						
<b>Correlate</b>	Gives options to <b>Correlate</b> request-response messages. If a selection is made here, there is no need to specify a <b>CorrelateProperty</b> (that is, the system automatically generates a sequence/key). <ul style="list-style-type: none"><li>• <b>msgID</b> - unique ID assigned by bus</li><li>• <b>sequence</b> - unique ID assigned by system</li><li>• <b>no</b> - no correlation</li></ul>	Output Message 						
<b>CorrelateProperty</b>	User-specified <b>Correlate</b> property (for example, <code>msg.jms.correlationPropertyId = orderid</code> )	Message  <b>JMS Attributes in Configuration Application</b> The table below describes the attributes that are exposed in Configuration application specifically for the JMS Service Provider. These attributes are found in <a href="#">Services</a> tab in Configuration application. When clicking a Port, Binding, Operation or message node in the <b>Service Port tree</b> , the Configuration attributes for the node appears at the <b>Configuration</b> section at the right where values can be provided. <a href="#">Attributes in Configuration application for the JMS Service Provider</a> . <table border="1"><thead><tr><th>Attribute</th><th>Description</th><th>Element</th></tr></thead><tbody><tr><td><b>Location</b></td><td>Specifies the location of JNDI Server.</td><td>Port </td></tr></tbody></table>	Attribute	Description	Element	<b>Location</b>	Specifies the location of JNDI Server.	Port 
Attribute	Description	Element						
<b>Location</b>	Specifies the location of JNDI Server.	Port 						

<b>JNDIClass</b>	Specifies the full name of Initial Context Factory class.	Port 
<b>Factory</b>	Specifies the name of queue connection factory.	Port 
<b>User</b>	Specifies the user name if the JNDI Server requires authentication.	Port 
<b>PasswordKey</b>	Specifies the password key to use for retrieving the password. The password key/value pairs are created in the Configuration application.	Port 
<b>cacheSize</b>	The number of connections to be cached for the Service Provider. 100 is the default, and also capped to 100.	Port 
<b>Transactional</b>	Yes/No - Request message will stay in queue until after receiving a confirmation response from receiving end. Only relevant for listeners and servers, and not clients.	Operation 
<b>ExpiryTime</b>	The duration, in seconds, that the message is in queue before expiry.	Operation 
<b>Priority</b>	JMS defines a 10 level priority value with 0 as the lowest and 9 as the highest. Clients should consider 0-4 as gradients of normal priority and 5-9 as gradients of expedited priority. Priority is set to 4, by default.	Operation 
<b>ReplyTo</b>	Set where a reply to this message should be sent.	Operation 
<b>Queue</b>	Specifies the queue name where messages are put into or retrieved from.	Message 
<b>Timeout</b>	Number of seconds to wait when retrieving a message from the queue. Default is 1 second. If the <b>Timeout</b> is set at a value of 0 (zero), the wait will be indefinite. Applies to output messages only.	Output Message 
<b>Properties</b>	Name of a global script that is invoked with the input data object. For sending messages, if it returns a data object (Document, Order or Data Structure) the names and values of the data leaves of the returned objects are added as	Message   <b>JMS Message Transaction Control (read)</b>  Transaction support is available only when an interface is used by the listener (incoming messages). This option, when set to 'yes' will allow listener perform message processing in transaction manner: If message processing encounters any errors (usually in script processing module), message will be put back on the queue for reprocessing. Message reprocessing parameters, such as message reprocessing attempt, priority, delay, and so on, are usually controlled by

properties to the message. Specifically:

- **JMSDeliveryMode** - value of 1 or 2 (1 = NON\_PERSISTENT, 2 = PERSISTENT)

For receiving messages (that is, for both client and listener), this field contains the Global function that returns a String expression. This expression's syntax is based on a subset of the SQL92 conditional expression syntax, such as NewsType = 'Sports' OR NewsType = 'Opinion'. Refer to JMS Message Selectors in the JavaScript documentation for details.

**Note:** By default, a JMS message is set to PERSISTENT.

JMS Server configuration.

#### Set JMS Connection-Specific Credentials

If values are found in the Config Variables tab in the Configuration application for port JMS connection credentials, they are used when creating queue connections (that is, `javax.jms.QueueConnectionFactory.createConnection(String user, String password)` is used).

To configure your configuration variables, do the following:

1. Log in to the Configuration application.
2. Select the CLUSTER node, which applies the changes to all nodes and uses the same credentials. If you want to apply different credentials or ports for different nodes, make the pertinent changes by selecting each node.
3. Click the Config Variables tab.
4. Click the **Add** button, and enter `<port>.cw.jms.user` in the **Name** field and the username in the **Value** field.
5. Click the **Add** button again, and enter `<port>.cw.jms.password` (for example, `10008.cw.jms.user`) in the **Name** field and the password in the **Value** field.
6. Click the **Save** button to save your changes.

**Note:** If set, the username and password set for each port in the configuration are still used to set up the initial context environment.

Logging support allows you to view whether your credentials have been located and used. To set up logging, do the following:

1. In the Configuration application, click the Logging tab.
2. Click the **Add** button and enter `com.conceptwave.serviceprovider.JMSServiceProvider` in the **Name** field.
3. Set the logging level to **DEBUG**.
4. Click the **Save** button to save your changes.

Once you have enabled your configuration, the following messages appear in the message log:

- JMSServiceProvider - <Port name>: <Port name>.cw.jms.connection.user found in configuration
- JMSServiceProvider - <Port name>: <Port name>.cw.jms.connection.password found in configuration
- JMSServiceProvider - <Port name>: Setting up Initial Context Environment with port credentials
- JMSServiceProvider - <Port name>: Creating queue connection with configuration variable credentials

#### JMS Topic Support and Connection-Level Authentication

You can publish JMS topics for JMS and [MQ service providers](#), which allows for publishing messages to existing JMS topics for client invocation. This feature also contains listener support.

Additionally, for JMS and MQ service providers and listeners, you can create a JMS connection with an explicit username and password, if properly configured. This feature is useful if the JMS modules and JMS destination have different security policies to access. It is also useful when creating a JMS connection to [Tibco](#), which requires two validations to connect.

**Note:** The product does not support receiving messages from topics. As a result, you can only use a one-way operation to publish messages to topics, and cannot use a request-response operation to publish and receive message from topics.

For connection-level authentication, the following configuration fields are available at the Message level:

A description of each field appears in the following table:

Field	Description
<b>Queue or Topic</b>	The default value is <b>Queue</b> . If the JMS destination is topic, you must click the dropdown menu and select <b>Topic</b> for this field.  <b>Note:</b> <b>Topic</b> is strictly for one-way operations. If you select <b>Topic</b> for a request-response operation, a validation warning appears.
<b>Use port credential</b>	By default, this field is unchecked. Selecting this checkbox indicates that the user and password are cleared and disabled.
<b>User</b>	This field denotes the username used in creating the JMS connection.  <b>Note:</b> The credential setting for INPUT or OUTPUT must be identical.
<b>Password</b>	This field denotes the password in creating the JMS connection.  <b>Note:</b> The credential setting for INPUT or OUTPUT must be identical.

Keep the following points in mind between port credentials versus message-level credentials:

- If the **Use port credential** field is selected, the JMS connection is created using the username and password from the Port setting. Additionally, the JNDI context contains the username and password from the Port setting.
- If the **Use port credential** field is unchecked, and the username and password on the Message level is empty, the JMS connection is created without the username and password. Instead, the JNDI context contains the username and password from the Port level.
- If the **Use port credential** field is unchecked, and the username and password on the Message level is not empty, the JMS connection is created with the username and password from the Message Level. The JNDI context contains the username and password from the Port level.

## JavaScript Service Provider

The JavaScript Service Provider allows invocation of JavaScripts. For example, JS Service Provider can be binded to an Interface as test-stubs, or to implement an Interface (see section [Listener Support](#)).

### JS Attributes in Velocity Studio

The table below describes the attributes that are exposed in Velocity Studio specifically for the JavaScript service provider.

These attributes are found at **Binding** metadata object in Velocity Studio. When an Interface, Operation or message element is selected in the **Interface element** tree, the attributes for the element appears in the **Element definitions** table where values can be provided.

#### [Attributes in Velocity Studio for the JavaScript Service Provider](#)

Attribute	Description	Element
Script	The script function of the operation to invoke.	Operation  <b>JS Attributes in Configuration Application</b> The JS Port is not shown in Configuration application. No JS attributes to be set.

## LDAP Service Provider

The LDAP Service Provider allows access to an LDAP Server from within the product framework and supports 5 standard LDAP protocol-defined operations: search, modify, add, delete and modifyRdn.

### LDAP Attributes in Velocity Studio

The table below describes the attributes that are exposed in Velocity Studio specifically for the LDAP Service Provider.

These attributes are found at [Binding](#) metadata object in Velocity Studio. When an Interface, Operation or message element is selected in the **Interface element** tree, the attributes for the element appears in the **Element definitions** table where values can be provided.

#### Attributes in Velocity Studio for the LDAP Service Provider

Attribute	Description	Element																								
<b>Operation</b>	Operation name (see Table below).	<p>Operation </p> <p>Each LDAP operation takes in a fixed product built-in data structure as input and for search operation. Additionally, an output data structure should be specified. These data structures are included in <b>Order Entry Product</b> library (See the Library tab in Velocity Studio). The following table illustrates the data structures required for each operation. These rules will be validated at runtime. <b>Note:</b> Misuse of input data structure will cause interface calls to fail and error messages will be displayed.</p> <p><u>Data Structures required for LDAP operations.</u></p> <table border="1"> <thead> <tr> <th>Operation</th> <th>Input</th> <th>Output</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>search</td> <td>cwf_oe.IdapSearchCriteria</td> <td>cwf_oe.IdapEntryArray</td> <td>Search for the entries that satisfy the criteria</td> </tr> <tr> <td>modify</td> <td>cwf_oe.IdapModificationItemArray</td> <td>N/A</td> <td>Modify (add, replace, remove) the attributes</td> </tr> <tr> <td>add</td> <td>cwf_oe.IdapEntry</td> <td>N/A</td> <td>Add a new entry</td> </tr> <tr> <td>delete</td> <td>cwf_oe.IdapDeleteRequest</td> <td>N/A</td> <td>Delete an existing entry</td> </tr> <tr> <td>modifyRdn</td> <td>cwf_oe.IdapModifyRdnRequest</td> <td>N/A</td> <td>Rename an entry</td> </tr> </tbody> </table>	Operation	Input	Output	Description	search	cwf_oe.IdapSearchCriteria	cwf_oe.IdapEntryArray	Search for the entries that satisfy the criteria	modify	cwf_oe.IdapModificationItemArray	N/A	Modify (add, replace, remove) the attributes	add	cwf_oe.IdapEntry	N/A	Add a new entry	delete	cwf_oe.IdapDeleteRequest	N/A	Delete an existing entry	modifyRdn	cwf_oe.IdapModifyRdnRequest	N/A	Rename an entry
Operation	Input	Output	Description																							
search	cwf_oe.IdapSearchCriteria	cwf_oe.IdapEntryArray	Search for the entries that satisfy the criteria																							
modify	cwf_oe.IdapModificationItemArray	N/A	Modify (add, replace, remove) the attributes																							
add	cwf_oe.IdapEntry	N/A	Add a new entry																							
delete	cwf_oe.IdapDeleteRequest	N/A	Delete an existing entry																							
modifyRdn	cwf_oe.IdapModifyRdnRequest	N/A	Rename an entry																							

### LDAP Attributes in Configuration Application

The table below describes the attributes that are exposed in Configuration application specifically for the LDAP Service Provider.

These attributes are found in [Services](#) tab in Configuration application. When clicking a Port, Binding, Operation or message node in the **Service Port tree**, the Configuration attributes for the node appears at the **Configuration** section at the right where values can be provided.

#### Attributes in Configuration application for the LDAP Service Provider

Attribute	Description	Element
<b>ldapURL</b>	Indicates the LDAP URL the interface wants to bind to (for example, <i>ldap://cw0001:389</i> ).	Port 
<b>protocolVersion</b>	Indicates what version of LDAP protocol is adopted on the server to which the interface calls on. When unspecified, the behavior is determined by the server (for example, v3)	Port 
<b>authentication</b>	Indicates the authentication method that the LDAP server uses. It accepts one of the following values: <ul style="list-style-type: none"> <li><b>simple</b> - uses weak authentication (clear-text password)</li> <li>a space-separated list of SASL mechanism names (for example, <i>CRAM-MD5</i>) means to use the CRAM-MD5 SASL mechanism described in RFC 2195 (<a href="http://www.ietf.org/rfc/rfc2195.txt">http://www.ietf.org/rfc/rfc2195.txt</a>)</li> <li><b>none</b> (or blank) - indicates no authentication is used (anonymous access)</li> </ul>	Port 
<b>principal</b>	User name that is used to connect to the LDAP server. When authentication is unspecified, this setting will be ignored (for example, <i>cn=Directory Manager</i> ).	Port 
<b>credentials_key</b>	Indicates the password key that is used with principal. When authentication is unspecified, this setting will be ignored. <b>Note:</b> The actual password needs to be set in Configuration application, Passwords, and associated with this key (for example, <i>LDAP_PWD</i> ).	Port 
<b>cacheSize</b>	The number of connections to be cached for the Service Provider. The default	Port 

value is 100 and is capped at this value. To disable connection pooling on LDAP, set cacheSize=0. Any value for this attribute that is greater than 0 means that a pool of that specified number of connections is created. For example, cacheSize=6 means that six connection pools are created.

For 0 connections, each connection is released after each use. For cacheSize=0, only one connection is created and used to access the LDAP server. After accessing the LDAP server, that connection is released (that is, removed). If the application needs to access the LDAP server again, a new connection is created, access to the LDAP server is available and is released again.

LDAP connections are closed under these conditions:

- After one hour of inactivity
- When the framework stops

#### LDAP Connection Pool Settings

You can specify the following JVM options as your Web container's JVM system parameters for LDAP connection pooling:

- com.sun.jndi.ldap.connect.pool.initsize
- com.sun.jndi.ldap.connect.pool.maxsize
- com.sun.jndi.ldap.connect.pool.prefsize
- com.sun.jndi.ldap.connect.pool.timeout

The following table provides a description of each setting and their valid values:

Operation	Description
com.sun.jndi.ldap.connect.pool.initsize	The string representation of an integer that indicates the number of connections for each connection identity to create when initially creating a connection for the identity. The default value is 1.
com.sun.jndi.ldap.connect.pool.maxsize	The string representation of an integer that denotes the maximum number of connections per connection identity that can be maintained concurrently. The default value is no maximum size that is specified
com.sun.jndi.ldap.connect.pool.prefsize	The string representation of an integer that indicates the preferred number of connections for each connection identity that needs to be maintained concurrently. The default value is that there is no preferred size.
com.sun.jndi.ldap.connect.pool.timeout	The string representation of an integer that denotes the number of milliseconds that an idle connection may remain in the pool without being closed and removed from the pool. The default value is that there is no timeout.

Examples of setting these parameters from a JVM command line are as follows:

```
-Dcom.sun.jndi.ldap.connect.pool.maxsize=20  
-Dcom.sun.jndi.ldap.connect.pool.prefsize=10  
-Dcom.sun.jndi.ldap.connect.pool.timeout=300000
```

**Note:** Each Web container has different ways to specify JVM options. Consult your Web container documentation for details.

## Set up LDAPS in OpenLdap Using OpenSSL

To set up LDAP over SSL (LDAPS) in OpenLDAP using OpenSSL, complete the following steps:

- [Server side](#)
- [Set up SSL in OpenLDAP](#)
- [Set up LDAPS URL in the System Administration application](#)
- [Start OpenLDAP](#)
- [Client side](#)

### Server Side

Using OpenSSL to create certificates, follow these steps:

1. Install OpenSSL in d:/OpenSSL.
2. You have the option of creating dummy folders under d:/OpenSSL/bin, to simulate a CA system. Create the following folder structure:
  - DemoCA
    - certs
    - newcerts
    - private
    - crl
  - index.txt
  - serial (with 01 inside)
3. Create a key file and certificate signing request for the server with the following command:

```
Openssl genrsa -des3 -out server.key 1024 -config "d:/OpenSSL/openssl.cnf"
Openssl req -new -key server.key -out server.csr
```

4. Sign server.csr with the following command:

```
Openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key
```

### Set up SSL in OpenLDAP

To set up SSL in OpenLDAP, complete these steps:

1. Open the slpad.conf file.
2. Edit the file by adding and change the following:

```
TLSCACertificateFile ./security/certs/ca.crt
TLSCertificateFile ./security/certs/server.crt
TLSCertificateKeyFile ./security/certs/server.key
```

### Set up LDAPS URL in the System Administration Application

To set up the LDAPS URL in the System Administration application, see [Set up an LDAP Provider](#). For step 5 of this procedure, enter your URL in the following form:

```
| ldaps://<host>:<port>
```

### Start OpenLDAP

To start OpenLDAP, use the following command:

```
| slapd -d 1 -h ldaps://<host>:636
```

### Client Side

To set up the client side, perform these steps:

1. Get the ca.crt file create by OpenSSL
2. Create and import a truststore file by using this command:

```
keytool -import -file "d:/ca.crt" -keystore ssl.keystore -alias ldaps
```

3. Add the following JVM option to your client program:

```
-Djavax.net.ssl.trustStore=<path>/ssl.keystore  
-Djavax.net.ssl.trustStorePassword=password  
-Djavax.net.debug=ssl
```

**Note:** Adding `-Djavax.net.debug=ssl` allows you to be able to debug JSSE.

## MQ Service Provider

The MQ Service Provider allows interactions with other systems through the IBM MQ Series Server. The MQ Service Provider supports connection queues only (no topics) for sending and receiving messages. A MQ Service Provider uses *Client MQ TCP/IP* transport when communicating with a MQ Series Server.

Note that the required MQ libraries must be made available to your application server (and Process Engine, if the Process Engine is issuing a response); see [Application Server Configuration](#) in Deployment Guide for details. For example, the library files of MQ can be added to the CLASSPATH of application server (or Process Engine) by

```
set MQ=C:\lib\com.ibm.mq.jar;C:\lib\com.ibm.mq.jmqi.jar;C:\lib\com.ibm.mqjms.jar;C:\lib\dhbcore.jar  
set Javax=C:\lib\connector.jar;C:\lib\jms.jar;C:\lib\jta.jar  
set CLASSPATH=%MQ%;%Javax%;....
```

Lastly, note that *com.ibm.mq.jmqi.jar* is only required for MQ 7.0.

### Important Note

MQ .jar files are not provided with the product. It is important to use the MQ7 jar files as they are backward compatible with MQ6.

## Invoking MQ Operations

When an MQ interface operation is invoked, a logical connection is initiated to a specific location. When completed, the connection is cached. If it is not used again within one hour (the time is not configurable), the connection is closed. The connection is not shared between threads. So, if interface operations are invoked with the same location simultaneously by different threads, extra connections are created and remain open for one hour, if they are not used. Additionally, if an error or exception occurs during the invocation of an operation, the connection is closed and is not cached.

## MQ Listeners

MQ listeners have a single logical connection created for each number of threads that are configured. During normal operation, that connection is open for the listener's lifetime. In the configuration, at the operation level, there is a **Distributed** flag. If this flag is set to true, the connection is shut down and an attempt is made to reconnect approximately every five minutes (the time is not configurable). The time is approximate, and depends on how long the listener has been waiting for a message and how many threads are defined. If multiple listener threads are defined, the time is staggered between them by 10 seconds so that not all threads are shut down at once. If any kind of error occurs in between establishing a connection and trying to receive a message, the connection is shut down after 600 attempts (the number of attempts is not configurable) and attempts to get a new connection. If an error occurs trying to obtain a connection, the listener waits for ten seconds (time is not configurable) before it tries again. These attempts continue indefinitely.

**Note:** MQ listeners are port based. That is, a port needs to be defined for each listener. Port settings for MQ listeners allow the choice of an operation meaning it can support multiple operations, however each MQ listener is tied to a specific operation. By default, the operation is the first operation in the interfaces operation list.

## MQ Attributes in Velocity Studio

The following table describes the attributes that are exposed in Velocity Studio specifically for the MQ service provider.

These attributes are found at **Binding** metadata object in Velocity Studio. When an Interface, Operation or message element is selected in the **Interface element** tree, the attributes for the element appears in the **Element definitions** table where values can be provided.

### Attributes in Velocity Studio for the MQ Service Provider

Attribute	Description	Element
<b>Correlate</b>	Gives options to <b>Correlate</b> request-response messages. If a selection is made here, there is no need to specify a <b>CorrelateProperty</b> (that is, the system automatically generates a sequence/key). <ul style="list-style-type: none"><li>• <b>msgID</b> - unique ID assigned by bus</li><li>• <b>sequence</b> - unique ID assigned by system</li><li>• <b>no</b> - no correlation (default value)</li></ul>	Output Message 

<b>CorrelateProperty</b>	User-specified <b>Correlate</b> property (for example, <code>msg.jms.correlationPropertyId = orderid</code> )	Message 
--------------------------	---	---

## MQ Attributes in Configuration Application

The following table describes the attributes that are exposed in Configuration application specifically for the MQ Service Provider.

These attributes are found in [Services](#) tab in Configuration application. When clicking a Port, Binding, Operation or message node in the **Service Port tree**, the Configuration attributes for the node appears at the **Configuration** section at the right where values can be provided.

[Attributes in Configuration application for the MQ Service Provider.](#)

Attribute	Description	Element
<b>Location</b>	Specifies the location of the MQ Series Server.	Port 
<b>QueueManager</b>	Specifies the name of the queue manager.	Port 
<b>Channel</b>	Specifies the channel to use. The default value is <i>none</i> .	Port 
<b>Port</b>	Specifies the port to use. The default value is 1414.	Port 
<b>cipherSuite</b>	Specifies the cipher suite algorithm to be used by MQ SSL. This parameter is sent to activate the connection factory.	Port 
<b>cacheSize</b>	The number of connections to be cached for the Service Provider. 100 is the default, and also capped to 100.	
<b>User</b>	Specifies the user name if the MQ Series Server requires authentication.	Port 
<b>PasswordKey</b>	Specifies the password key to use for retrieving the password. The password key/value pairs are created in the Configuration application.	Port 
<b>Transactional</b>	<b>Yes/No</b> - Request message will stay in queue until after receiving a confirmation response from receiving end. Only relevant for listeners and servers, and not clients. Please see note below.	Operation 
<b>ExpiryTime</b>	The duration, in seconds, that the message is in queue before expiry.	Operation 
<b>Priority</b>	A 10 level priority value with 0 as the lowest and 9 as the highest. Clients should consider 0-4 as gradients of normal priority and 5-9 as gradients of expedited priority. Priority is set to 4, by default.	Operation 
<b>ReplyTo</b>	Set where a reply to this message should be sent.	Operation 
<b>Queue</b>	Specifies queue name where messages are put into or retrieved from.  <b>Notes:</b> <ul style="list-style-type: none"><li>• For asynchronous messages that must come back to a specific queue, the value of Queue specified for the Output message is automatically appended into the input message header in the field <code>JMSReplyTo</code>. This action ensures the response comes back to the correct Queue.</li><li>• For listeners, if the inbound message has the <code>ReplyToQ</code> parameter, it is used by the product to post the response. Otherwise, if the message does not contain <code>ReplyToQ</code>, the product uses the output's <code>Queue</code> value from the configuration to post the response. The <code>ReplyTo</code> setting is never used for a listener.</li></ul>	Message 
<b>Timeout</b>	Number of seconds to wait when retrieving a message from the queue. Default is 1 second. If the <b>Timeout</b> is set at a value of 0 (zero), wait will be indefinite. Applies to output messages only.	Output Message 
<b>Properties</b>	Name of a global script that is invoked with the input data object. For sending messages, it returns a data object (Document, Order or Data Structure) the names and values of the data leaves of the returned objects are added as properties to the message. The Data Structure returned by the global script may contain specific child elements which will be used as properties for the message header. Specifically: <ul style="list-style-type: none"><li>• <b>time_to_live</b> - indicates the message expiry duration (in ms)</li><li>• <b>JMSPriority</b> - a value from 0 to 9 indicating priority (0 = lowest, 9 = highest)</li><li>• <b>JMSDeliveryMode</b> - value of 1 or 2 (1 = NON_PERSISTENT, 2 = PERSISTENT)</li></ul> For receiving messages (that is, for both client and listener), this field contains the Global function that returns a String expression. This expression's syntax is based on a subset of the SQL92 conditional expression syntax, such as <code>NewsType = 'Sports' OR NewsType = 'Opinion'</code> . Refer to JMS Message Selectors in the JavaScript documentation for details.	Message 

	<b>Note:</b> By default, a JMS message is set to PERSISTENT.	
<b>JMSClient</b>	This property indicates whether the remote application supports JMS. By default, this field defaults to <i>No</i> , which tells the MQ server that it should format the message for a non-JMS client. Set this property to <i>Yes</i> for input and output objects with the <b>Queue or Topic</b> field specified.	Message 

#### MQ Message Transaction Control (read)

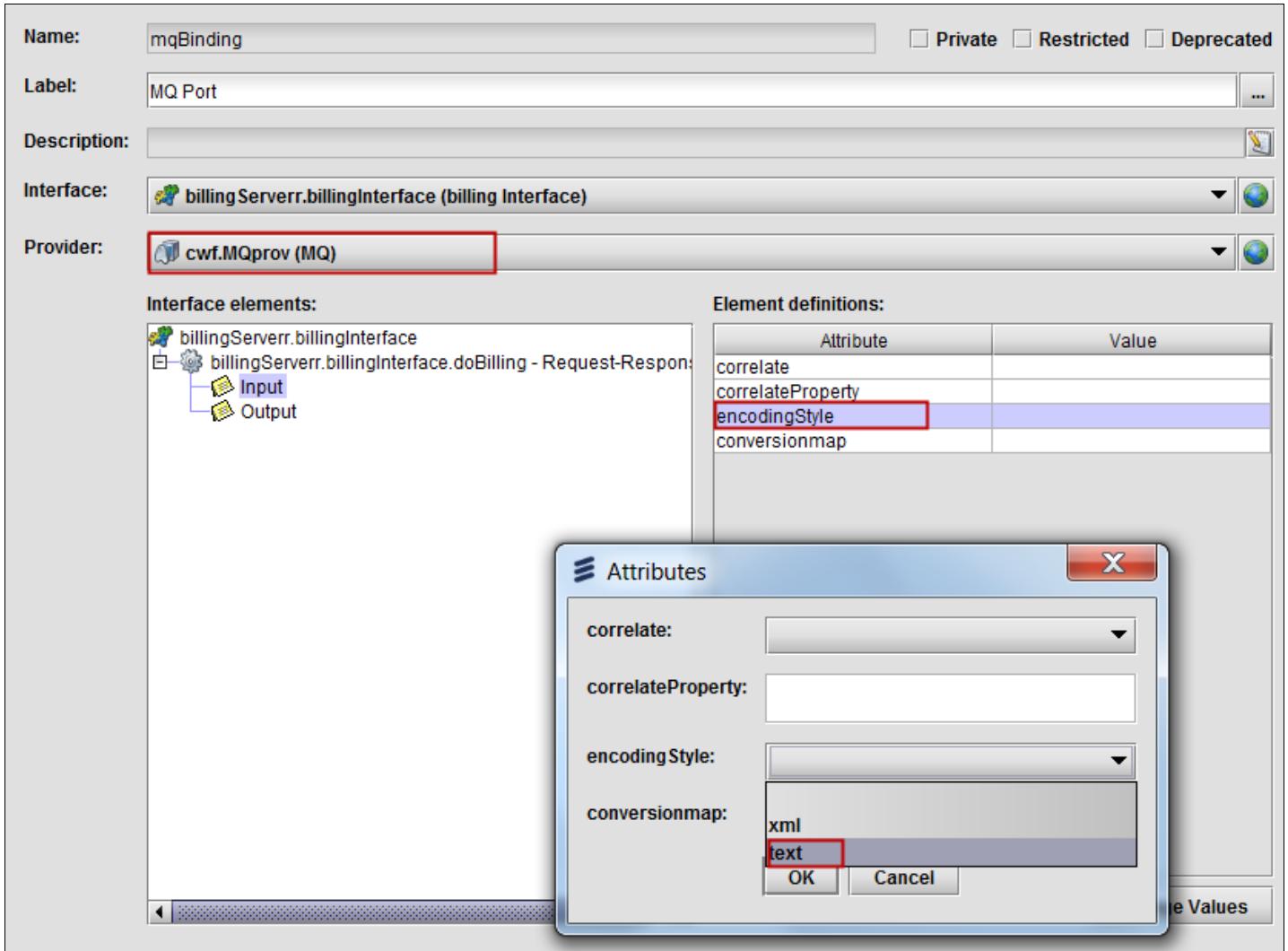
Transaction support is available only when an interface is used by the listener (incoming messages). This option, when set to yes, allows the listener perform message processing in transaction manner: If message processing encounters any errors (usually in script processing module), message will be put back on the queue for reprocessing. Message reprocessing parameters (that is, message reprocessing attempt, priority, delay, and so on are usually controlled by MQ Server configuration).

**Note:** For MQ, messages can be either plain text or XML.

## Implement an MQ Listener that Receives a Raw String

The MQ service provider allows you to implement an MQ listener that receives a raw string (that is, with no XML) by following these steps:

1. In your MQ binding, ensure that the **Provider** field is set to **MQ**.
2. Double-click the **encodingStyle** attribute to launch the Attributes dialog. Click the encodingStyle field's drop-down menu and select **text**. Click the **OK** button to continue.



3. Define the listener **Script** parameter as a string.

General Methods

Name: mqPort

Label: MQ Port

Description:

Listener

Script: <NONE>

Binding:  billingClient.MQBinding (MQ Binding)



4. Save your metadata changes.

## SMTP Service Provider

The SMTP Service Provider allows sending email messages over an SMTP connection. SMTP Service Provider operations are always *one-way* (since it only sends email message). Email message content is encoded as a XML document based on an input message Document or Data Structure.

### SMTP Attributes in Velocity Studio

There are no Binding Interface, Operation or message attributes that are exposed by the SMTP Service Provider, therefore, no attributes will appear in the **Element definitions** table of the Binding properties when an element is selected in the **Interface element** tree in **Binding** metadata object.

### SMTP Attributes in Configuration Application

The following table describes the attributes that are exposed in Configuration application specifically for the SMTP Service Provider.

These attributes are found in **Services** tab in Configuration application. When clicking a Port, Binding, Operation or message node in the **Service Port tree**, the Configuration attributes for the node appears at the **Configuration** section at the right where values can be provided.

#### Attributes in Configuration application for the SMTP Service Provider

Attribute	Description	Element
<b>Location</b>	Specifies the <i>send to</i> email address.	Port 
<b>SmtpServer</b>	Specifies the address of the SMTP server where the mail message is sent.	Port 
<b>Port</b>	Specifies the port to use. If not specified, the standard SMTP port 25 is used.	Port 
<b>ReplyAddress</b>	Specifies the <i>sender</i> email address.	Port 
<b>User</b>	Specifies the user name if the SMTP Server requires authentication.	Port 
<b>PasswordKey</b>	Specifies the password key to use for retrieving the password. The password key/value pairs are created in the Configuration application.	Port 

See the `sendEmailMessage` API in the JavaScript documentation for information on sending an e-mail message with optional attachments extracted from document attachments or files, and support for CC and BCC.

### Add an Inline Image in an E-mail

Support for including an inline image in an e-mail is available. The following is an example on using an inline image, `logo_example.png`, in an e-mail:

1. In `Global.setEmailMessage()`, create your HTML e-mail message body, which contains an image URL in the following format:

```

```

2. Pass the full path of `logo_example.png` as an attachment parameter of this API:

```
var attachInfoArr = new Array();
attachInfoArr = [null, "c:/temp/logo_example.png",null];
```

3. The body of your HTML e-mail message appears as follows:

```
var emailDoc_body = '<html>';
emailDoc_body += '<header>';
emailDoc_body += '<style type="text/css">';
emailDoc_body += 'body {font-family:Verdana,Bitstream Vera Sans,sans-serif; font-size:10px; }';
emailDoc_body += 'td.col1 {color:#0a6878; font-weight:bold; padding:5px; border:1px solid #616668; text-align:center;}';
emailDoc_body += 'td.col2 {color:#616668; font-weight:normal; font-size:12px; padding:5px; border:1px solid #616668; text-align:center;}';
emailDoc_body += 'h2.headerTwo{ font-weight:bold; color:#616668; }';
emailDoc_body += '</style>';
emailDoc_body += '</header>';
emailDoc_body += '<body>';
emailDoc_body += '<h1></h1>';
emailDoc_body += '<hr><p><h2 class="headerTwo">Order Management</h2></p><hr> <p></p>';
emailDoc_body += '<table>';
emailDoc_body += '<tr> <td class="col1" >Commercial Service ID</td> <td class="col2">IXS.12.11238</td>
</tr>';
```

```

emailDoc_body += '<tr> <td class="col1" >External Order ID</td> <td class="col2">PO.12.65432</td> </tr>';
emailDoc_body += '<tr> <td class="col1" >Offer</td> <td class="col2">CFSEWEB</td> </tr>';
emailDoc_body += '<tr> <td class="col1" >Task</td> <td class="col2">Change Routes/Protocols</td> </tr>';
emailDoc_body += '<tr> <td class="col1" >Customer</td> <td class="col2">Newport Municipality</td> </tr>';
emailDoc_body += '<tr> <td class="col1" >Project Manager</td> <td class="col2">Margaret Smith</td> </tr>';
emailDoc_body += '<tr> <td class="col1" >Contact Phone Number</td> <td class="col2">678854567</td> </tr>';
emailDoc_body += '<tr> <td class="col1" >Technical Manager</td> <td class="col2">Anthony Bell</td> </tr>';
emailDoc_body += '<tr> <td class="col1" >Contact Phone Number</td> <td class="col2">678456234</td> </tr>';
emailDoc_body += '<tr> <td class="col1">Offer Date</td> <td class="col2">23/05/2012</td> </tr>';
emailDoc_body += '</table>';
emailDoc_body += '</body>';
emailDoc_body += '</html>';
var attachInfoArr = new Array();
attachInfoArr = [null, "c:/temp/logo_example.png",null];
Global.sendEmailMessage("from_Email", "to_Email", "Email Subject", emailDoc_body, "text/html", "UTF-8", false,
attachInfoArr, "email user", "email password", "host", "port", true);

```

## STARTTLS Command

STARTTLS is an extension to the SMTP service that allows an SMTP server and client to use TLS (Transport Layer Security) to provide private, authenticated communication. This extension offers a way to upgrade a plain text connection to an encrypted connection, instead of using a separate port for encrypted communication.

By default, STARTTLS is enabled. For the SMTP session to enable TLS, mail.smtp.starttls.enable must be set to true.

To disable TLS if your e-mail server does not support this extension, use the following JVM option:

```
-Dmail.smtp.starttls.enable=false
```

## SOAP Service Provider

The SOAP Service Provider allows interactions with other systems through SOAP (Simple Object Access Protocol).

### SOAP Attributes in Velocity Studio

The following table describes the attributes that are exposed in Velocity Studio specifically for the SOAP Service Provider.

These attributes are found at **Binding** metadata object in Velocity Studio. When an Interface, Operation or message element is selected in the **Interface element** tree, the attributes for the element appears in the **Element definitions** table where values can be provided.

#### Attributes in Velocity Studio for the SOAP Service Provider

Attribute	Description	Element
<b>Style</b>	<p>Specifies if messages are RPC-oriented (data passed as parameters) or document-oriented (data passed as a XML document). This attribute can have one of the following values:</p> <ul style="list-style-type: none"> <li>• <i>rpc</i></li> <li>• <i>document</i></li> </ul> <p>The default value is <i>document</i>.</p> <p>The choice of style affects WSDL generation and formation of the SOAP message's BODY element in both SOAP requests and responses:</p> <ul style="list-style-type: none"> <li>• Use <i>document</i> when operation messages are defined as elements and you want full control in the metadata on constructing the SOAP message's BODY contents.</li> <li>• Use <i>rpc</i> when operation messages are defined as types.</li> </ul>	
<b>TargetURI</b>	Specifies the target URI of the SOAP Service. Usually the service name.	
<b>SOAPAction</b>	Specifies the SOAPAction HTML header for this message. When the value of this header is identical to the Interface Operation, it can be used to create the endpoint URL address as Port <i>location</i> + the Binding <i>SoapAction</i> .	
<b>Parts</b>	Accepts a single string value, or multiple values separated with by semicolons (";"). If specified, each value in the string will be considered a child tag name of the <i>input</i> data structure and thus extracted from the <i>input</i> data structure and as part of SOAP body element in the SOAP request. If left empty, the whole input data structure will be used in the SOAP body.	
<b>Header</b>	<p>Specifies what to set as the header of the SOAP request. Accepts a string value containing the header name and part, separated by a semicolon(for example, <i>hname;hpart</i>).</p> <ul style="list-style-type: none"> <li>• If <i>hname</i> equals the Data Structure name used as the input message, the input Data Structure will be taken as the base object for the header. Otherwise, the passed-in header parameter (that is, when calling <i>invokeInterface()</i>) will be used.</li> <li>• The <i>hpart</i> value works similarly to the <i>parts</i> setting (above). The product will try to find the child node in the base object whose name equals <i>hpart</i> and this child node will be set into the SOAP header.</li> </ul>	
<b>Use</b>	Specifies encoding rules of the message. Can have the value of <i>literal</i> (mainly for document style messages).	
<b>EncodingStyle</b>	Specifies the encoding style (URI) to use. Optional.	
<b>Namespace</b>	Specifies the Namespace to use for encoding. Optional.	

### SOAP Attributes in Configuration Application

The following table describes the attributes that are exposed in Configuration application specifically for the SOAP Service Provider.

These attributes are found in **Services** tab in Configuration application. When clicking a Port, Binding, Operation or message node in the **Service Port tree**, the Configuration attributes for the node appears at the **Configuration** section at the right where values can be provided.

#### Attributes in Configuration application for the SOAP Service Provider.

<b>Location</b>	<p>Specifies the location of the SOAP Service. This URL takes the form</p> <div style="border: 1px dashed #ccc; padding: 5px; margin-top: 10px;"> <pre>http://host:port/cwf/services/&lt;interfaceName&gt;</pre> </div> <p>where <i>interfaceName</i> specifies the metadata Interface object implementing this service.</p>	
<b>User</b>	Specifies the user name if the SOAP Server requires authentication.	
<b>PasswordKey</b>	Specifies the password key to use for retrieving the password. The password key/value pairs are created in the Configuration application.	
<b>Reuse session</b>	If set to 'no', a new user context is created for each request. Absence of this attribute is defaulted to yes, in which the user context will be reused. Used by listener ports only.	
<b>Maintain session</b>	Used by client ports only. If set to 'no', client session will not be maintained (that is, Cookie will not be reused). Absence of this attribute is defaulted to yes, in which the session between clients requests may be maintained.	

<b>Cache size</b>	The number of connections to be cached for the Service Provider. 100 is the default, and also capped to 100.	Port 
<b>Disable Chunked Encoding</b>	This attribute is only valid for SOAP clients, and not listeners. When you select this checkbox, the SOAP request sent when invoking a SOAP interface is not chunked. In the sent properties in the message log, the Content-Length header appears. Otherwise, leaving this checkbox unchecked indicates that the SOAP request is chunked. In the sent properties in the message log, the Transfer-Encoding header contains the <i>chunked</i> value.	Port 
<b>Timeout</b>	Number of second to wait for the response. The default is 60 seconds.	Operation 

#### Notes:

- When there is a timeout, the SOAP interface retries connecting exactly one more time.
- After a SOAP request is complete, the HTTP connection is left in ESTABLISHED state and is available for reuse by another request to the same host.

#### Deposit SOAP-formatted messages to an MQ Queue

You can use the following APIs to deposit messages that have been formatted in SOAP to an MQ queue:

- `Global.generateInterfaceMessage(interfaceName, operationName, params)`  
This function behaves the same way as `Global.invokeInterface()`, except that it does not call the interface. Instead, the function generates the message content and returns it as a string.
- `Global.sendInterfaceMessage(interfaceName, operationName, stringInput)`  
This function calls the interface with the original string input.
- `Global.sendInterfaceMessageUsePE()`  
This function is similar to and works with `Global.sendInterfaceMessageUsePE()`. This function spawns from `Global.invokeInterfaceUsePE()`

#### SOAP Faults

SOAP faults have the following standard fields:

- `faultcode`
- `faultstring`
- `faultactor`
- `detail`

The child nodes of the fault `detail` element must be elements. There can be more than one. The following is an example:

```
<detail>
  <errorname>...</errorname>
  <stacktrace>...</stacktrace>
</detail>
```

Previously, when a SOAP fault was received, an attempt occurred when mapping the first detail element to the fault message structure provided in the operation metadata. As a result, the `faultcode`, `faultstring`, and `faultfactor` information was lost, including any additional detail elements after the first one.

To access this information, the fault metadata object specified in the operation must have the following child nodes:

- `faultcode`: String datatype
- `faultstring`: String datatype
- `faultactor`: String datatype
- `detail`: Container (its children must be elements)

If the fields with the `fault` prefix are found, the product maps the fault information to the appropriate fields. Otherwise, the product defaults to the previous behaviour.

#### Notes:

- This feature is only available for SOAP 1.1 messages.
- Returning fault messages for SOAP listeners is only supported with the `Global.throwSoapFaultException()` API. Refer to the API document for more information.

#### Receiving SOAP Data

For the SOAP service provider, when receiving SOAP data as either a response when invoking a SOAP interface or as a request to a SOAP listener, the data that is recorded in **message log received** data field does not always reflect exactly what was actually transmitted to the product. One common case occurs when the received SOAP requests contains CDATA sections. In the message log, text nodes that were wrapped in CDATA have their content escaped with entity references. From the product's perspective, it makes no difference whether the data transmitted was wrapped with CDATA or had its text escaped in the actual request. Semantically, the value of the text is the same and any product script reading the value from a data structure receives the same value.

## Set up a SOAP Service Listener

To set up the SOAP service listener, complete these steps:

1. Create an **Interface** with all the required Operations of type *one-way* or *request-response* (Select **One-Way** or **Request-Response** in the **Operation type** combo box).
2. Create a **Binding** that binds this Interface to the SOAP service provider (Select the Interface in the **Interface** combo box and the **SOAP Service Provider** in the **Provider** combo box).
3. Define the **Interface element** attribute values of the **SOAP Binding** properties. The SOAP listener expects all messages encoded as Documents and the Action URI should be set to the Operation name to be invoked (matching the Operation name of the Interface).
  - o Select the *Interface*  element in the **Interface element** tree and set the **style** attribute value in the **Element definitions** table to *document*, and the **targetURI** attribute value to the Operation name to be invoked.
  - o Next, select the *message*  element in the **Interface element** tree and set the **use** attribute value in the **Element definitions** table to *literal*.
4. Create a **Service Port** for this Binding and set it as listener port (Check the **Listener** check box of the Port properties).
5. Create a Binding that binds the same Interface to the JS (JavaScript) service provider (Select the Interface in the **Interface** combo box and the **JS Service Provider** in the **Provider** combo box).
6. Implement the Interface Operation script functions in the **JavaScript Binding** (Select the *Operation*  element in the **Interface element** tree of the Binding properties, and define the **Script** attribute value in the **Element definitions** table).
7. Create a Service Port for this JS Binding (Select the JS Binding in the **Binding** combo box).
8. Save and then run your metadata.
9. In the Configuration application, this Port can be found and configured at the **Services** and **Listeners** horizontal tab of the **Services** vertical tab.
10. The AVM will set up this service under the following URL: <http://host:port/cwf/services/InterfaceName>.

When running the metadata, a WSDL file for each SOAP Interface can be accessed by the following URL:

<http://host:port/cwf/services/InterfaceName?wsdl>.

**Note:** When an incoming message is received by the SOAP service, it calls the corresponding JavaScript function using the action URI as a requested Operation name. If action URI is not present it uses the SOAP message operation name.

## SOAP Encoding and RPC Import Support

The SOAP Service Provider provides SOAP (Simple Object Access Protocol) encoding and basic RPC import support, which does the following:

- Prevents you from experiencing unnecessary errors or file changes when importing WSDL or XSD due to either the WSDL or XSD containing types that make use of SOAP-encoded arrays, or RPC-style bindings or operations
- Provides runtime support for XML formatting of arrays in SOAP-encoded messages when creating SOAP messages

### Import a SOAP-encoding Schema

If a WSDL or XSD imports the SOAP-encoding schema (<http://schemas.xmlsoap.org/soap/encoding/>), it is no longer necessary for you to change the WSDL or XSD file to specify the schemaLocation attribute for schema if and only if the WSDL or XSD makes use of SOAP-encoded arrays. In this case, all types are imported without error and no namespace is created for the SOAP-encoding schema.

If the WSDL or XSD makes use of SOAP encoding other than for arrays, inclusion of the schemaLocation attribute is required. If the WSDL or XSD already has the schemaLocation attribute, importing that schema proceeds as normal, with errors occurring for schema resolution and for unsupported content. Additionally, a namespace appears in the metadata for that schema.

### Data Structures with SOAP-encoded Array Elements

Data structures that are created from complex types with SOAP-encoded array elements are formatted to match the expected XML output structure in a SOAP message.

### Use Property in Metadata Bindings

The **use** property of operation messages in the metadata bindings is set to **encoded** for SOAP-encoded messages. If the **use** property is set to **encoded** in the metadata bindings for an operation message, any arrays in a SOAP message have the **arrayType** attribute dynamically added, specifying the XSD type of the array instances and indicating how many array instances there are. The following is an example:

```
<BODY>
<ArrayOfOrders soapenc:arrayType="cwf:Order[2]">
<cwf:Order>....</cwf:Order>
<cwf:Order>....</cwf:Order>
</ArrayOfOrders>
</BODY>
```

### Messages with RPC-style Operations or Bindings

In WSDLs, messages used by operations or bindings whose style is RPC have data structures created for the operations that more closely match the expected structure in a SOAP request.

### Notes

SOAP-encoding multi-references are not supported.

## **SOAP WSS (Web Services Security) Support**

The SOAP service provider supports specification of a policy file conforming to the W3C WS-Policy and WS-SecurityPolicy specifications. Understanding of these specifications is highly recommended before using these settings, to allow Web services to express their constraints and requirements, such as encryption, message signing, timestamping, and use of authentication tokens in SOAP messages.

### **SOAP Port Security Configuration Settings (SOAP Service Provider)**

Basic operations such as signing, encryption, user token authentication are supported with additional configuration values. See the table that follows.

#### **Restrictions and limitations:**

- If a user name token is provided in a message to a SOAP listener, the user is authenticated against product user accounts. Similar to how HTTP authentication is handled.
- Signing and encryption certificates must be located in the same Java JKS keystore file. The location of the keystore must be specified with a system property.
- The keystore and certificate passwords must be the same.
- For encryption, only one certificate can be referenced per policy.
- For signing, only one certificate can be referenced per policy.
- For username tokens, only one user can be referenced per policy.

#### **Required System Properties:**

- org.apache.ws.security.crypto.merlin.file: Location of Java JKS keystore file.
- org.apache.ws.security.crypto.merlin.keystore.password: Java JKS keystore password

The following are fields that you can use to configure your port settings for SOAP WSS:

Attribute	Description	Element
<b>Policy File</b>	This field represents an XML file conforming to the W3C WS-Policy specification. Can be set for both client and listener ports.	Port 
<b>Username Token User</b>	This field represents the username value for the authentication token if it is required by policy file.	Port 
<b>Username Token Password</b>	This field denotes the password value for the authentication token if required by the policy file.	Port 
<b>Signing Certificate Alias</b>	This field contains the Java keystore alias for either the certificate or private key used to sign parts of a message.	Port 
<b>Encryption Certificate Alias</b>	This field represents the Java keystore alias for either the certificate or public key used to encrypt parts of a message.	Port 

## Socket Service Provider

The Socket service provider allows sending and receiving data over sockets. It is used for internal purposes only.

### Socket Attributes in Velocity Studio

There are no Binding Interface or message attributes that are exposed by the Socket service provider. Therefore, no attributes will appear in the **Element definitions** table of the Binding properties when an element is selected in the **Interface element** tree in **Binding** metadata object.

The Socket Service Provider uses the Interface Operation's Type. Its behaviour based on operation type is as follows:

- **One-Way:** The Socket provider opens a socket, sends the request data, and then closes the socket. It does not wait for any kind of acknowledgement from the remote socket.
- **Request-Response:** The Socket provider opens a socket, sends a request date, reads the response data, and then closes the socket. The provider blocks (with no timeout) after sending a request until the remote socket starts sending data.

**Note:** Notification and Solicit responses are not supported.

### Socket Attributes in Configuration Application

The following table describes the attributes that are exposed in Configuration application specifically for the Socket service provider.

These attributes are found in **Services** tab in Configuration application. When clicking a Port, Binding, Operation, or message node in the **Service Port tree**, the Configuration attributes for the node appears at the **Configuration** section at the right where values can be provided.

#### Attributes in Configuration application for the Socket Service Provider

Attribute	Description	Element
Location	TCP/IP server name.	Port 
Port	TCP/IP port to use.	Port 

## TIBCO Service Provider

The TIBCO Service Provider allows interactions with other systems through TIBCO Rendezvous™ Server. The TIBCO Service Provider sends all messages as a XML string and expects incoming data to be a XML string as well.

### Notes:

- The required Tibco libraries must be made available to your application server (and Process Engine, if the Process Engine is issuing a response); see [Application Server Configuration](#) in Deployment Guide for details. For example, the library files of Tibco can be added to the CLASSPATH of application server (or Process Engine) by

```
set TIBCO=C:\lib\tibrvj.jar;C:\lib\tibrvjsd.jar;C:\lib\tibrvtxj.jar;
set CLASSPATH=%TIBCO%;....
```

- The Rendezvous version (for example, 7.2 or 7.4) is decided by the version of tibrvj.jar that is used.
- When using TIBCO with JMS, the jms.jar file is delivered with TIBCO client libraries and should be used.

### TIBCO Attributes in Velocity Studio

The table below describes the attributes that are exposed in Velocity Studio specifically for the TIBCO Service Provider.

These attributes are found at **Binding** metadata object in Velocity Studio. When an Interface, Operation or message element is selected in the **Interface element** tree, the attributes for the element appears in the **Element definitions** table where values can be provided.

[Attributes in Velocity Studio for the TIBCO Service Provider.](#)

Attribute	Description	Element
<b>Subject</b>	Specifies the subject name for sending/receiving messages.	Message 
<b>TimeOut</b>	Specifies the time out interval when waiting for a response (in seconds). The default wait is indefinite.	Message 
<b>Fieldname</b>	Specifies the name of the field where the message data (XML string) is stored.	Message 
<b>Message</b>	<b>yes/no</b> - If yes, the message is in Tibco Rendezvous format, also known as (Active Enterprise) AE 3.0.	Message 

### TIBCO Attributes in Configuration Application

The table below describes the attributes that are exposed in Configuration application specifically for the TIBCO Service Provider.

These attributes are found in **Services** tab in Configuration application. When clicking a Port, Binding, Operation or message node in the **Service Port tree**, the Configuration attributes for the node appears at the **Configuration** section at the right where values can be provided.

[Attributes in Configuration application for the TIBCO Service Provider.](#)

Attribute	Description	Element
<b>Service</b>	Specifies the TIBCO Server service.	Port 
<b>Network</b>	Specifies the TIBCO Server network.	Port 
<b>Daemon</b>	Specifies the TIBCO Server daemon.	Port 
<b>User</b>	Specifies the user name if the TIBCO Server requires authentication.	Port 
<b>PasswordKey</b>	Specifies the password key to use for retrieving the password. The password key/value pairs are created in the Configuration application.	Port 
<b>Transactional</b>	Specifies if the Rendezvous™ TX service is used that provides message persistence.	Operation 

## External Configuration

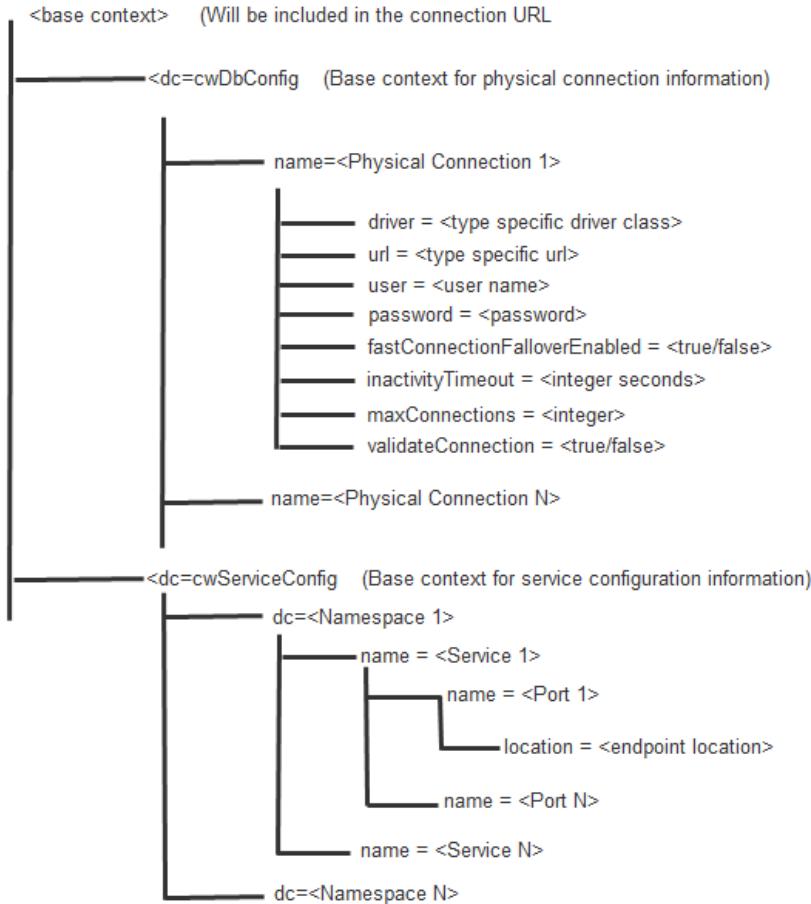
### System Parameters

Default values for portions of the cluster and node configuration can be read using JNDI allowing for configuration to be taken from an external source and configuration of an entire system to be centralized. That source must conform to the product naming convention. The following configuration elements can be configured externally:

- Physical database connections
- Interface port locations

### LDAP

LDAP sources must conform to the following schema:



Field	Cluster Level	Node Level	Description
Type	Editable	N/A	JNDI Type name. Currently LDAP is supported.
URL	Editable	N/A	JNDI type specific URL.
Username Token Password	Editable	N/A	JNDI type specific identifier
Signing Certificate Alias	Editable	N/A	JNDI type specific password for specified principal

# Introduction to Business Processes

The Process Management product provides a metadata-driven mechanism for defining business processes. The product uses the BPML (Business Process Modeling Language) model for describing business processes.

## What is a Business Process?

---

A *Business Process* is an interaction between workflow and its participants, and an execution of workflow activities according to a defined set of rules to achieve a common goal.

## Process Components and Terminology

---

- **Activity** – A component that performs a specific function within the process, such as invoking a service or another process. Activities can be as simple as sending or receiving a message or as complex as coordinating the execution of other processes and activities.
- **Atomic activity** – An elementary unit of work that cannot be further decomposed. Atomic activities can be used to start other processes, perform calculations, or perform operations.
- **Complex activity** – Composed of other activities and directs the execution of these activities. The complex activity instructs these activities to execute in sequential order or in parallel, to execute once or repeatedly, or even whether to execute or not conditionally.
- **Participant** – An entity with which the process interacts. Participants in a process are varied and include external services, users, partners, and other processes.
- **Interface Operation** – An abstract description of an action supported by the [external service](#) or participant.
- **Interface** – An abstract set of [operations](#).
- **Operation** – A simple activity that performs a message exchange with a process participant (invokes an interface operation).
- **Order Instance** – In the database each Order is represented as an *Order Instance Document*. The Order Instance Document can be extended and overridden by a the user's application, if necessary (in the Library tab, *Order Entry Product* namespace).
- **Process Revision** - an archived, read-only version of a Process metadata object that was then "current" process metadata before changes were made to it and revisioned. Among other purposes, it provides the accurate snapshot of the Process definition for "in-flight" process instances (that is, process instances that have started but not completed) to continue execution upon process metadata change. The terminology may also refer to the *Process Revision number* of the Process metadata object, which keeps track of the numbering in the latest revision.
- **Exception handling** – Defines activities that respond to an unexpected event. An exception event could be a time-out or an exception activity.
- **Alert** – Defines a destination for a notification mechanism. An alert destination could be an email or pager address, a user or an external service.
- **Process Document** – Serves as a process's local variables that helps process decision-making (process flow) and activities coordination. The total size of all data in a process Document cannot exceed 4000 characters. This limitation is enforced by the Process Management product to optimize database access (the process Document is stored as part of the process information and should not be mapped to database tables).

If a process is using a process document that is mapped to the database, the process document will be stored in a separate database table. Otherwise, an unmapped process document is stored with the process.

- **Process Order** – The Order instance that a process is working on. Each process instance can be associated with only one Order instance.
- **Process Engine** – The Process Management software module that reads the process definition (metadata) and performs process activity execution. Depending on hardware configuration, a single Process Engine is capable of executing tens of thousands of processes simultaneously. Several Process Engines can run concurrently, providing scalability, load balancing and fault tolerance.
- **Privilege** – A user right which is assigned to a user and that specifies allowable actions in the system.

## Main Features of Process Management

---

- Capable of running multiple Process Engines simultaneously.
- Process Engine can run on any [supported J2EE platform](#).
- Load balancing mechanisms across Process Engines.
- Recovery of processes that were running on a failed/stopped Process Engine.
- Automatic notification of failed processes (pager, email, worklist or external system).
- Low hardware requirements for Process Engine (as low as 1GHz/512MB server).
- Single Process Engine running on low-end server capable of performing over 100 process transitions per second.
- Ability to run specific processes on dedicated servers.
- Ability to dedicate servers for external system interfaces (so it runs on specialized hardware/software configuration).
- Ability to communicate asynchronously to any external system.
- Ability to specify external system availability and buffer all requests transparently.
- Available APIs for integrating with external systems.
- Ability to run listener (global) processes that are dedicated to specific tasks.
- Ability to assign process priority.
- Ability to define and handle jeopardy situations (business exceptions).
- Ability to synchronize between parallel tasks within a process and between processes (process signals).
- Ability to define and use business calendars for task durations, task schedules and systems availability.
- Ability to define business milestones to measure process efficiency.
- Integrated worklist functionality (manual tasks) with powerful user interface.
- Powerful UI designer for creating process workflow visually.

## Metadata Elements Used in Process Management Product

---

The following metadata elements can be defined in the Velocity Studio to support Process Management:

- *Process*: Metadata element that defines a business process and all process activities (activities diagram).
- *Participant*: Metadata element that defines a process participant.
- *Alert*: Metadata element that defines an alert destination.
- *Exception*: Metadata element that defines an exception code.
- *Signal*: Metadata element that defines a signal type.

In addition to the metadata elements listed above, you can also define scripts to provide additional functionality.

# Process Types

Two types of processes may be defined in the Process Management product: *Global* and *User*.

## Global Processes

A Global Process is usually created to provide certain specific services to other processes (such as automated credit approval) or to perform recurring automated tasks (such as processing batch Orders from a channel partner).

Global Processes are started automatically by the Process Engine. It is possible to configure the Process Management product to run more than one instance of the same Global Process (see [Configuration application: Processes](#)). If more than one Global Process of the same type is running, the processes would share the workload of incoming requests (for increased throughput). It is also possible to configure the Process Management product to run specific Global Process types only on specific computers that run a Process Engine; this allows for better resource management or special hardware requirements (like SNA gateway for mainframe connectivity).

Every Global Process defines the Interface used by processes that want to invoke the services of this Global Process. The Interface used by a Global Process should not have Binding and Port details (see [External Services](#)).

**Note:** Global processes cannot create user subprocesses. Attempting to create a subprocess as a child of a global process fails and an exception is thrown.

## User Processes

A User Process is a process that is designed to complete one specific job (such as Order fulfillment). Such a process will be usually, but not necessarily, associated with one Order instance. A User Process's life span depends on the complexity of the job this process is designed to perform, and can be anywhere from a few hours to a few months. Complex User Processes can further delegate part of the work to child sub-processes (refer to section [Signal](#)). User Processes are started manually from the Menu, in a Decision Tree or Script (see [Starting User Processes](#)).

## Main Differences Between Global and User Processes

- Global processes are started automatically by the Process Engine.
- Global processes can run on a dedicated computer(s).
- Global processes usually perform the same task over and over.
- User processes can be associated to specific Order instances.
- User processes cannot be used as a Participant (does not provide services to other processes).

## Interface Processes

Process Management product automatically defines special types of processes, called *Interface Processes*. Interface processes perform interactions with external services that are used in global or user processes. Process Management system creates one interface process type for each external service used in global or user processes. Interface processes can be considered a special type of global process. As with global processes, interface processes are started automatically by the Process Engine, and can be configured to run on specific computers and have several instances of the same interface process type.

In this approach, all requests from global and user processes (Interface Operations) to specific external services are sent to the interface process that serves this interface. The interface process performs these requests sequentially, where requests from processes with higher [priority](#) are served first. When higher throughput is required, more instances of specific interface processes can be configured.

The interface process will find a currently available [port](#) that implements this interface based on the [port calendar](#). If no available port is found, the interface process will delay external service invocation until the nearest available time based on the port calendar. This feature allows for the control of when, and which, external service is available for use.

## Memory-Based Processes

---

Memory-based processes enable all three types of processes (user, global, and interface) to be executed in memory only without storing process states in the databases. Although these processes can still be run in standard mode, memory-based processes do not store their states in the database (only the first activity with initial timestamp is stored). If the process is restarted, the system will start again from the first activity. This feature is a performance enhancement and is useful for simple global processes (for example, interface processes), but may also need memory-based user processes.

### Velocity Studio:

- To set global and user processes as memory-based, navigate to process property panel and set flag to Memory only. Each time the process is set as memory-based, a warning appears stating that process will not persist its state

### Configuration Application:

- To set interface process as Memory Only, navigate to Configuration and click the **Processes** tab. Set the parameter **Store interface processes' states** to *false*. By default, this parameter is set to true.

#### Important Note:

Resume and Rollback activities not allowed within memory processes as they require database persistence of process activities.

### Runtime:

- Process Manager and Global Process Manager do not display memory-based processes.
- As memory processes do no persist their states and previously performed activities, user can only view a "snapshot" of memory processes that is currently available in the memory cache (that is, user can view running processes and their activities).

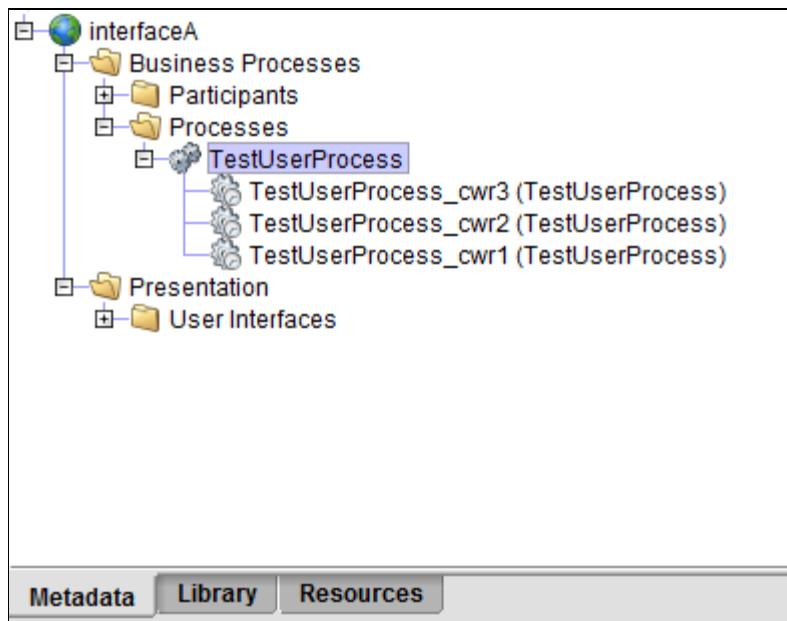
To view memory processes, navigate to Administration > AVM > Process Engines. Select one Process Engine and click the **Memory Processes** menu. A new finder similar to Process Manager will appear displaying currently running memory processes (user, global and interface) on the selected process engine and details available from memory. User can further view current activities by selecting the process and clicking Activities menu. Once again, only a snapshot of activities will appear in the finder.

## Process Revisions

A **Process Revision** is an archive of User Process metadata object, a "snapshot" taken of the Process metadata object, after which the Process metadata may be changed (that is, changes made to Process Activities in Process). As a Process metadata object is changed over application development cycles, the "latest" process metadata may contain many Process Revisions.

Process Revisions are the same type as Process metadata. Each Process Revision is a read-only Process.

In Velocity Studio, Process Revisions can be found under their **Process** node in the **Metadata** tab of the **Navigation** pane, after you click the **Show Revisions** command in the right-click menu of the Process node. Process Revisions are read-only metadata objects.



With Process Revisions, every runtime process instance can be executed to completion with the Process metadata that it was instantiated with. This is the case, and the default behaviour if no migration is defined, even if Process metadata is changed over the span of Process execution (which may take days or months). This feature is commonly referred to as *grand-parenting*.

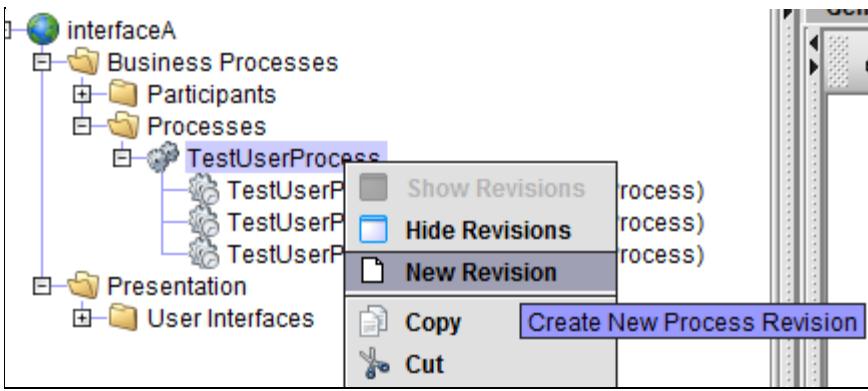
Process Revisioning is only available for User Processes, but not to Global Processes. Existing Global Process instances are restarted by Process Engine startup with new Global Process metadata. Specifically, if revision number of Global Process instances in database differs from its metadata's revision number, they are removed at PE startup and then new Global Process instances are started with the current Process metadata. When PE starts it checks whether global process metadata is in revision # different from the revision # of the process instance in database. If so, it deletes previous global process record from database. Next, PE thread finds that there is no record in the database for the given process and starts process with latest metadata. In simple words, if global process database record revision # differs from metadata revision #, PE re-starts global process on the new metadata.

### Creating Process Revisions

Process Revisions are generated by the user, manually and individually for each process. The user is in full control of what revisions are needed and when to generate them.

Before working on a process, the user must first generate a new revision to solidify the previous workflow in a revision object. The user "acquires" the process in a new revision for his modifications and should continue working on the process in the same revision number until all changes are done. The user can deploy and run the process as many times as needed. However, it is recommended that the user stays with the same revision number until all the desired changes are done. This way, all modifications can stay within the context of a single revision without leaving multiple non-used revision objects behind.

To create a Process Revision, right-click the Process and select **New Revision**.



## Removing Process Revisions

Over time, many old Process Revisions may accumulate under a Process. These old Process Revisions can be safely [removed](#) from your application metadata as long as there are no more process instances that require them. You should look into your process instances and their corresponding [process status](#) in the target environment. From a runtime perspective, a Process Revision is no longer necessary to Process Engine if Process metadata has defined [migration record](#) on that revision, even though you may want to continue archiving it for informational purpose. In general, it is recommended that you keep those Process Revisions that still bear business process significance to your organization.

## Process Revisioning in Runtime

In deployment environment, all Process metadata objects are guaranteed to have the same definition as its latest Process Revision, thanks to [packaging](#).

For new process instances, they are labeled with the latest Process Revision number upon instantiation and executed according to that revision.

For existing process instances, [Process Migration](#) may occur when processes are started or resumed. The Process Instance's labeled Process Revision number is compared with the **Revision** property of the Process metadata:

- If the two numbers are the same, their execution remains unchanged.
- If the instance's number is less than its Process metadata Revision number, then
  - the process instance executes at its Process Revision if no [migration record](#) for this revision number is defined in the Process metadata (that is, *grand-parenting*), or
  - executes as per migration record if defined (may it be *suspend*, *terminate* or *migrate*).
  - if neither migration record or its Process Revision is present, the process instances goes into [error status](#).

Furthermore for existing process instances, the migration record in Process Revisions are not relevant to process migration; only migration record of the current Process metadata is executed.

## Analyze Processes Tool

The Analyze Processes tool allows you to compare and check the compatibility between the local process metadata and the deployed process metadata. It is highly recommended that you perform the comparison prior to deploying metadata, especially in production, to ensure metadata compatibility and to avoid runtime process exceptions. To read more about the tool, see [Analyze Processes](#).

## Process Pinning

---

Metadata can have multiple versions of the same process that are not in use and are of no interest in your current project, such as old process reincarnations, different process versions, non-applicable process versions, and so on. These metadata versions are regular processes that already exist, must remain in the metadata as-is, and have the ability to run in the process engine, if required. However, they overpopulate the Process folder and you may choose to temporarily hide them.

The key is to be able to logically organize processes by moving unused process versions under the main process folder. When needed, you can view these versions by accessing the process folder and hide them by closing it. Additionally, you must be able to move these processes back to their original location. These actions are called **process pinning** and **process unpinning**, respectively. The old version that is pinned under the process is called the **process pin**, and can be unpinned or moved back to its original location.

**Note:** Having multiple versions of the same process that are not in use is different than creating a [process revision](#). The latter refers to an archive of the user process metadata object, which is a snapshot taken previously by Velocity Studio of the Process metadata object and can be changed later.

### Process Folder

The Process folder contains two types of metadata:

- Process revisions
- Pinned processes

Each process in the Velocity Studio acts as folder which, by default, is closed and hidden from users. If this folder is empty, you cannot open it.

#### Open a Non-empty Folder

To open a non-empty folder, choose from one of the following options:

1. Double-click the process.
2. Right-click and select **View Process Folder**.

#### Close a Folder

To close a folder, do one of the following:

1. Double-click the process with an open folder.
2. Right-click and select **Hide Process Folder**.

Refer to the [Process Revisions](#) section to see how revisions are generated and used.

### Pin Process Versions

To pin process versions to the parent process, complete these steps:

1. Select the processes to be pinned, and either select **Copy** or **Cut** them, to place these processes on the clipboard.
2. Select the process pin, right-click, and then select **Pin to Process**.
3. Pinned processes are no longer displayed under the Process category. However, you can see them in the Process pin folder.

**Note:** Processes can be pinned to a Process element only (that is, the **Pin to Process** option is only available in the right-click menu of the Process element). This option is unavailable in either the Process Revision or any other metadata element.

The **Pin to Process** option is disabled when:

- The clipboard is empty.
- The process pin is non-editable.

- The process pin itself is pinned to another process to avoid multi-level pinning.
- One of the elements on the clipboard is not a process (for example, a process revision *is not* a process).
- One of the clipboard processes has other processes pinned to it to avoid multi-level pinning.
- One of the elements in the clipboard is outside of the process pin's namespace. In Velocity Studio, you cannot identify which namespace the process belongs to unless it is displayed within the namespace. If you pin a process to a process pin belonging to another namespace, you cannot identify the original namespace of the pinned process. This restriction does not pose a problem, as you are working with different versions of the same process that are mostly located inside the same namespace.

## Unpin Process Versions

To unpin process versions back to their original location, do the following:

1. Select the processes to be pinned, right-click, and then select **Unpin from process**.
2. The selected processes no longer display in the process pin folder. Instead, they return to the original location (that is, in the Process category folder in their respective namespaces).

The **Unpin from process** option is disabled when:

- No process to unpin is selected.
- One of the selected elements is not a process.
- One of the processes is not pinned

## Examples of Valid Operations

The following are examples of using pinned processes:

- You can select pinned processes across multiple process pins in different namespaces and unpin them in one action.
- You can select pinned and unpinned processes in the same namespace, and pin them to a third process in one action.

## Considerations

When the Process pin folder is hidden and you click the pinned process in either the **Search** or **Problems** panel, the pinned process panel displays and the Velocity Studio tree is expanded properly. However, the process node in the Velocity Studio tree is not shown as selected.

## Sub-Process and Sub-Flow

---

Because business processes in telecom are often complex and multi-facet, it is often impossible to represent and manage an end-to-end business process with a single [Process Diagram](#) in a Process metadata. Instead, with a divide-and-conquer approach, [process activities](#) of a complex business process are often organized into sub-processes or sub-flows, each to encapsulate a particular business process flow, and then invoked using [sub-process activity](#) or [sub-flow activity](#) respectively by its main process diagram.

This article illustrates the differences between sub-processes and sub-flows.

### Sub-Process

There are two types of sub-processes:

- *Asynchronous sub-process* - sub-process and main process run simultaneously in Process Engine. This is the default type.
- *Synchronous sub-process* - main process waits for the sub-process to complete before continuing to next activity.

This is configured with the boolean **Synchronization** flag in the [sub-process activity](#).

The CWPRODUCTPROPERTIES table contains the CW\_NO\_SUB\_OF\_GLOBAL data value. When you set this value to 1, it indicates that the sub-process is not allowed for the global process.

### Runtime

In both asynchronous and synchronous sub-processes, the Process Engine starts a sub-process asynchronously by the sub-process (Spawn) activity. The sub-process is started as independent process, running in parallel with its main process in the same Process Engine (even if in clustered-PE environment).

For asynchronous sub-process, the sub-process activity completes immediately. The main process may use a [Wait Activity](#) to wait for completion of sub-processes of given type(s) or all sub-processes. It cannot, however, wait for completion of a specific sub-process instance.

For synchronous sub-process, the sub-process activity does not complete until the sub-process completes. The main process effectively waits for the sub-process' completion before it runs the next activity.

### Before and After Scripts

For synchronous sub-process activity, the *before* script can initialize the sub-process document that is accessible as *this.activityData* member. Upon completion of the sub-process, the activity runs the *after* script.

For asynchronous sub-process activity, the *after* script is not allowed.

### Impact of main process status change

The following is the impact to sub-processes when the main process' [status](#) is changed to:

- *Terminate* - all synchronous sub-processes as well as any other sub-processes are terminated.
- *Suspend* as a result of manual command or migration action - all synchronous sub-processes are also suspended.
- *Error* as a result of error in a parallel branch - all synchronous sub-processes are suspended.
- Suspended main process is *resumed* - all suspended synchronous sub-processes are resumed, too.
- Main process in error is *resumed* - all suspended synchronous sub-processes are resumed too.

Suspending (as result of exception) Spawn activity that waits for synchronous sub-process does not changes the status of the sub-process.

### Impact of synchronous sub-process status change

The following is the impact to the main process when synchronous sub-process' [status](#) is changed to:

- (Manually) *terminated* - main process throws the termination exception.
- *Suspended* as a result of manual command or migration action - the main process will wait until it is resumed and complete.
- *Error* - the main process will wait until it is resumed and complete.
- Suspended synchronous sub-process is *resumed*, and then completes - main process is notified.

## Exceptions

Asynchronous sub-processes handled their own exceptions, and if there is no suitable exception handlers, the sub-process goes in error.

For synchronous sub-processes, if an uncaught exception happens, the synchronous sub-process is terminated, and the corresponding exception type is stored in the column USER\_DATA in the completion participant message. If the main process wants to compensate synchronous sub-process' actions, it should define compensate activity for the main Spawn activity. The main process cannot trigger execution of compensate activities within the sub-process.

## Parameterization

Both asynchronous and synchronous sub-processes may have process document. The rules are:

- The synchronous sub-process can receive parameters only through its process document
- The synchronous sub-process can change the values of its own process. The process document is returned back to the main process through the corresponding participant message. This is the only way to send data from the sub-flow process to the main process.
- The synchronous sub-process can access the main process document data, but cannot change it. Keep in mind that the content of the main document may change at any moment during the synchronous sub-process process life time, if the main process has parallel branches.

## Sub-Flow

Sub-flows should be considered a presentation feature only -- presenting big process diagrams as a set of smaller diagrams. They should be used to increase the readability, not reusability.

## Runtime

The Process Engine expands activities of sub-flows in-line into the main process (as macros). Their activities become part of the main activities and they run synchronously within the process. (Since Release 5.0) Sub-flows are local components of the process diagram.

## Exceptions

Sub-flows are part of the main process. The main process exception handlers process any exception in a sub-flow.

## Best Practices for Batch Processing

---

Batch processing often involves processing sizable files that can lock system resources for a considerable period. To guarantee overall system performance is maintained while generating or processing batch files or both, follow these guidelines:

- Avoid generating or processing batch files within the foreground or order-related workflow. Leave the heavy lifting to background processes.
- Assign a low priority to the background processes.
- Schedule these background processes so that they perform their work only as required, rather than continually polling for work.
- Use asynchronous messaging to avoid locking the foreground processes that are generating or waiting for transactions, or both.
- Minimize the number of accesses to the batch file being created to reduce the probability of locking and exceptions (that is, do not allow one record to be written at a time by multiple processes or events).

### Generate a Batch Transaction

When generating a batch transaction file, follow these steps:

- Model a data structure to represent the message that is to be generated (a single line in the batch file).
- Model an interface to represent the external participant that will be processing the batch file (for example, switch activation).
  - The interface should contain an interface, operation, binding, service, and port.
  - The operation should have as the input message the data structure modelled previously.
  - Bind the interface to JavaScript.
  - Specify the script to be `return nameSpace.scriptName(input);` where the `nameSpace` and `scriptName` reflect those assigned in your metadata.
- Model a database table using a document to act as a message queue.
  - Define a primary key, such as a System Document ID.
  - Define a lengthy string field to carry the message, or use the methods to get or set BLOB fields if the length is indeterminate.
  - Define a field to carry the response.
  - Define a status field, for example, written, batched, processed, closed, or rejected.
  - Define fields to identify the initiator, such as the Process ID.
- Define the script referred to in the binding:
  - It should take the data structure (passed to it as the variable `input`) and insert a row in a message queue (that is, create the document, set the fields, and save).
  - If needed, it may return another data structure indicating success or failure.
- Define a global process to generate the batch file:
  - Define the process as a low priority process.
  - Schedule the process to iterate periodically (for example, every hour, or every day at 02:00).
  - The global process would, through a single script activity:
    - Create a new file.
    - Write a record for each row in the message queue where the status is `written`.
    - Close the file.
    - Update the status of each row processed to `batched`.

### Process a Batch Transaction

When processing a batch transaction file, follow these steps:

- Define a global process to process the batch file.
  - Define the process as a low priority process.
  - Schedule the process to iterate periodically (for example, every hour, or every day at 02:00).
  - The global process would, through a single script activity:
    - Get a directory listing of the files available and iterate through the files to be processed, if any.

- For each file, open it.
  - For each record in the file, perform the required action.
  - Close the file.
- Where the batch file processed is a response file to prior requests, the required action may be one of the following:
    - Update the appropriate row in the message queue with the response.
    - Set the status to **processed**.
    - If required or expected, send the response message to the initiator (identified by, for example, the Process ID) using a message such as *sendMessageToProcess*.
    - The initiator, upon receipt of the message, can set the message queue's status to **closed** (define an archiving process).
    - Where the batch file processed is a request for service (for example, a new order, status update, and so on), it is recommended that the actions being performed be decoupled from the processing the batch file. To accomplish this task, send a message to a participant whose interface is managed by a global process. See the next section.

## Define a Global Process-Based Message Queue

To allow parallel processing of batch files and to better control the resources allocated to batch file processing, define a system-managed message queue through global processes and an interface.

- Define an interface.
- Define a global process to manage the interface.
  - Define the global process.
  - Set a priority suitable to the operation.
  - Specify the defined interface as the interface in the General tab.
  - Add an operation activity in the process node where the operation is specified, but the participant is left blank.
  - Perform the required action (for example, create an update in the after script of the operation node).
  - Iterate the process with the required frequency.
- Define as many copies of this global process as needed to obtain required performance. For example, to process 5 in parallel, define 5 global processes.

# Participants

A participant of a process is an entity with which the process interacts. A process interacts with its participants by exchanging messages: sending to (producing) or receiving from (consuming) messages.

Three types of participants can be defined in the Process Management product:

- [User Participant](#)
- [External Service Participant](#)
- [Global Process Participant](#)

To allow a process to interact with its participant in a uniform way regardless of the participant type, the Process Management product requires every participant to expose the interfaces that define all the operations (requests or tasks) that this participant supports.

Participant interfaces are defined the same way as [external service interfaces](#). This way, process definitions always refer to participant operations (interface operation of the participant) and are isolated from the actual implementation of the message exchange.

## Create New Participant

To create a new Participant, do one of the following:

In the **Metadata** tab of **Navigation** pane, either:

1. Right-click a Namespace and select **New ...**
  2. **New Metadata Object** wizard appears as a popup. Expand **Business Processes**, and select **Participant**, and then click the **Next** button.
  3. Step two of wizard appears. Type in the name of Participant (must conform to JavaScript naming conventions).
  4. For **Task Type**, select *Manual* for User Participant, or *Automated* for External Service Participant or Global Process Participant.  
If **Manual** is selected, the Privileges list box appears. Select the privilege for the Participant.  
If **Automated** is selected, the Actor list box appears. Select the an existing Participant or None.
  5. Then, click the **Finish** button.
- OR
1. Right-click the **Participants** folder or the **Business Processes** folder in a Namespace, if it is present. Select **New Participant**
  2. Continue by following step 3 from above.

After the Participant is created by the wizard, you can no longer view or change the Participant Task Type, and different Participant properties are available based on the selected type. The newly created Participant is added under **Participants** folder.



## Notes:

- The Participant User Interface is of type `com.conceptwave.system.WLTaskUserInterface`.
- Each participant that is used to create worklist task must have a UserInterface in metadata (that is, right-click the participant and select **Create UI**). Otherwise, opening and displaying the task results in an exception.

## User Participant

---

The User Participant represents a person who performs manual tasks within a process. When a process flow reaches a manual task, the process sends a request to the user to perform this manual activity. The user notifies the process when he is finished the task. To allow interactions between process and user, the User Participant defines the interface such that the requests that a process sends to the user are *one-way operations*. The replies that the user sends to the process are *notification operations*.

For example, a User Participant *Approver* can be defined that has an interface with three Operations:

- *credit check*: one-way
- *approved*: notification
- *rejected*: notification

This interface definition allows the process to send only *creditcheck* requests to the User Participant, and allows the User Participant to reply with either an *approved* or *rejected* message.

Unlike message exchanges performed when invoking external service interfaces, interaction with a User Participant do not involve data exchange, but rather an indication of what operation to perform, through the use of an operation name. In the example above, when the process sends a *creditcheck* request, it does not send any data with the request. The process data (Process Order and Process Document) is available for any user task that this process requests.

Process Management implements user tasks through a worklist queue. The product User Interface (order management system) displays all current user tasks in a list, allowing the user to select tasks to work on. Every task in the list has an operation column that indicates the kind of task it is (such as *credit check* in the example above). When the user opens a particular task to work on, all *notification* type operations (for this task participant type, in our example *approved* and *rejected*) are automatically made available to the users through an action menu.

## User Participant Privilege

To allow certain manual tasks to be performed only by specific users, the Process Management product assigns user privileges to User Participants. User privileges are a user right, assigned to a user, specifying allowable actions in the system. User privileges are defined and assigned to the user in the [User Profile Management](#) application.

For example, we can define a *credit approval* privilege for the participant in the above example, allowing only users with this privilege to perform the *credit check* tasks.

Since users can have a number of user privileges assigned to them, users can perform any user participant task (operation) according to the privileges that the user possesses.

## Task Effort

Every manual task should be assigned with an effort metric in the **Effort** field on the **General** tab. This metric is used to calculate the current load for each user (the sum of all efforts for the currently assigned tasks). The system distribution type uses effort metrics to decide to whom to assign tasks.

## User Participant Tasks Distribution

When a process sends a task to a User Participant (invoking one of the *one-way* operations of the participant interface), these tasks are added to the global worklist queue, but not assigned to any specific user. The User Participant properties have a distribution type attribute which specifies how manual tasks should be assigned to users. This is defined by the **Distribution type** field on the **General** tab.

The Process Management product supports the following distribution types:

- **Manual**: The worklist manager assigns these tasks to each user manually. The worklist manager is a user with the [Worklist Administrator privilege](#).
- **Shared**: Users with the appropriate privilege can get new tasks by clicking the **Get Task** button on the user interface. The tasks with the highest priority will be selected first.
- **System**: The Process Management product automatically assigns tasks to users with the appropriate privilege. This method of distribution uses effort metrics to decide which user gets the next task. The system automatically assigns the next task to the user with the lesser load (sum of all efforts for the currently assigned tasks) that does not exceed the user's maximum effort defined in the [User Profile Manager](#). The user must log in at least once to start receiving tasks. When a user logs in or completes a task and this user's current load becomes less than a minimum effort defined in User Profile Manager, the system automatically assigns the next task to the user.

## User Participant Task Duration

The User Participant definition includes duration values for every task that the User Participant can perform (*one-way operations*), and duration values used for calculating task due dates. These values are defined in the **Duration** section of the **General** tab.

Due date calculation is based on the [calendar](#) specified in the User Profile Manager that is applied to the User Participant. Calendars specify working hours in 15 minutes increments; this allows weekends and holidays to be taken into account for due date calculations. For example, a task that is assigned on Friday afternoon with a duration of 8 hours would have a due date of Monday afternoon (assuming that Saturday and Sunday are a non-working weekend).

The User Interface shows tasks sorted by due dates (with the closest dates on top), and overdue tasks display in red.

## User Participant General Properties

The general User Participant properties are defined on the **General** tab.

The screenshot shows the 'General' tab of the User Participant configuration interface. The 'Name' field is set to 'commercial'. Under 'Task type', 'Manual' is selected. 'Distribution type' is set to 'Shared' with an effort of 1. The 'Privilege' dropdown also contains 'commercial'. A checkbox for 'Remove taken tasks' is unchecked. The 'Duration' section shows 1 day, 0 hours, and 0 minutes. The 'Script' section contains the following code:

```
function cwOnParticipantDuration(orderId, orderItemId) { // Calculates the duration when requested
    var process = this;
}

}
```

The following table describes the fields:

Field	Description
Name	Mandatory. Name of the User Participant within the namespace.
Private	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
Restricted	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
Deprecated	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object

	selection lists.
<b>Label</b>	Mandatory. Label to identify the User Participant.
<b>Description</b>	Description of the User Participant for documentation purposes.
<b>Task type</b>	Refers to the way in which tasks are delegated to participants. The selection is made in the setup Wizard (automated (external service participant) or manual (user processes)).
<b>Distribution type</b>	User participant task distribution type. See above for details.
<b>Effort</b>	Metric used to calculate current load for each user. See above for details.
<b>Privilege</b>	User participant privilege. See above for details.
<b>Remove taken tasks</b>	If checked, does not generate an alert when a task is taken from the user.
<b>Duration</b>	Sets default task duration by either defining the duration time in <b>Days</b> , <b>Hours</b> , and <b>Minutes</b> or by writing a JavaScript script in the <b>Script</b> field. The duration can only be expressed by one of the two ways. When a duration attribute specifies a script for duration value calculation, this script can return either the number of seconds (maximum amount of time in which the activity should complete) or a specific date (date by which activity should complete).

**Important:** If you configured the User Participant for Automated processes, the General tab appearance is different than if the User Participant was set for manual. In addition, the Interface, Orders and Methods tab does not appear for that User Participant configuration.

## User Participant Interface Tab

The **Interface** tab of the User Participant is used to define an Interface for interaction between a process and a user. For example, the tasks that can be assigned to the user and all the actions the user can perform on the assigned tasks.

Interface:  interfaceA.creditAssure ▼ 

**Operations and Conditions**

-  [Cancel](#)
-  [Complete](#)

<b>States:</b>	<input type="text"/>	...
<b>Tasks:</b>	<input type="text"/>	...
<b>Image:</b>		
<input checked="" type="checkbox"/> <b>Allow invalid order</b> <input type="checkbox"/> <b>No process reply</b> <input type="checkbox"/> <b>Keep task open</b> <input type="checkbox"/> <b>Bulk complete</b>		
<b>Condition:</b>	<pre>function cwOnPartOpCond(\$psCondition) { // Participant operation condition script     var permissionObject= this; // Deprecated - use "this" }  }</pre>	
<b>Confirmation:</b>	<pre>function cwOnPartOpConf(document, order, selectList, finder, menuOwner, menuObject, c     var worklistItem = document instanceof WorklistItem ? document : selectList;  }  }</pre>	
<b>Action:</b>	<pre>function cwOnParticipantOpAct(confirmObject) { // Called when the participant selects a gi     var worklistItem = this;  }  }</pre>	

The Interface can be selected from the **Interface** list box, which lists pre-existing Interfaces. Once the Interface is defined for this User Participant, the **notification** Operations (those with an **Operation type** of *Notification* in [Interface Operation](#)) will be listed in the **Operations and conditions** tree. For the selected Operation in the tree, the Operation condition properties are displayed on the right.

Field	Description
<b>States</b>	Assigns the Order states the Operation is allowed under. Shows the list of all Order states. Disabled if no Order states are specified in the metadata.
<b>Tasks</b>	List of tasks that this Operation is applicable to. When this list is empty, this Operation is available for every User Participant task. When the button at the end of the field is clicked, the <b>Select Tasks</b> dialog opens.
<b>Image</b>	Image to use in the Action menu for this Operation.
<b>Allow invalid order</b>	If checked, this Operation can be used regardless of the Order validity.
<b>No process reply</b>	If checked, a reply message will not be sent to the process on Operation completion.
<b>Keep task open</b>	If checked, the task will not be removed from the worklist on Operation completion.
<b>Bulk complete</b>	Allows this operation to be complete in bulk at runtime, allowing several tasks of this type to be selected and completed at once. Operations of this type can be multi-selected at runtime by the user and completed at once as a single action (when there is no data entry required). It is only allowed when all operations are of the same type.
<b>Condition</b>	<p>The Condition script, <code>cwOnPartOpCond(\$psCondition)</code>, is invoked when a worklist task is selected and determines whether this Participant operation is allowed. The script sets up a condition for the operation to be shown under the Action menu at runtime.</p> <p>The return values are either true or false. If the script returns true, the operation is allowed for the given task and is visible in the</p>

	<p>list of available task operations. If it returns false, the operation is removed from the list of available task operations.</p> <p><b>Example of use:</b> If you we need to prevent performing an operation on the task (for example, the user is not authorized or the task is not ready for this operation).</p>
<b>Confirmation</b>	<p>The Confirmation script, cwOnPartOpConf(document, order, selectList, finder, menuOwner, menuObject, confirmObject, uploadFile, orderList), is invoked when user selected an operation on the worklist task, right before the <b>Action</b> script is run. The Confirmation script may ask for user confirmation before the <b>Action</b> script is run.</p> <p>If the script returns true, the worklist <b>Action</b> script is processed; otherwise, the action is cancelled.</p> <p>The script takes the following parameters:</p> <ul style="list-style-type: none"> <li>• document - If a single worklist task is selected, the document corresponds to the selected worklist task (in other words, the worklist task on which the action is to be performed).</li> <li>• order - If a single worklist task is selected, it is the order associated with the current worklist task.</li> <li>• selectList - If multiple worklist tasks are selected, selectList is a list of all selected tasks.</li> <li>• finder - Not in use</li> <li>• menuOwner - Not in use</li> <li>• menuObject - Not in use</li> <li>• confirmObject - Not in use</li> <li>• uploadFile - Not in use</li> <li>• orderList - If multiple worklist tasks are selected, orderList is a list of all corresponding tasks orders.</li> </ul> <p><b>Return values:</b> true or false, or string or object to display in the confirmation message box.</p> <p><b>Example of use:</b> If there is a need to explicitly get user confirmation that the operation on this task is to be performed.</p>
<b>Action</b>	<p>The Action script, cwOnParticipantOpAct(confirmObject), is invoked when actions are to be performed on the current worklist task. The confirmObject parameter represent an object returned from the <b>Confirmation</b> script. The Action script does not have any return values.</p> <p><b>Example of use:</b> Change the order state, create another worklist task, or notify other users of the processed action.</p>

**Note:** You can set breakpoints in User Participant interface operation methods, such as Condition, Confirmation, and Action.

## User Participant Orders Tab

The **Orders** tab of the User Participant may define details of how the particular Order types are processed. It is not mandatory to add any Orders to this tab.

General Interface Orders Methods

Order Type	Default Order Item	Duration	
customerOrder	customerOrder		
orderATM	orderATM		

**Add** **Properties** **Remove**

The **Orders** table contains a list of all the the Orders that have had properties defined for the User Participant.

When the **Add** button is clicked the **Order Properties** dialog opens where the User Participant Order properties can be defined. To view or edit the Order properties, select the desired row in the **Orders** table and click the **Properties** button to open the **Order Properties** dialog.

When the **Remove** button is clicked, the selected element is removed from the **Orders** table.

**Order Properties**

Operation:	<code>orderManagement.customerServiceInterface.handleRejectedCredit (Handle Rejected Credit)</code>	Effort:	<input type="text" value="1"/>
Order type:	<code>com.conceptwave.system.Order</code>		
Default order item:	<code>com.conceptwave.system.Order</code>		
Duration:	Days: <input type="text"/>	Hours: <input type="text"/>	Minutes: <input type="text"/>
Script:	<pre>function cwOnDuration() { /* Calculates the duration when requested */     var process = this; }</pre>		

**Save**

The following table describes the fields

Field	Description
<b>Operation</b>	A list box containing all <i>one-way</i> Operations (refer to section on <a href="#">External Services</a> for an <b>Operation type</b> of <i>One-Way</i> on the <b>General</b> tab of the Operation properties) defined for the <b>Interface</b> selected on the <b>Interface</b> tab for this User Participant.
<b>Effort</b>	Reserved for future use.
<b>Order type</b>	A list box containing all defined Orders. When the <b>Order type</b> is selected, either the <b>Default order item</b> or <b>Duration</b> must be defined.
<b>Default order item</b>	A list box containing all Documents of the underlying Order. If an Order Document is selected in this field, it is opened automatically when the User Participant works on the Order worklist task for the selected Operation.
<b>Duration</b>	Defines a time constraint for the User Participant to work on the underlying Order. The duration value defined here overwrites the default <b>Duration</b> value provided on the <b>General</b> tab. The duration can only be expressed by one of two ways: by expressing the duration time in <b>Days</b> , <b>Hours</b> , and <b>Minutes</b> or by writing a JavaScript script in the <b>Script</b> field.

## User Participant Methods Tab

The **Methods** tab of the User Participant may define details of what methods may be performed on this object.

The `cwOnPartDistrib` script is a custom distribution rule for worklist tasks created for the participant type. The script is invoked when the worklist task for the participant is created and returns the string value representation of the user. If the returned string is an existing user, the worklist task is assigned to it. Otherwise, an error message appears.

General Interface Orders Methods

 **userP**

 cwOnPartDistrib

**Name:** cwOnPartDistrib

**Description:**

**Script:**

```
function cwOnPartDistrib(document, itemData, process) { // Distributes the load between particip }
```

In addition, you can add a New Method defining parameters for the script and a return type.

General Interface Orders Methods

**commercial**

- cwOnPartDistrib
- NewMethod

Name: NewMethod  Is private

Description:

Parameters:

Name	Type

Script:

```
function NewMethod() { // Metadata type method. Can be called by scripts.  
}  
}
```

Return:  com.conceptwave.system.Void

## External Service Participant

The External Service Participant represents the external services that the process needs to interact with. The External Service Participant defines the metadata Interface element which defines all the Operations that can be invoked on the service.

The framework uses an abstract definition ([Interface](#) and [Operation](#)) when it needs to specify an external service invocation. The Process Manager will automatically select the proper [port](#) where the actual [service](#) is located.

All invocations of External Service Participant Operations are delegated to the [interface process](#) that services this Interface. If any errors occur when invoking the external service operation by the interface process, the interface process will send an *External interface error exception* to the process that requested this Operation.

The External Service Participant can be configured to [retry an Interface Operation](#) until it succeeds, instead of returning an exception to the requesting process. This can be useful, for example, when connecting to an external service over a public network (such as the Internet), where occasional network glitches are expected.

### External Service Participant General Properties

The External Service Participant is [created with Participant's Task type as Automated](#), and then select an *Interface* in the **Actor** property in the **General** tab as shown below.

The screenshot shows the 'General' tab of the External Service Participant configuration dialog. It includes fields for Name (externalServiceP), Label (externalServiceP), Description (empty), Actor (interfaceA.creditAssure), and checkboxes for Private, Restricted, and Deprecated. Below the Actor field are two checkboxes: 'Invoke directly' (unchecked) and 'Return error to process' (unchecked). There is also a small icon for adding a new actor.

The **Return error to process** flag should be selected if the process expects to receive an error message when the Operation fails. This flag works in conjunction with an Operation of the type "Request-Response". Operations of the type Request-Response will wait for a successful response or an error message.

When the Participant has the **Return error to process** checkbox selected, the Interface process sends a document about the error to the calling user process. In CW templates, the **message** (String) field is available in `document.(“cwf_pm:processExcpDoc”)`. This document is delivered to appropriate exception handler in the user process.

In the **Before** script, extended information about error may be received by using the following:

```
var errorString = document.message;
```

The **Invoke directly** flag should be selected to avoid invocation of the Interface through the [interface process](#) in Process Engine. The default value is false. When this flag is checked, all 'Operation' activities sending message to the current participant will invoke the Interface directly, regardless of the status of the **Invoke Directly** flag in those [activities](#). (Note: In the process activity **Operation**, when you choose an External Service Participant as the participant, then option **Invoke Directly** is also available.)

The interface process is a system process and is not exposed in the metadata. This process is in charge of converting data message objects from their string format to XML objects, so the actual message being sent is a Java XML object and not a simple string. When you invoke directly, the external service receives the message content as a string (with XML tags), and not as a .xml file/object.

## Global Process Participant

The Global Process Participant represents global processes that provide services to other processes. The Global Process Participant defines the global process that the participant represents. The Interface that the Global Process Participant exposes is defined in the global process definition.

### Global Process Participant General Properties

The External Service Participant is created with Participant's Task type as *Automated*, and then select an *Process* in the **Actor** property in the **General** tab as shown below.

The screenshot shows the 'General' tab of the 'Global Process Participant' properties dialog. The 'Name' field contains 'globalProcessParticipant'. The 'Label' field also contains 'globalProcessParticipant'. The 'Description' field is empty. The 'Actor' field contains 'myNS.orderH' with a small icon to its left. To the right of the 'Actor' field are three checkboxes: 'Private', 'Restricted', and 'Deprecated', all of which are unchecked. There is also a small icon with a pencil and a globe in the top right corner of the dialog.

## Alerts/Jeopardies

An Alert defines a destination for a notification mechanism; four mechanisms are supported:

- *Email*
- *Pager*
- *External System*
- *Worklist*

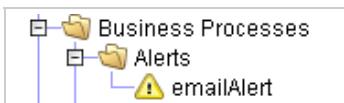
This destination is defined by an Alert metadata element. Jeopardies are implemented through Alerts.

### Create New Alert

To create a new Alert, do the following steps:

- In the **Metadata** tab of **Navigation** pane, either:
  1. Right-click a Namespace and select **New ...**
  2. **New Metadata Object** wizard appears as a popup. Expand **Business Processes**, select **Alert**, and then click the **Next** button.
  3. Step two of wizard appears. Enter the name of Alert (must conform to JavaScript naming conventions), and populate other fields, as necessary. Then, click the **Finish** button.
- OR
  1. Right-click the **Alerts** folder or the **Business Processes** folder in a Namespace, if it is present. Select **New Alert**.
  2. Follow step 3 from the previous procedure.

The newly created Alert appears under the **Alerts** folder.



### Alert Properties

The fields of the Alert properties may change depending on the **Type** field selection, whether it is *email* or *pager*, *external system* or *worklist*. The table below describes the fields.

*Alert properties for the Email or Pager type.*

<b>Name:</b>	emailAlert	<input type="checkbox"/> Private	<input type="checkbox"/> Restricted	<input type="checkbox"/> Deprecated
<b>Label:</b>	emailAlert			
<b>Description:</b>	<input checked="" type="button"/>			
<b>Type:</b>	email	<input type="button"/>		
<b>Address:</b>	admin@email.com			

*Alert properties for the External System type.*

--	--

Name:	alert1	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated
Label:	alert1	
Description:	<input type="button" value="Edit"/>	
Type:	external system	<input type="button" value="..."/>
Element:	interfaceA.creditAssure	<input type="button" value="..."/>
Operation:	interfaceA.creditAssure.creditCheck	
Script:	<pre>function cwOnAlert(document) {</pre>	
	<pre>}</pre>	

Alert properties for the Worklist type.

Name:	alert2	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated
Label:	alert2	
Description:	<input type="button" value="Edit"/>	
Type:	worklist	<input type="button" value="..."/>
Element:	interfaceA.userP	<input type="button" value="..."/>

Field	Description	Type
Name	Mandatory. Name of the Alert within the namespace.	All
Private	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).	All
Restricted	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).	All
Deprecated	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.	All
Label	Mandatory. Label to identify the Alert.	All
Description	Description of the Alert for documentation purposes.	All
Type	Mandatory. The destination type; choose one of the following: <ul style="list-style-type: none"> <li>• <i>Email</i>: This alert type will dispatch an email message to a defined email address. The <a href="#">Configuration application: Processes - Alerts</a> tab contains an SMTP server configuration parameter that is used to define the SMTP server address. The Email address specified in the alert definition can be overridden in that configuration page as well. For an <a href="#">alert activity</a>, the message content is derived from the <b>Before</b> script return value.</li> <li>• <i>Pager</i>: As per email, where the email address specified is the pager address.</li> <li>• <i>External System</i>: This alert type will invoke an external service Operation. The script defined in the Alert will be invoked with Document and process parameters, where the Document parameter type is defined in the input of the Interface Operation and a process parameter is a process that issues the Alert.</li> <li>• <i>Worklist</i>: This alert type will send a request (task) to the worklist queue. The Alert specifies the User Participant type that should receive this request (alert). The label of the Alert is used for the worklist task description. For an <a href="#">alert</a></li> </ul>	All

	<a href="#">activity</a> , the worklist task description can be set by an activity <b>Before</b> script return value.	
<b>Address</b>	Mandatory. The e-mail address.	Email
<b>Address</b>	Mandatory. The pager number.	Pager
<b>Element</b>	Mandatory. The external service Interface.	External System
<b>Operation</b>	Mandatory. The external service Interface Operation that initializes the output Document.	External System
<b>Script</b>	A JavaScript script that initializes the output Document.	External System
<b>Element</b>	Mandatory. The User Participant.	Worklist

# Dispatching Alerts

---

## Alert Activity

As part of the process flow, an alert activity can issue an Alert to indicate and/or escalate a business error.

## Process Abnormal Termination

This Alert is dispatched when a process instance terminates prior to completing all process activities. The causes for process abnormal termination include such conditions as JavaScript errors, failure to handle process exceptions and database errors.

When issuing email/pager Alerts for a process failure, the email message content is a fixed system message (WE0013) that contains the process ID and process type.

Process abnormal termination Alerts are configured in the Configuration application: [Processes - Alerts Assignments](#).

## Process Engine Abnormal Termination

This Alert is dispatched when the Process Management product detects a Process Engine failure. Only User Participant Alerts are supported for Process Engine failure notifications.

Process Engine abnormal termination can be caused by server shutdown or by Java VM (virtual machine) shutdown. Process Engine abnormal termination is detected by a database stored procedure that runs periodically as a database job. Every Process Engine updates the database approximately every 5 minutes. This database job checks that every Process Engine has updated its information in the last 15 minutes. Any Process Engine that has stopped running will not update its timestamp, and will be considered dead within 15 minutes.

The Process Engine abnormal termination Alerts are also configured in the Configuration application: [Processes - Alerts Assignments](#).

## Process Exceptions

The Process Management product defines the handling of unexpected events (jeopardy situations) in business processes. An exception event can be a timeout or an [exception activity](#). This handling is defined through a special [On Exception activity](#) that is invoked by the Process Management framework, in response to an unexpected event.

An *On Exception* activity can handle either one specific exception type or any exception type (exception type omitted in the *On Exception* activity properties). An *On Exception* activity specifies activities that should be executed when this exception type (specified in the *On Exception* activity properties) occurs.

An *On Exception* activity can be only assigned to a *complex* activity. Multiple *On Exception* activities within the same *complex* activity may be used to handle different exception codes. A set of *On Exception* activities must use unique exception codes, and at most one activity may omit the exception code attribute.

An *On Exception* activity also specifies how to handle execution of the activities once the exception has occurred. The activities' execution can be either proceed normally (in this case, the *On Exception* handler can notify or escalate the problem) or terminate execution of all activities of the *complex* activity to which the *On Exception* activity belongs (in this case, the *On Exception* handler can perform an alternate flow, or even terminate the process.)

### Create New Exception

To create a new Exception, follow these steps:

- In the **Metadata** tab of **Navigation** pane, either:
  1. Right-click a Namespace and select **New ...**
  2. **New Metadata Object** wizard appears as a popup. Expand **Business Processes**, select **Exception**, and then click the **Next** button.
  3. Step two of the wizard appears. Enter the name of Exception (must conform to JavaScript naming conventions), and populate other fields, as necessary. Then, click the **Finish** button.
- OR
  1. Right-click the **Exceptions** folder or the **Business Processes** folder in a Namespace, if it is present. Select **New Exception**.
  2. Follow step 3 from the previous procedure.

The newly created Exception appears under the **Exceptions** folder.



### Exception Properties

The Exception element's properties appear on the General tab.

General		Methods
Name:	CreditException	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated
Label:	Credit Exception	...
Description:	<input style="width: 150px; height: 20px;" type="button" value="..."/>	
Default Handler:	<NONE>	<input style="width: 20px; height: 20px;" type="button" value="..."/>

The General tab contains the following fields:

Field	Description
Name	Mandatory. Name of the Exception within the namespace.
Private	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is,

	library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Label</b>	Mandatory. Label to identify the Exception.
<b>Description</b>	Description of the Exception for documentation purposes.
<b>Default Handler</b>	<p>Use this field to define the default exception handler. When an exception happens and the process engine cannot find an appropriate exception handler, it runs the default handler as it is defined for the root or first activity of the process. See Default External Exception Handler</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• This handler does not appear in the process flow diagram.</li> <li>• If no exception handler is found, the process goes into error state.</li> </ul>

## Default External Exception Handlers

When an exception occurs and the process does not have an exception handler for the given exception type, it processes a default external exception handler, if specified. Default external exception handlers are external exception handler processes that are specified in the corresponding exception metadata elements.

The Exception metadata element can specify the name of exception handler process either statically or dynamically. For a statically defined handler, select the handling process from the **Default Handler** field's drop-down menu. Otherwise, select **<Dynamic Exception Handler>** from the **Default Handler** field and return the handler's name in the cwOnExceptionInit() script. This script is processed at runtime when the exception is generated.

For the process to be a legitimate handler, it must use a document that has two fields:

- cwf\_pm:cwResumeIndicator
- cwf\_pm:cwRollbackCode

These fields are used to pass resume and rollback details, respectively, to the process containing the exception node. The process document can extend the template's document cwf\_pm:ExternalExceptionHandlerDoc containing these two fields. Only legitimate handler processes are available in the Exception's **Default Handler** field's selection list.

The AVM processes the exception as follows:

1. It creates an empty process document for the default handler and then maps the parent process document to it using a default conversion map, if any.
2. It processes the external exception handler sub-process synchronously. If the process has an unprocessed exception, it is returned to the parent process.

**Note:** Default exception handlers are processed in the external exception handler sub-process.

3. Maps the external exception handler sub-process document to the parent process document using a default conversion map, if any.
4. Processes the rollback and resume actions as specified by the external exception handler process document. If an exception happens in the default exception handler, it may process another default exception handler. If no exception handler is found, the exception is returned to the parent process.

## Exception Types

---

The following exceptions are defined and used by the Process Management product (find them in **Library** tab of **Navigation** pane, under `cwf_pm` namespace):

### Timeout Exception

System-defined exception type. The Process Management framework automatically issues a *timeout* exception for activities that fail to complete in a specified time provided they have a [duration attribute](#) defined.

### External Interface Error

System-defined exception type. The Process Management framework automatically issues an *external interface* error exception for any error that occurs when an *operation* activity invokes an [external service participant](#). The interface process will log an actual error, and the Process Management framework will send an *external interface* error exception to the *operation* activity that sent that particular request. The AVM can be configured not to return the *external interface* error to a calling process, but instead retry indefinitely, by unchecking the **Return error to process** flag of the External Services Participant properties. The operation activity that sent the request must be of the type *Request-Response* so that it will receive the error and the external service participant must have the *Return error to process* flag set to receive the error. Once the error is received by the External Service Participant, then the error can be sent back to the awaiting operation of the type *Request-Response*.

### External Service Fault

System-defined exception type. The Process Management framework automatically issues an *external service fault* exception when an external service that was invoked by an *operation* activity returns a *fault message*. The fault message returned by an external service will be passed as a Document to the **Before** script of the *On Exception* activity.

### Stop Exception

System-defined exception type. Can be used to send a stop request to a Global process. To ensure that Global processes stop properly (since only the process itself knows how to stop properly), the Process Management framework provides only a facility to deliver a stop request to a process but does not automatically stop it. The *stop* exception is always delivered to the process' top activity. A properly designed Global process should provide an *On Exception* handler for this exception and either complete the process immediately or mark it with a *stop request* and check for this condition in the process flow.

### Cancel Exception

System-defined exception type. Can be used to send a cancel request to a User process. Since only the process itself knows how to cancel properly, the Process Management framework provides only the facility to deliver a cancel request to a process but does not automatically cancel it. The *cancel* exception is always delivered to the process' top activity. The process should provide an *OnException* handler for this exception and perform whatever is appropriate for the current process state (for example, ignore completely, perform rollback, notify children or exit immediately).

### Change Exception

System-defined exception type. Can be used to send a change request to a User process. Since only the process itself knows how to perform a process change properly, the Process Management framework provides only the facility to deliver a change request to a process. The *change* exception is always delivered to the process' top activity. The process should provide an *OnException* handler for this exception and perform whatever is appropriate for the current process state (for example, ignore completely, perform rollback, notify children or exit immediately).

### User Exception

Exception type defined in the application metadata. This exception can be triggered by an [exception activity](#). This exception usually indicates a form of business error (for example, an external system failed to locate a customer record) and requires some form of corrective action such as creating a new customer record, changing the flow of the process or even terminating the process.



## Exception Handling Processing

---

The following logic is employed by the Process Management framework in handling exception processing:

- The system locates the *On Exception* activity that handles the exception type (code).
- The Process Management framework checks the parent-child chain for the *On Exception* activity that can handle the required exception type, starting with the current activity (the activity that caused the exception), and going up to the parent, and the parent's parent, etc.
- The *On Exception* activity can handle the exception if it specifies this exception type, or if it omits the exception type (any exception) in the activity definition.
- If no *On Exception* activity that can handle the exception is found, the process is terminated with an *error* status.
- If an *On Exception* activity is found but is currently executing another exception, the current exception will be processed after the *On Exception* activity completes the previous exception handling.
- If an *On Exception* activity is found, all activities of a *complex* activity (*On Exception* parent) are suspended and the *On Exception* activity is queued for execution.
- The *On Exception* activity flow can include a *Resume* activity, to resume previously suspended activities of a *complex* activity. The *On Exception* sub-activities are executed in parallel to *Resume* activities. This is the only way to continue execution of suspended activities.
- The *On Exception* activity flow can include a *Rollback* activity, to invoke *Compensate* activities for all completed activities of a *complex* activity. The *Rollback* activity will wait till all *Compensate* activities complete.
- The *On Exception* activity cannot throw an exception; if this happens, the process will be terminated with an error status.
- An unhandled exception is delivered to the parent process only if the child process started from the spawn activity as a synchronous process.

## On Exception Activity Execution

---

If the *On Exception* activity has a **Before** script defined, this script receives the following information:

- The *currentActivity* property of a process (refer to the JavaScript Extensions HTML documentation) points to an activity that caused the exception. This property is applicable only for the exceptions thrown by the current process and not for the exceptions sent from parent or child processes.
- For an *external service fault* exception, the Document parameter contains the fault message returned by the external service.
- For an exception sent from a parent or child process, the Document parameter contains the ID of the process that sent this exception (use *document.senderID*).

## Rollback Activity Execution

---

The following logic is employed by the Process Management product in handling a *Rollback* activity:

- Execute the **Before** script. This script should return one or more compensation codes (single code or array of codes) or null.
- If the script returns compensation codes, a *Rollback* activity will invoke only the *Compensation* activities that are marked with one of the compensation code(s) returned by this script.
- A *Compensation* activity that has no compensation code set will always be invoked. To enable the **Compensate** field on the general type, you need to define a data type that extends `isEnabledWhenInMD` Data Type that extends or overrides the `cwf.compensate` enumeration type.  
So user will be able to select from them.
- If the **Before** script returns null, all of the *Compensation* activities are invoked.
- All manual activities that are rolled back and not yet assigned to any user are deleted.
- All manual activities that are rolled back and already assigned to user are converted to `alert`.
- All operation activities that are rolled back and have sent a request to the external system, which has not yet been processed, are deleted from the interface process queue.
- All message responses from external systems that are not yet processed (for operation activities that are rolled back) are deleted from the process queue.
- When there are several non-parallel activities of a suspended tree, they can be in the activity queue simultaneously. The *Rollback* points to a restart target. At runtime, when an exception occurs, the process engine suspends activity above this reset target. *Resume* resumes the suspended activity and also restarts the restart target.

## Exception Between Processes

---

The Process Management framework allows sending exceptions between parent and child processes. Sending exceptions to another processes can be used not only to propagate error conditions, but also as a notification or synchronization mechanism (a child can send an exception to a parent to indicate it completed a certain step).

To send an exception to another process, the **Before** script of an *Exception* activity should return a process ID or process object, indicating to whom an exception should be sent (refer to the [JavaScript Extensions HTML documentation](#)). For example, to send an exception to a parent process, use the following script:

```
return process.parent
```

## Signals

The Process Management product provides the ability to define Signals to coordinate the execution of activities executing in the same process or of activities executing in different processes. Signals are often used to synchronize between parallel activities (executing in the same or different process).

The *signal* activity raises the Signal by producing a new instance of the Signal. The *sync* activity synchronizes the Signal by waiting for the Signal to be raised, and then lowers the Signal by consuming it.

Signals are uniquely identified by the Signal's **Name** and defined as a global metadata element.

Signals can be raised locally (within the same process) or sent to another process. To send a Signal to another process, the **Before** script of the *signal* activity should return a process ID or process object, indicating to whom a Signal should be sent (refer to the [JavaScript Extensions HTML documentation](#)). For example, to send a Signal to a parent process, use following script:

```
return process.parent
```

### Create New Signal

To create a new Signal:

- In the **Metadata** tab of **Navigation** pane, either:
  1. Right-click a Namespace. Select **New ...**
  2. **New Metadata Object** wizard appears as pop-up. Expand **Business Processes**, and select **Signal**, and then click **Next**.
  3. Step two of wizard appears. Type in the name of Signal (must conform to JavaScript naming conventions), and populate other fields as desired. Then click **Finish**.
- OR
  1. Right-click the **Signals** folder or the **Business Processes** folder in a Namespace, if it is present. Select **New Signal**
  2. Follow step3 from above.

The newly created Signal is added under **Signals** folder.



### Signal Properties

The Signal properties are defined as follows.

<b>Name:</b>	readySignal	<input type="checkbox"/> Private	<input type="checkbox"/> Restricted	<input type="checkbox"/> Deprecated
<b>Label:</b>	readySignal			
<b>Description:</b>	<input type="button" value="Edit"/>			

Field	Description
<b>Name</b>	Mandatory. Name of the Signal within the namespace.
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.

<b>Label</b>	Mandatory. Label to identify the Signal.
<b>Description</b>	Description of the Signal for documentation purposes.

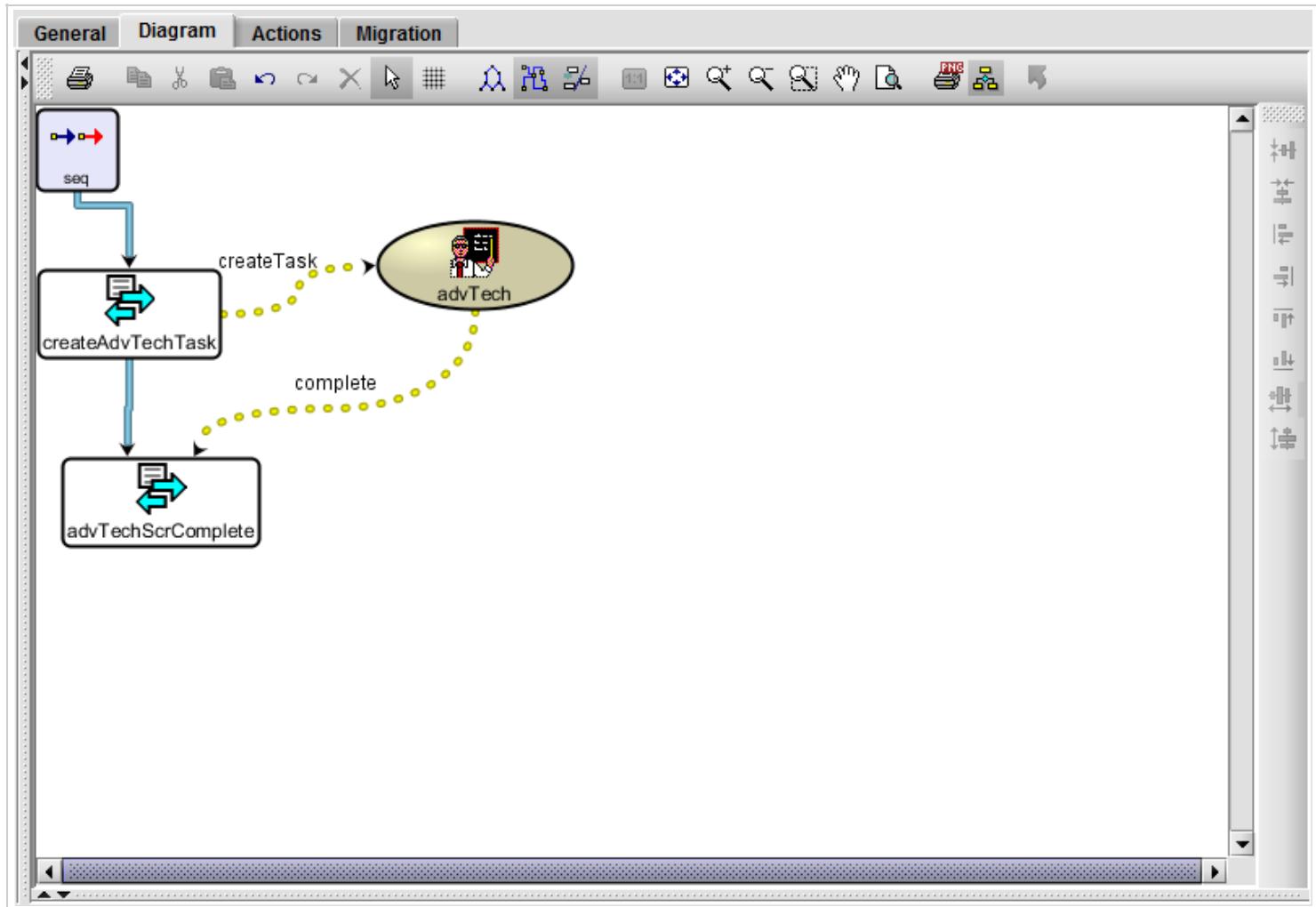
## Process Activities

A Process is a progressively continuing flow of executable activities, directed toward a particular goal. An Activity is a component that performs a specific function within a process. Activities can be as simple as sending or receiving a message, or as complex as coordinating the execution of other processes and activities.

Activities are either *atomic* or *complex*:

- An *atomic* activity is an elementary unit of work that cannot be further decomposed. *Atomic* activities can be used to start other processes, perform calculations, or perform operations.
- A *complex* activity is composed of other activities, and directs the execution of these activities.

All activities of a specific process instance are executed by the same Process Engine.



# Process Activities Types

---

The following process activity types are available.

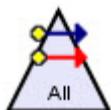
## Alert



An *atomic* activity that executes a specified Alert.

- Not defined in the BPML specification.
- For *email* and *pager* Alerts, **Before** script is used for message content.
- For a *worklist* Alert, **Before** script is used for alert description.
- For an *external service* Alert, **Before** script is used for output message initialization.
- An **After** script is not allowed.

## All



A *complex* activity that finishes after all its sub-activities have completed.

- The *All* activity cannot include more than 16 sub-activities.
- These sub-activities can be run in parallel (concurrently) or in series (one after another).
- In addition, sub-activities are run in no particular order, meaning a sub-activity that is defined first may not be run first.

## Break



Activity that is a child of the [\*While\*](#) activity.

- When the condition script returns true, the closest *While* loop (that is, the parent of this *Break*) is completed.
- The *Break* activity is optional in the *While* activity's loop.
- This activity does not have **Before** and **After** scripts.

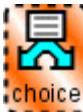
## Case



Associates a condition with a *Switch* activity.

- Can contain zero or one sub-activity.
- No **Before** script or **After** script allowed.

## Choice



A *complex* activity that completes after one of its sub-activities have been completed. All sub-activities are defined in parallel

but only one activity is processed. The [\*Choice Setup\*](#) section gives an example of a Choice setup process diagram.

The *Choice* activity allows a process to wait for one of the sub-activities to arrive such as an Operation activity with a notification messages (for example, the credit approver participant can reply to the process with either an *approved* or *rejected* message).

- Choice sub-activities include: *Operation*, *Synchronize*, *Timeout*, *On Exception* and *Compensate*.
- Two (2) or more sub-activities must be specified of the types: *Operation*, *Synchronize* and *Timeout*.
- The *Operation* sub-activity must be defined with its **Operation** parameter set to a notification operation.
- **Before** and **After** script are allowed.

## Compensate



Associates a *Compensate* activity with a *atomic* or *complex* activity. The *Compensate activity* is used to revert any

changes resulting from the completion of the activity in the event of *Rollback* activity.

- Can contain one and only one sub-activity.
- The *Compensate* activity will be executed if the activity has completed, but the *Rollback* activity is executed by the *On Exception* activity of the ancestor.
- *Compensate* activities will be executed in the reverse order in which they were defined.
- If a *Compensate* activity is defined for a *complex* activity, it overrides any *Compensate* activities defined for any of its sub-activities, once the *complex* activity has completed.
- An **After** script is not allowed.

## Complete



An *atomic* activity that completes the process.

- Can appear as the last activity in a sequence, or once in a *Choice* or *Switch* activity.
- The *complete* activity causes the process to complete immediately. It does not cause children or parent processes to complete.
- Activities that are waiting for their children (in the current process) to complete do not run the **After** script, if it exists, and do not have a completion time in the database.
- An **After** script is not allowed.

## Exception (Fault)



An *atomic* activity causes an exception to occur in the process or to be sent to another process.

- To send an exception to another process, the **Before** script should return a process ID or process object indicating to whom the exception should be sent (refer to the [JavaScript Extensions HTML documentation](#)).
- The exception occurs in the parent *complex* activity.
- The exception code name provides a global identifier for the exception type.
- An **After** script is not allowed.

## On Exception



Associates an *Exception* activity with a *complex* activity. It is used for recovering from exceptions, and allows the

process to continue.

- Can contain one and only one sub-activity.
- The *Exception* activity will be executed if the activity ends with an exception.
- If the *Exception* activity completes successfully, then the failed activity also completes successfully.
- Multiple *On Exception* activities may be used to handle different exception codes. A set of *On Exception activities* must use unique exception codes, and at most one element may omit the exception code attribute.
- An **After** script is not allowed.
- **Note:** On Exception activities for process and decision trees cannot use subflow processes.

## Operation



An *atomic* activity that performs a message exchange with a process participant.

- The participant is mandatory for non-global processes.
- A global process can specify the participant (using services of another participant), or omit the participant and use its own Interface Operation (providing services to other participants).
- This activity will either consume the message (*notification*), produce a message (*one-way*), or both, depending on the Interface Operation selected.
- When a global process specifies its own Interface Operation, it will be interpreted in opposition to the operation type (that is, a *one-way* Operation will perform consume).
- Can contain zero or one sub-activity.

## Poll



An *atomic* activity that allows you to define the process flow dynamically and recalculate it later.

- The **Schedule** option in any activity allows you to delay an activity only one time, and this activity eventually runs and completes. Sometimes, business-process logic needs to recalculate and delay the process flow to wait for some changing condition, which can be done using the **Poll** activity.
- The process engine expects to receive a positive number from the *Before* script of this activity that represents the activity's delay time in seconds. After this time passes, the process engine calls the **Before** script again to receive a new time for new delay.
- When the script returns 0 (zero), the process engine completes this activity and performs the next activity in the flow.
- Negative numbers are not allowed. Additionally, any other type of data that not an integer number is not allowed.
- This activity has both **Before** and **After** scripts.

### Poll scenarios:

- If the **Before** script is empty (that is, it returns nothing), the process goes into an error state (DE0332: Invalid poll interval).
- If the **Before** script is not empty, but does not return anything, the process goes into an error state (DE0332: Invalid poll interval).
- If the **Before** script returns null, the process goes into the error state (DE0332: Invalid poll interval)
- If the **Poll** activity defines a delay time in **Schedule** property, the first delay activity time is the time defined in the **Schedule** property. For the process engine, it is the delay before this activity starts. After this time, the process engine starts the activity, runs its **Before** script, and then delays the flow as a result of this script.
- If the **Poll** activity defines the duration time in the **Schedule** property, the **Schedule** property (that is, the waiting timeout period starts only when the scheduled delay has passed) is first evaluated. The waiting timeout period starts when the **Before** script runs for first time because this delay (from the **Before** script) occurs after starting this activity.

## Repeat



An *atomic* activity that repeats a *complex* parent activity. The **Repeat** activity is used to model *while* loops.

- Can appear as the last activity in a sequence, or once in a *Choice* or *Switch* activity.
- The parent activity is not limited to the immediate parent (it can go back to grandparent or above).
- An **After** script is not allowed.

## Resume

An *atomic* activity that resumes previously suspended activities by the *On Exception* activity.



Resume

- Can only be used as part of the *On Exception* activity flow.
- A *Resume* activity is executed in parallel to the *On Exception* activity flow.
- An **After** script is not allowed.

## Rollback



An *atomic* activity that invokes a *Compensate* activity for all completed activities of a *complex* activity that is a parent of an

*On Exception* activity

- Can only be used as part of the *On Exception* activity flow.
- The *Rollback* activity will wait until all *Compensate* activities are complete.

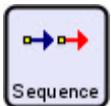
## Script



An *atomic* activity that executes a **Before** script.

- An **After** script is not allowed.

## Sequence



A *complex* activity that completes after all its sub-activities have completed. All sub-activities are executed in sequence.

## Signal



An *atomic* activity causes a Signal to be raised in the process or to be sent to another process.

- To send a Signal to another process, the **Before** script should return a process ID or process object indicating to whom the Signal should be sent (refer to the [JavaScript Extensions HTML documentation](#)).
- The Signal name provides a global identifier for the Signal type.
- An **After** script is not allowed.

## Spawn (Sub-process)



An *atomic* activity that spawns a child process.

- A sub-process is started asynchronously by the Spawn activity -- as independent process, running in parallel with its parent.
- If **Synchronization** flag is unchecked, the activity completes immediately, which means it may complete before its children complete. If **Synchronization** flag is checked, the activity completes only when the sub-process completes.
- The parent process can wait for child process completion by using a *Join* activity.
- An **After** script is not allowed if **Synchronization** flag is unchecked.
- Contrast this activity with *sub-flow* activity, where sub-flow is expanded in-line into parent process. See [sub-processes and sub-flows](#) for details.
- **Notes:**
  - A child process cannot write into process document of parent process; it is read-only for child processes.

- The process engine creates a new document instance for a child process and copies all fields, except for system objects, such as cwf.cwDocId. It is a document copy of the parent when creating the sub-process. If a field is changed in the parent process document at a later period, this change is not visible in the sub-process document, even with the **Use parent process document** field selected for the sub-process document.
- Subprocess activities for process and decision trees cannot use subflow processes.

## Sub-flow



A *complex* activity that calls a subflow (akin to a procedure call). The **Sub-flow** activity is used for breaking a complex process into separate components, but in contrast to a [Spawn activity](#), the activities in **Sub-flow** become part of the parent activities and they run synchronously within the process. See [sub-processes and sub-flows](#) for details.

- Not defined in the BPML specification.
- Current process priority used for subflow activities.
- When executing scripts of included process activities, the process parameter passed to those scripts is the process where the **Sub-flow** activity is defined (refer to the [JavaScript Extensions HTML documentation](#)).
- Cannot include *On Exception* activities.
- **Note:** The context menu of the **Sub-flow** activity includes the **Edit** option, which opens up the top-level subflow property panel where subflow is in editable mode.

## Switch



A *complex* activity which completes after zero or more of its sub-activities have completed, subject to one or more conditions. This models process branching.

- Can only have *Case* sub-activities.
- Can only have one default *Case* activity (a case with no condition).
- Can execute more than one *Case* activity (not exclusive). When *Switch* activities are marked as none-exclusive, all *Case* activities that have a condition evaluated to true execute the same way.
- If no *Case* activity has a condition that evaluates to true, the default case is executed.
- Cannot have more than 16 sub-activities (optimization restriction).

## Sync



An *atomic* activity that synchronizes on a Signal. It waits for a specified Signal and completes when such a Signal arrives. Used for synchronization of parallel execution branches within a process or for synchronization of parent/child processes.

- A **Before** script is not allowed.

## Timeout



An *atomic* activity that executes based on a schedule. Can be used to ensure a process does not get stuck waiting infinitely for an incoming message.

- Can only be the child of a *Choice* activity.
- When completed, the parent *Choice* activity will also complete.
- Can contain only one sub-activity.

## Wait



An *atomic* activity that waits for spawned children processes to complete before proceeding.

- If a process type is specified, the activity completes after all spawned processes of this type have completed execution.
- If no process type is specified, the activity completes after all spawned children processes have completed execution.
- This activity will wait only for sub-processes instantiated from the same process in which the activity is executed.
- If no nested processes are executing when this activity occurs, it completes immediately.
- A **Before** script is not allowed.
- **Notes:**
  - Under parallel activities of a process, there must not exist multiple Wait activities in parallel waiting for the same sub-process type to complete. Under such circumstances, it is possible for some Wait activities permanently waiting even when all such sub-processes are completed.
  - Wait activities for process and decision trees cannot use subflow processes.

## While



A complex activity that performs loops in the flow of processes.

- Activities that run within the loop are children of the *While* activity.
- Loop performs as long as the condition script of the *While* activity returns true.
- The process engine checks the condition script for the first time when the *While* activity starts and after every loop iteration of loop, except when the *Break* script returns true in the middle of the loop.
- Loop may be finished in the middle if it has a *Break* activity inside and the *Break* activity's condition script returns true.
- The **Before** script of the *While* activity runs only before the first iteration of the loop.
- The **After** script of the *While* activity runs only after the loop completes, with a return value of the condition script as a result (any *While* or *Break*).

## Define a Choice Process

This section displays an example of a Choice Process setup.

### 1. External Service Interface

To define an Operation activity that contains a Notification, an external interface must be created. Create an external interface with the **Operation Type** field set to **Notification**.

Name:	NOperation
Label:	Interface Operation Notification
Description:	
Input:	<NONE>
Output:	<NONE>
Fault:	<NONE>
Operation type:	Notification
Privileges:	

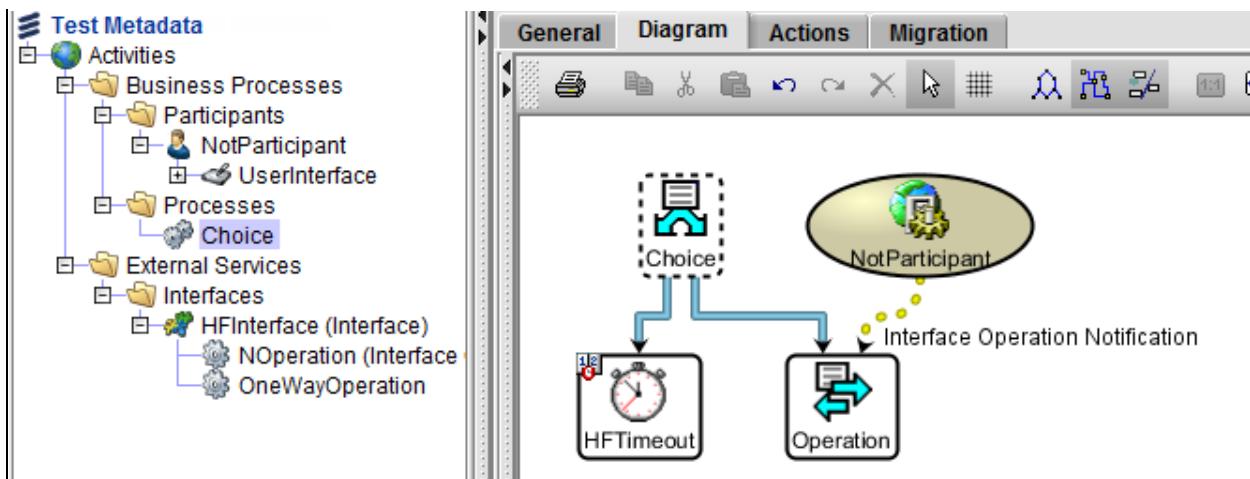
### 2. Participant

Create a participant associated with your external interface.

Name:	NotParticipant	<input type="checkbox"/> Private	<input type="checkbox"/> Restricted	<input type="checkbox"/> Deprecated
Label:	NotParticipant	...		
Description:		<input type="checkbox"/>		
Task type:	<input checked="" type="radio"/> Automated	<input type="radio"/> Manual		
Actor:	Activities.HFInterface (Interface)			
<input type="checkbox"/> Invoke directly <input type="checkbox"/> Return error to process				

### 3. Create a Choice Process Flow Diagram

The following figure shows the process flow diagram for a *Choice*. There is a minimum of two activities that must be defined. In this diagram, there are three activities: *On Exception*, *Timeout* and *Operation*. In the **Choice** process, the system will follow one of the branches or sub-activities of the choice. The **Choice** branch will wait until it either receives a Notification (in that case it will follow the appropriate *Operation* branch with that Notification specified), the system has timed out (it will follow the *Timeout* branch) or there is an exception.



## Operation

When creating an Operation Activity as a branch of **Choice**, a participant must be defined as well as it needs to be linked to an External Interface of the Type Notification (refer to Step 1):

General		Actions		Schedule		Activities	
Name:	Operation						
Label:	Operation						
Description:							
Participant:	<input type="button" value="Activities.NotParticipant"/> <input style="float: right;" type="button" value="..."/>						
Operation:	<input type="button" value="Activities.HFInterface.NOperation - Notification"/> <input style="float: right;" type="button" value="..."/>						
Participant instance:							
Milestone:	<input checked="" type="checkbox"/>						
<input type="checkbox"/> Allows rerun <input type="checkbox"/> Invoke directly							

## Common Attributes of Process Activities

---

Process activities can have the following optional attributes defined (unless specified in [Process Activities Types](#)):

### Before Script

A set of JavaScript statements that are executed before an activity-specific function is performed. For example, the **Before** script of an *operation* activity would usually initialize input message data (Document or Data Structure).

### After Script

A set of JavaScript statements that are executed after an activity-specific function is performed. For *complex* activities, the script is executed after all sub-activities are completed. For example, the **After** script of the *request-response* Operation activity would usually process output message data (Document or Data Structure).

### Condition Script

A Condition script can be used to skip an activity altogether.

### Duration

Specifies the maximum amount of time (in seconds) in which the activity should complete (see [Process Status](#)). An activity is always constrained by any time constraint placed on the parent activity, with the most restrictive time constraint taking precedence.

The Process Management product will automatically generate a *timeout* exception for activities that fail to complete in time. The product also limits the activity duration to a maximum period of 6 months (from activity start time).

When a duration attribute specifies a script for duration value calculation, this script can return either the number of seconds (maximum amount of time in which the activity should complete) or a specific date (date by which activity should complete). A value of 0 (zero) returned from the script will disable the duration timeout.

The following table shows the return values for non-dynamic scripts and their expected actions:

Return Value	Action
Null	No timeout exception occurs.
0 (zero)	No timeout exception occurs.
Negative number	No timeout exception is thrown if the duration script returns a negative value. Additionally, the script is no longer called.
Positive number	A timeout exception is thrown after the time returned by script.

In addition, when a duration attribute specifies a script for duration value calculation, it can be marked as dynamic. When a dynamic duration is set, the Process Management product will re-evaluate the script when the duration time has expired.

The following table shows the return values for dynamic scripts and their expected actions:

Return Value	Action
Null (first occurrence)	No timeout exception occurs. The script stops running.
0 (zero) (first occurrence)	No timeout exception occurs. The script stops running.
Negative number	No timeout exception is thrown if the duration script returns a negative value. Additionally, the script is no longer called.
Positive number	The Process Engine waits for this amount of time and then calls the script again to receive a new time to wait. However, if the script returns null, 0, or a negative number (does not apply to a first call to the script -- applies to the second and subsequent script calls), a timeout exception is thrown.

### Schedule

Specifies the time instance at which an activity should begin processing. It is conceivable that an activity will be forced to complete before its scheduled time due to a duration constraint placed on the activity.

The actual time instance at which an activity will be executed depends on the process priority and the number of activities pending execution (the Process Engine has a fixed number of activity execution threads). The Process Management product only guarantees that the activity will not be executed before the scheduled time.

The product checks the type of value returned from the duration script. Expected types are as follows:

- Date - the date object that is created and initialized in the script
- Number - an integer number that is parsed and is a value in seconds

### Milestone

Specifies that the activity has a special business meaning associated with reaching a certain step in the business process.

The Process Management product will record additional information for activities:

- **Milestone Code:** User-defined code assigned in the activity milestone properties. Milestone codes are defined in a milestone enumeration (enumeration that is based on the Milestone data type). The milestone code is used to identify the milestone for display (milestone description) and calculation (retrieving all milestones with the same code) purposes.
- **Order instance:** If the process has an Order instance associated with it.
- **Activity start time:** The time that the activity started.
- **Activity end time:** For operation activities, the time the message was added to the process queue.
- **User participant type:** Defined if the activity is a User Participant Operation.
- **User ID:** Defined if the activity is a User Participant Operation.
- **Due date:** Defined if the activity is a User Participant Operation. This is a due date as calculated based on the User Participant definition.
- **Expected due date:** Specified in the activity milestone properties.
- **Activity duration:** Number of business seconds (based on calendars) taken to complete this activity.
- **Process duration:** Number of business seconds (based on calendars) taken to complete this activity relative to the process start time.

This information can be used for the following purposes:

- To view the business status of a particular Order instance. This permits an understanding of where the Order is within a business process, which milestones are completed (and if completed on time), which milestones are still pending and which milestones are late.
- To create reports that show the average time taken to achieve milestones. This will help identify the problem areas in the business process and provide metrics on business process efficiency.

Milestone definition can be created by creating a [Data Type](#) that extends from `cwf.milestone` and overrides `cwf.allMilestones`, and with [enumeration](#) code defined for each milestone definition.

The product checks the type of value returned from the duration script. Expected types are as follows:

- Date - the date object that is created and initialized in the script
- Number - an integer number that is parsed and is a value in seconds

Milestone information is stored in the CWMILESTONE table of the process product database. The table can be extended to include additional columns (custom attributes). Those additional attributes can be populated using the Process JavaScript function `updateProcessMilestone` (refer to the [JavaScript Extensions HTML documentation](#)).

#### Metadata Objects

The following table describes the metadata object pertaining to milestones:

Metadata Object	Description
<code>cwf.milestoneDoc</code>	This object allows you to initialize, add, and update process milestones. It replaces the <code>cwf_pm.milestone</code> document. It has the same structure (variable names, variable order, and mapping to database) as the old document to support backward compatibility. The document is located in system metadata and has no child UI.

#### Events

The following table describes events pertaining to milestones. Parameters that are bolded signify that they are mandatory:

Event	Parameters	Description
<code>PROCESS_MILESTONE_COMPLETE</code>	<ul style="list-style-type: none"><li>• <b>process</b></li><li>• <b>activity</b></li><li>• <b>endTime</b></li><li>• <b>activityDuration</b></li><li>• <b>processDuration</b></li><li>• <b>participantId</b></li><li>• <b>participantType</b></li><li>• <b>participantsUser</b></li><li>• <b>expDueDate</b></li></ul>	This event updates the activity milestone endTime, if needed (that is, when activity ends).
<code>PROCESS_MILESTONE_CREATE</code>	<ul style="list-style-type: none"><li>• <b>process</b></li><li>• <b>procMetadataActivities</b></li></ul>	This event initializes an unconditional milestone for each activity in a process.
<code>PROCESS_MILESTONE_INSERT</code>	<ul style="list-style-type: none"><li>• <b>process</b></li><li>• <b>activity</b></li><li>• <b>expDueDate</b></li><li>• <b>code</b></li></ul>	This event inserts a new record in the cwMilestone table. The event does not do anything when there is no milestone request.
<code>PROCESS_MILESTONE_UPDATE</code>	<ul style="list-style-type: none"><li>• <b>process</b></li><li>• <b>activity</b></li><li>• <b>expDueDate</b></li><li>• <b>code</b></li></ul>	This event updates a record in the cwMilestone table. The event does not do anything when there is no milestone request.
<code>PROCESS_MILESTONE_UPDATE_CUSTOM</code>	<ul style="list-style-type: none"><li>• <b>processId</b></li><li>• <b>code</b></li><li>• <b>doc</b></li><li>• <b>endTime</b></li></ul>	<p>This event updates the process milestone table for a given milestone code. If there is more than one milestone row for a given code for this process, the latest inserted row is updated. If both the doc and endDate parameters are omitted or are null, the END_TIME column of the specified milestone row is updated with the current time.</p> <p>The doc parameter is an optional Document object that contains custom data for the milestone table. The document should be either the system milestone document or extend the <code>cwf:milestoneDoc</code> system milestone document. If the specified document is not the system milestone document or an extension of it, an exception occurs. The milestone table is updated with every non-null attribute in the document.</p>



## Process Activity Properties

The process activity properties can be defined and changed by selecting a process under the Processes node. The title bar of the dialog displays the activity type and label. The tabs contained in the dialog and the fields displayed on each tab will depend on the activity type. The tabs may include the following: **General**, **Actions**, **Schedule**, and **Activities**.

### General Tab

The general activity properties are defined on the **General** tab of the property dialog.

*General tab of the Activities properties for a Choice activity with a milestone.*

The screenshot shows the 'General' tab of a property dialog for a 'Choice' activity. The tab contains several configuration fields:

- Name:** choice1
- Label:** choice1
- Description:** (empty field with a pencil icon)
- Milestone:** Activated (checkbox checked)
- Conditional:** (checkbox unchecked)
- Expected due date:** Days: 1, Hours: (empty), Minutes: (empty)
- Script:** Dynamic (checkbox checked)  
function cwOnDuration() { // Calculates the duration when rec  
var process = this;  
}  
The script block contains a function definition with a comment and a variable declaration.
- Calendar:** Use calendar (checkbox checked)
- Allows rerun:** (checkbox unchecked)

*General tab of the Activities properties for an Operation activity.*

Name:	getCredit
Label:	getCredit
Description:	<input type="text"/>
Participant:	interfaceA.userP <input type="button" value="▼"/>
Operation:	interfaceA.creditAssure.creditCheck - Request-Response <input type="button" value="▼"/>
Participant instance:	<None> <input type="button" value="▼"/>
Milestone:	<input type="checkbox"/> <input checked="" type="checkbox"/>
<input type="checkbox"/> Allows rerun <input type="checkbox"/> Invoke directly	

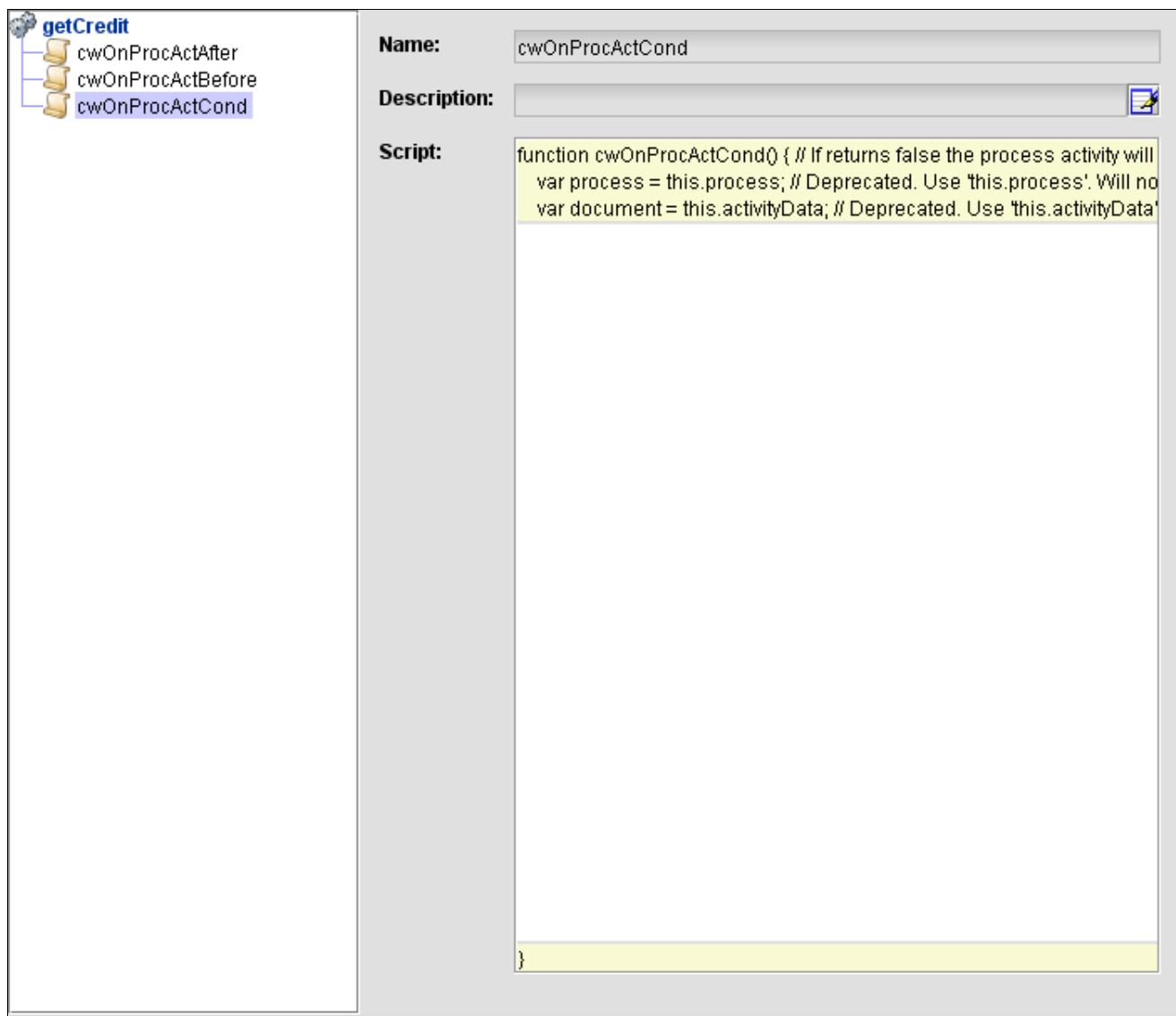
Field	Description	Type
<b>Name</b>	Mandatory. Name of the Activity within the namespace.	All types
<b>Label</b>	Mandatory. Label to identify the Activity.	All types
<b>Description</b>	Description of the Activity for documentation purposes.	All types
<b>Milestone</b>	The current activity is defined with a <a href="#">milestone</a> .	All types except Compensate
<b>Conditional</b>	Indicates that this milestone is conditional on execution of this activity.	All types
<b>Expected due date</b>	Mandatory if the area is visible. The area is visible when the <b>Milestone</b> field has a selection. Used for viewing purposes.	All types
<b>Allows rerun</b>	Indicates the activity can be rerun without impact to the process as a whole. The state of activities marked with this flag are not recorded into the database until an activity without this flag is run. Skillful use of this flag can result in increased process performance.  <b>Note:</b> In PE mode 0 (low speed debug mode), all activity Start/End times are stored. Please refer to <i>Configuration User Guide</i> for more details.	All types
<b>Alert</b>	Associates an Alert that is predefined.	Alert
<b>Condition</b>	Specifies a condition JavaScript script which is evaluated to determine whether or not to execute this case.	Case
<b>Compensates</b>	Specifies a compensate activity.	Compensate
<b>Exception</b>	Associates an Exception that is predefined.	Exception and On Exception
<b>Participant</b>	Associates a Participant to send a task to or to receive notification from.	Operation
<b>Operation</b>	The Participant's <i>one-way</i> , <i>request-response</i> , or <i>notification</i> Operation. For sending a task to or receiving a response from the Participant.	Operation
<b>Participant instance</b>	Used only for sending a task to a specific Participant from whom the system has already received notification.	Operation
<b>Invoke directly</b>	If checked, the activity will not be invoked through the interface process.  <b>Note:</b> For request/response operations, if the <b>Invoke Directly</b> property is checked, the <b>Retry count</b> and <b>Retry delay</b> settings in the Configuration application are not used.	Operation
<b>Activity</b>	Specifies an activity to repeat.	Repeat
<b>Signal</b>	Specifies a Signal that is predefined.	Signal

<b>Sub-Flow</b>	Specifies a Process that is predefined. The Process activities will be treated as a sub-flow.	Sub-Flow
<b>Sub-Process</b>	Specifies a Process that is predefined that will be treated as a sub-process.	Sub-Process
<b>Synchronization</b>	If checked, the sub-process activity does not complete until the sub-process completes. If unchecked (default), the sub-process activity completes immediately while the sub-process runs asynchronously. In both cases, the activities in sub-process runs in a separate process instance from the parent.	Sub-Process
<b>Exclusive</b>	If unchecked, all the case activities that evaluate to true will be executed. When checked, only the first Case activity that evaluates to true will be executed.	Switch
<b>Signal</b>	Specifies a Signal that is predefined for which the Synchronize activity waits.	Synchronize
<b>Sub-Process</b>	Specifies a Process that is predefined that will be treated as a sub-process to wait for.	Wait

## Actions Tab

The activities may have any of **Before**, **After** or **Condition** scripts in the **Actions** tab in the property dialog.

The **Before** tree node contains a script that will be executed before the activity starts. The **After** tree node contains a script that will be executed after the activity is completed. If one of the scripts is not relevant for this type of activity, only the applicable field will be shown.



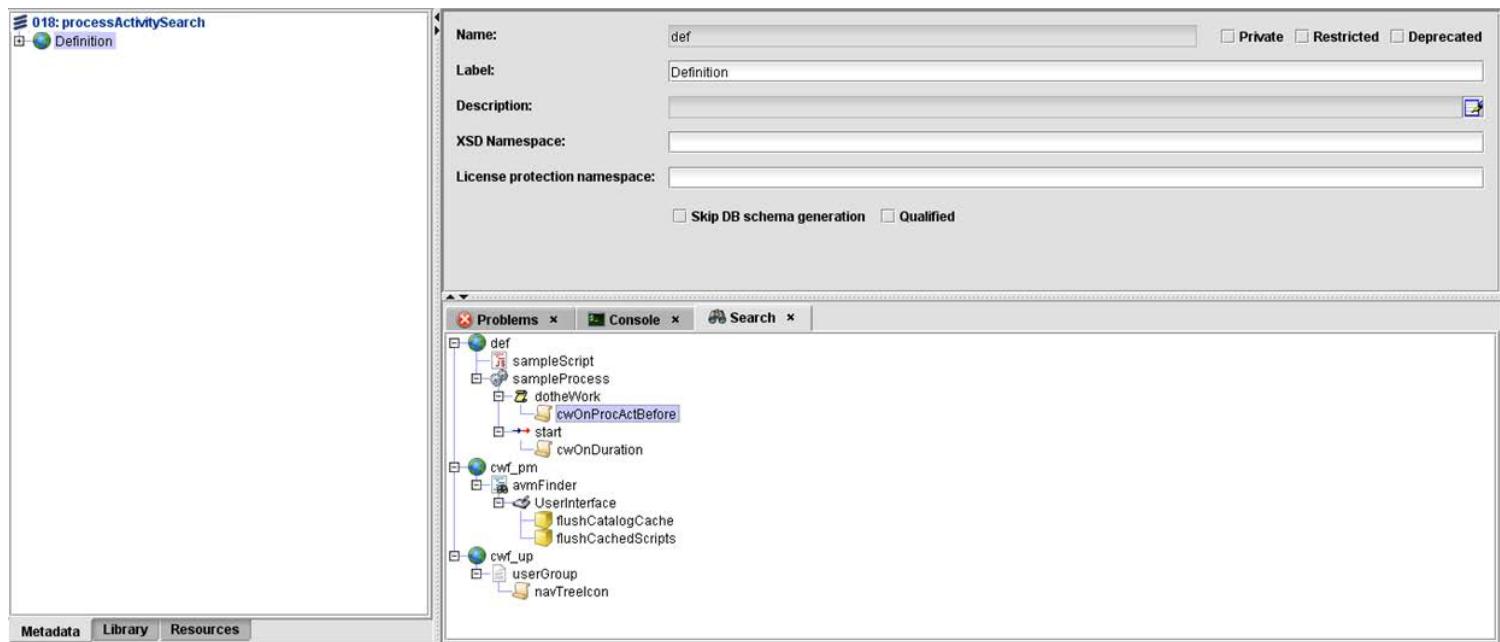
Notably, process activity scripts have access to the following parameters:

- *this.process*: The process object of the activity, which gives access to all types of process data, and allows the performing of different process functions (such as access to process Document, process Order or process activity information).
- *this.activityData*: This is an activity-specific Document (such as an input Document in an Operation activity). This value can be null.

## Notes:

- Refer to the [JavaScript Extensions HTML documentation: Process Activity object](#) for reference.
- Velocity Studio validates process activity scripts. If you have validation errors in your metadata, you must fix them before you can run your metadata project.

In Velocity Studio's Search pane, you can double-click an activity (for example, cwOnProcActBefore) or one of its scripts to launch the Script field for editing.



## Schedule Tab

All activities have the **Schedule** tab in the property dialog.

### Duration

**Use duration**

Days:  Hours:  Minutes:

Script:  **Dynamic**

```
function cwOnDuration() { // Calculates the duration when requested
    var process = this;
}

}
```

Reference:  Relative to end <None>

Calendar:  Use calendar

### Schedule

**Use schedule:**  Duration  Instance

Days:  Hours:  Minutes:

Script: 

```
function cwOnDuration() { // Calculates the duration when requested
    var process = this;
}

}
```

Reference:  Relative to end <None>

Calendar:  Use calendar

The tab is divided into these sections:

- **Duration** pane: Area used to define a time interval needed to complete the activity. If the **Use duration** check box is checked, the **Duration** area will be enabled and the duration can be defined. Table below describes the fields.
- **Schedule** pane: Area used to schedule the activity execution. If the **Use schedule** checkbox is checked, the **Schedule** area will be enabled and the schedule can be defined. The schedule can be based on either a duration or an instance. If the **Duration** radio button is selected then duration properties are used to define the schedule. The same Table below describes the fields of the duration properties in Schedule pane, except for *Dynamic* field which is exclusive to Duration Pane. If the **Instance** radio button is selected then an instance pane is used to define the schedule. The instance pane schedules activities on a calendar basis and has Daily, Weekly and Monthly options.

- *Daily*: The activity will be started daily at the specified start time.

### Schedule

**Use schedule:**  Duration  Instance

Schedule:  Start time:  Hour(s)  Minute(s)

Interval: Days:  Hours:  Minutes:

- *Weekly*: The activity will be started every week on the specified day of week at the specified start time.

**Schedule**

**Use schedule:**  Duration  Instance

Schedule: **Weekly** Start time: **00** Hour(s) **00** Minute(s)

Every week on: **Monday**

Interval: Days:  Hours:  Minutes:

- Monthly: The activity will be started every month. When the **Day** radio-button is selected, the activity will be started every month on the specified day of the month at the specified start time. When the **The** radio-button is selected, the activity will be started every month on the specified occurrence of the day or day of week in the month at the specified start time.

**Schedule**

**Use schedule:**  Duration  Instance

Schedule: **Monthly** Start time: **00** Hour(s) **00** Minute(s)

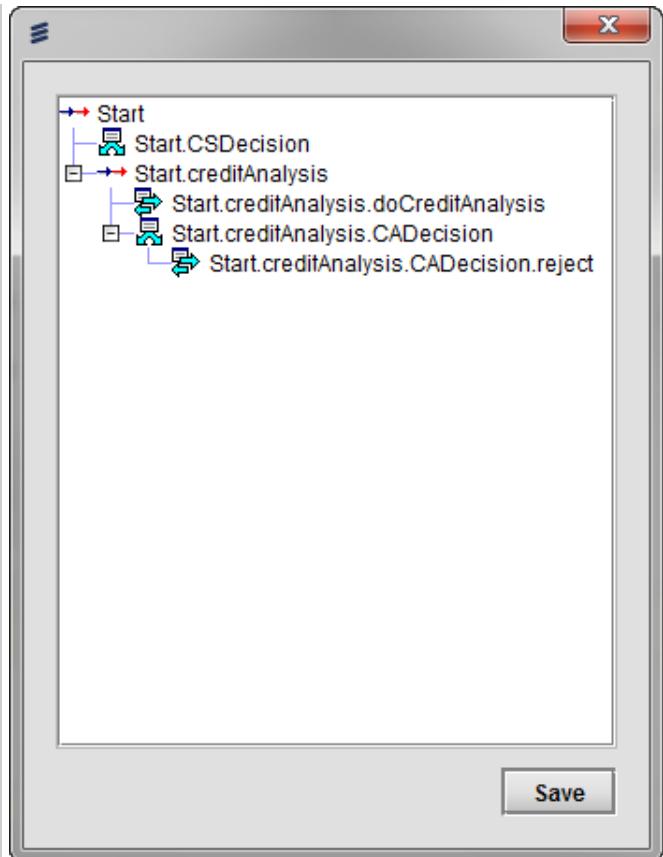
Day **1** of the month

The **first** **Monday** of the month

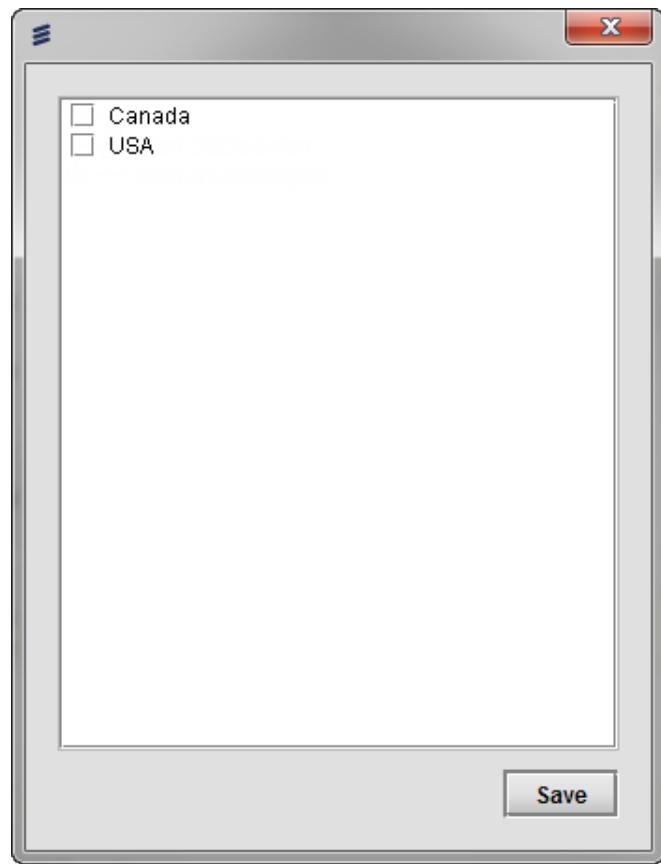
Interval: Days:  Hours:  Minutes:

The **Interval** fields in the **Schedule instance** pane specify a time interval in which the activity will be repeatedly executed by the system.

Field	Description
<b>Days</b>	Way to define duration time. Cannot be used if a <b>Script</b> is defined.
<b>Hours</b>	Way to define duration time. Cannot be used if a <b>Script</b> is defined.
<b>Minutes</b>	Way to define duration time. Cannot be used if a <b>Script</b> is defined.
<b>Script</b>	Way to define duration time. Cannot be used if <b>Days</b> , <b>Hours</b> and <b>Minutes</b> are defined.  <b>Note:</b> When returning a Date object type, the schedule script overrides the calendar. As a result, the processing engine does not use the calendar if the script returns the Date object.
<b>Dynamic</b>	If checked, a <b>Script</b> is specified and should be dynamically updated at runtime.
<b>Reference</b>	Defines a start point of the duration clock. A reference activity can be selected from the combo box or by using the <b>Select Activity</b> dialog that shows the activity hierarchy. To open the dialog, click the button to the right of the combo box.



<b>Relative to end</b>	Check box used to define a start point of the duration clock for the activity. If unchecked, the duration clock will start when the <b>Reference</b> runtime activity starts executing. If checked the clock starts when the <b>Reference</b> runtime activity finishes execution.
<b>Use Calendar</b>	If checked, the <b>Calendar</b> field will be enabled so that a business calendar can be defined to be used to do duration time calculations.
<b>Calendar</b>	Defines the business calendar that is used when determining the duration. The <i>&lt;Default&gt;</i> selection means the default calendar selected in the User Profile Administration application will be used. To select the calendar, click the button to the right side of the <b>Calendar</b> field. The <b>Select Calendar</b> dialog opens.

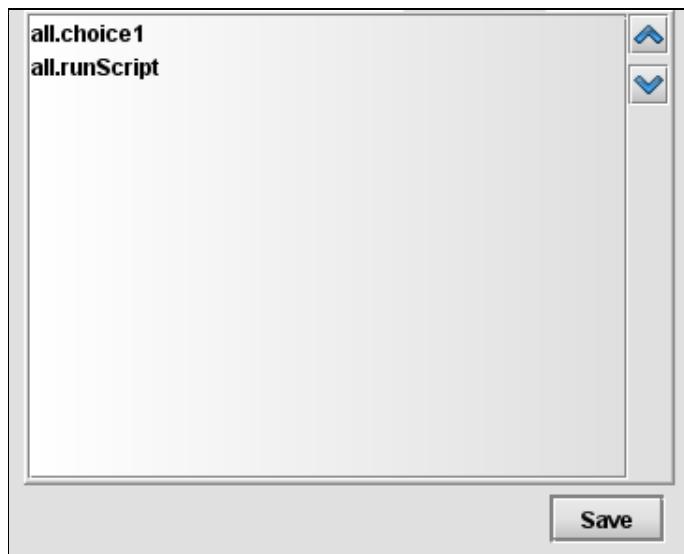


Not choosing an entry results in <Default> selection.

**Note:** Calendars are defined and managed in the System Administration application's [Calendar page](#), and the enumeration of these selections in Velocity Studio are defined by the *cwf.allCalendars* Data Type.

## Activities Tab

When an activity has children activities, the **Activities** tab will contain a list which lists all the children activities. The children activities will be executed accordingly to their order in the list, from top to bottom. The arrow buttons located at the side of the list are used to move the selected element in the list to the desired location, thus changing the execution order of the selected activity.

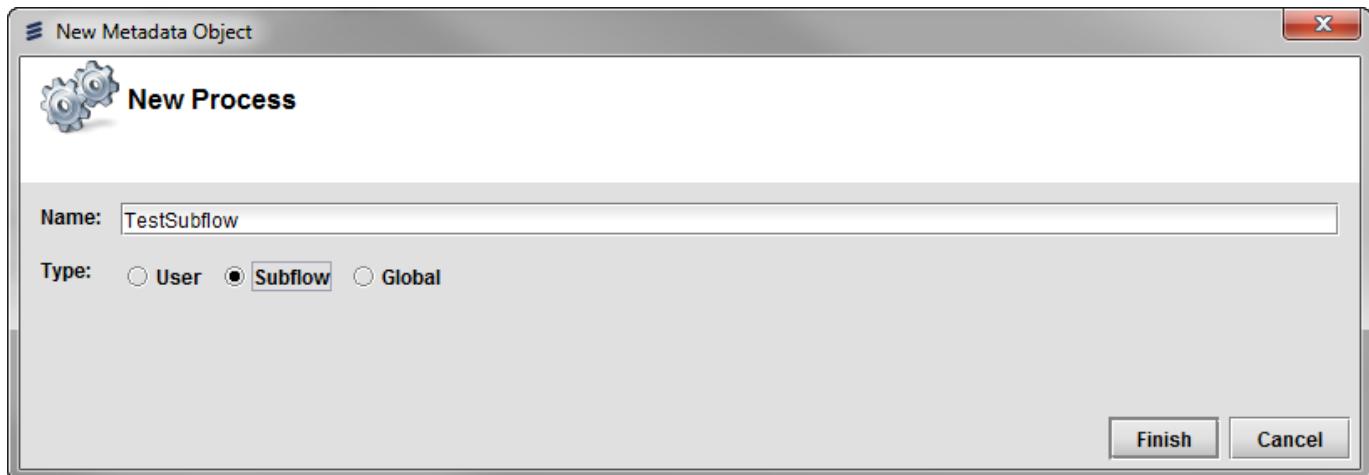


## Building Processes

To create a new process, do one of the following:

In the **Metadata** tab of **Navigation** pane, either:

1. Right-click a Namespace and select **New ...** from the menu.
2. The **New Metadata Object** wizard appears as a popup. Expand **Business Processes**, select **Process**, and then click the **Next** button.
3. Step two of the wizard appears. For the **Name** field, enter the process name that you want, which must conform to JavaScript naming conventions.
4. For **Type**, select one of the following options:
  - **User** for user processes
  - **Subflows** for subflow processes
  - **Global** for external service processes or global process processes



5. Click the **Finish** button.

• Or

1. Right-click the **Processes** folder or the **Business Processes** folder in a Namespace, if it is present. Select **New Process**.
2. Continue by following step 3 from the first set of steps.

After the wizard creates the process, you can no longer view or change the process type, and different process properties are available based on the selected type. The newly created process is added under **Processes** node.

**Note:** When creating a process as a Global type, the Migration tab is not available. When creating a process as a User type, the **Interface** field is not available on the General tab.

Your newly created process appears in your namespace, under **Business Processes > Processes** in the metadata tree:

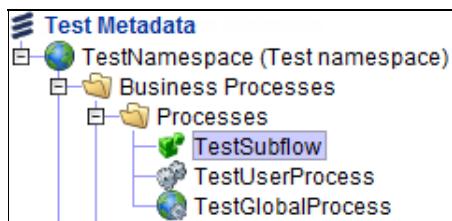


The icons for user, subflow, and global processes are as follows:

Icon	Description
	User process
	Subflow process
	Global process

## Process General Properties

Clicking a process from the metadata tree displays its general process properties, which are defined on the **General** tab.



The fields visible on the tab change, depending on the **Type** field selection (that is, Global, Subflow, or User).

### General Properties for a Global Process

The General properties tab for a global process is as follows:

General		Diagram	Actions	Subflows
Name:	TestGlobalProcess	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated		
Label:	TestGlobalProcess	...		
Description:	<input style="width: 100px; height: 20px; vertical-align: middle;" type="button" value="..."/>			
Type:	Global			
Document:	<NONE>	<input style="width: 100px; height: 20px; vertical-align: middle;" type="button" value="..."/>		
<input type="checkbox"/> Store in a separate table				
<input type="checkbox"/> Use parent process document				
Priority:	<input style="width: 100px; height: 20px; vertical-align: middle;" type="button" value="Normal"/>			
Interface:	<NONE>			
Revision:	1			
<input type="checkbox"/> Memory Only				
<input type="checkbox"/> If in error restart from last running activities				

### General Properties for a Subflow Process

The General properties tab for a subflow process is as follows:

**General** **Diagram** **Actions** **Subflows**

Name:	TestSubflow	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated
Label:	TestSubflow	...
Description:		
Type:	Sub-flow	...
Revision:	1	

**Note:** The property pane for a top-level subflow process only contains the **Name**, **Label**, **Description**, and **Revision** fields. The remaining properties are inherited from the process using the subflow.

## General Properties for a User Process

The following is the General properties tab for a user process:

**General** **Diagram** **Actions** **Subflows** **Migration**

Name:	TestUserProcess	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated
Label:	TestUserProcess	...
Description:		
Type:	User	...
Document:	<NONE>	▼
<input type="checkbox"/> Store in a separate table <input type="checkbox"/> Use parent process document		
Priority:	Normal	▼
Revision:	1	
Stale Alert:	<NONE>	▼
<input type="checkbox"/> Stale exception <input type="checkbox"/> Memory Only		

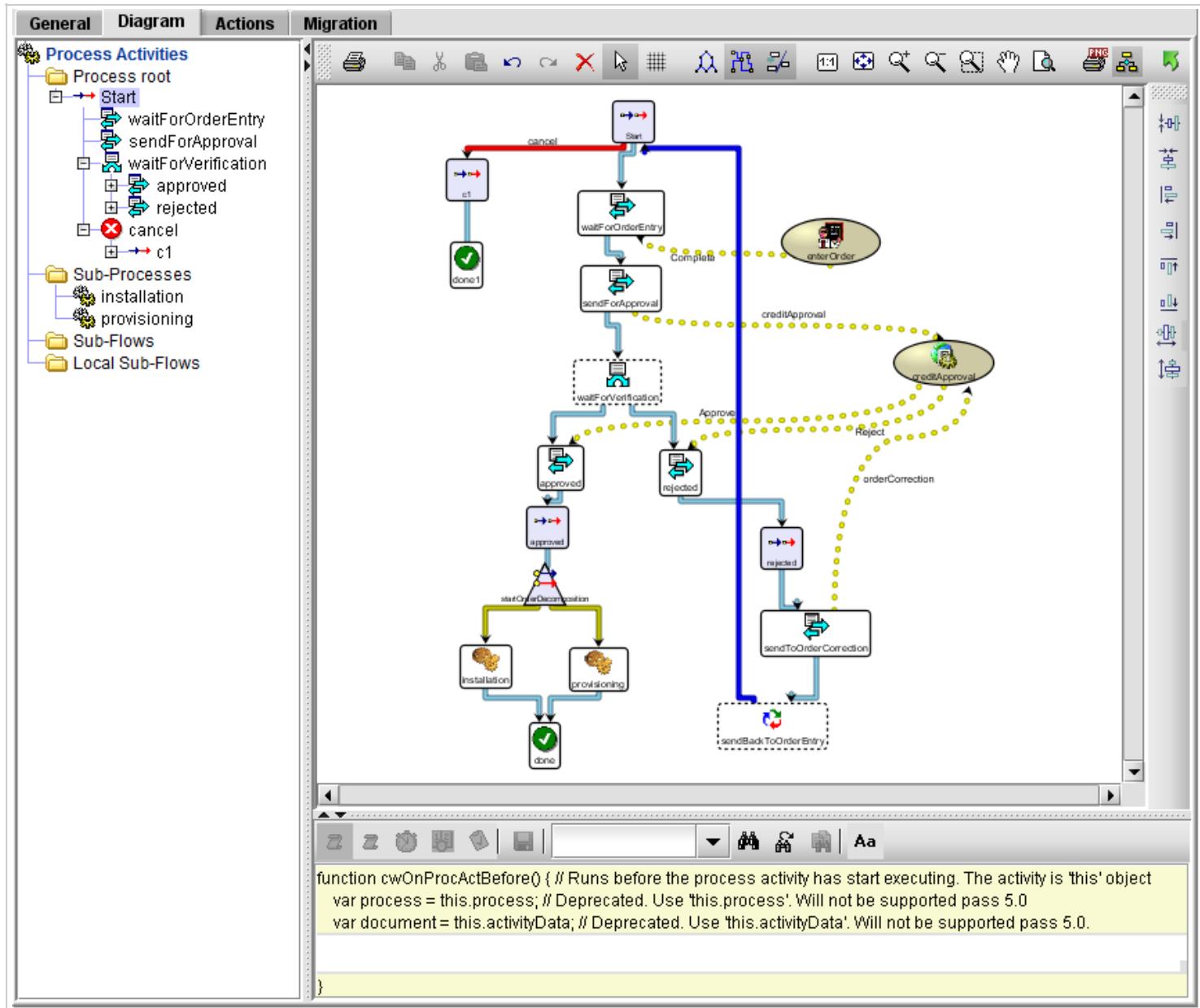
The General tab has the following fields, depending on the **Type** field selected:

Field	Description	Type
<b>Name</b>	Mandatory. Name of the Process within the namespace.	All
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).	All
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).	All
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.	All
<b>Label</b>	Mandatory. Label to identify the Process.	All
<b>Description</b>	Description of the Process for documentation purposes.	All
<b>Type</b>	Mandatory; not changeable upon metadata object creation. The <a href="#">Process type</a> . The values include <i>Global</i> and <i>User</i> .	All
<b>Document</b>	The Process Document. If specified, an instance of the Document is created and assigned to each process instance. The	All

	process instance can store instance-specific data in this Document.	
<b>Store in a separate table</b>	If checked, the Process Document is stored in a separate database table. Otherwise, an unmapped process document is stored with the process.	User, Global
<b>Store as a separate document</b>	Available only when <b>Process Document</b> is assigned. If checked, the Process Document is stored in a separate database table, instead of as NCLOB in XML format in process table. Storing as a separate document enables readable access of the document to application metadata (for example, you may create report over this data); in addition, enabling this flag may have slight, but visible performance improvements.	All
<b>In error restart from last running activities</b>	By default, this property's checkbox is unchecked, meaning that the process engine restarts the process that is in an error state from the first activity. When this property is checked, the process engine restarts this global process in an error state from the last running activity when the error occurred.	Global
<b>Priority</b>	The user can assign the <a href="#">Process priority</a> . The default values include <i>High</i> , <i>Normal</i> , and <i>Low</i> if no customization is performed on <i>cwf.allPriorities</i> .	All
<b>Interface</b>	This field associates the global process with an interface so that messages can be sent to the global process. Set the <b>Interface</b> field before adding operation activities to the workflow for the operation to treat the specified interface as a default global participant. Operation activities are not automatically updated after changing an interface.	Global
<b>Revision</b>	Read-only. This is the revision number of the Process. The latest <a href="#">Process Revision</a> of this Process is at this revision number. <ul style="list-style-type: none"> <li>• Revision number is set to 1 for a newly created process.</li> <li>• Revision number is incremented by 1 every time a <a href="#">new Process Revision</a> is created, except when this is a newly created Process that is without Process Revisions.</li> </ul>	All
<b>Stale Alert</b>	At the process level for a user process, you can specify a metadata-defined alert to be issued in the event of a stale dated process. Such an alert can be configured to go to e-mail, interface, or a worklist.	User
<b>Stale Exception</b>	At the process level, a checkbox on the General tab for a user process that indicates whether a system-defined exception is thrown into a stale-dated process.	User
<b>Memory Only</b>	Select this checkbox to indicate that it is a memory-based process. Memory-based processes do not store their state. If they get an error or the process engine is restarted, these processes start from the beginning.	User, Global

## Diagram Tab

The **Diagram** tab of the Process presents the process activities and their relationships.



The tab is divided into the following panes. Some of them are minimized by default, which requires dragging the small triangular controls on the horizontal or vertical split bars between the panes to have them appear:

- **Process Activities** tree: The left side area shows a hierarchical tree of process, sub-process and sub-flow activities.
- **Graphic Diagram** pane: The right side top area contains a graphic diagram of the activities' relationships. The table below contains a description of the button functions.
- **Script**: The right side bottom area contains a toolbar and a JavaScript editing field that allows script editing for the selected activity. The toolbar has a set of buttons on its left side that allows fast switching between the activity scripts. The **Save** button is used to save all changes. An editable combo box and a set of buttons on the right side can be used to search for particular words in the current activity or in all process activities. The tool tips can be used to find out the button meaning.

The areas on the left and right sides of the **Diagram** tab are synchronized. If you click an item in the **Process Activities** tree, it is also selected in the **Graphic Diagram** pane, and vice versa.

The small triangular controls on the split bars between the sections can be used to hide some of the areas and to restore their previous size again.

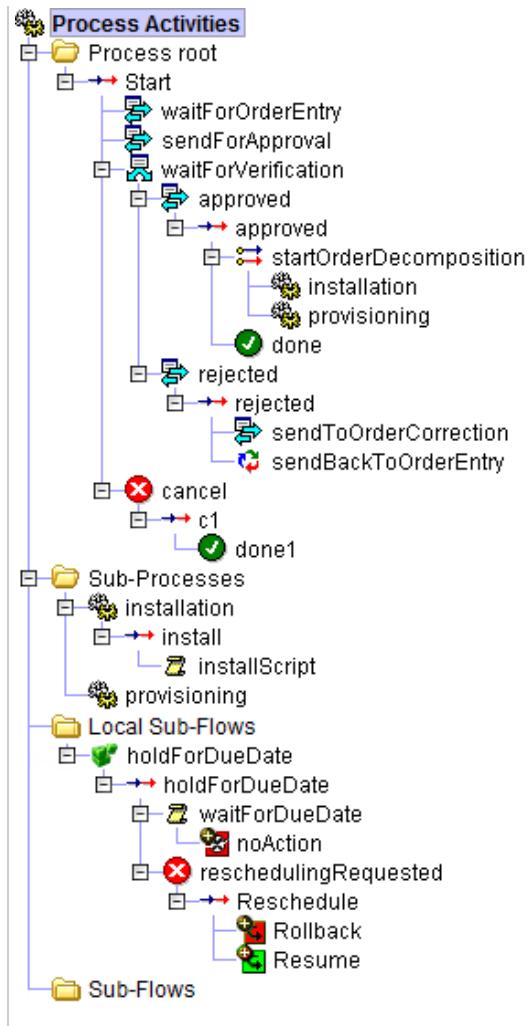
Description of button functions for the Diagram tab toolbar.

Button	Function
	Print the diagram.
	Copy selected activity.
	Cut selected activity.
	Paste selected activity.
	Undo action.
	Redo action.
	Delete selected activity.
	Select (shows the arrow cursor which should be used to select activities on the diagram).
	Grid (shows a grid on the diagram).
	Automatic node layout.
	Automatic link layout.
	Automatic labels layout.
	Show diagram in actual size.
	Fit the diagram into visible area.
	Zoom in.
	Zoom out.
	Zoom box (click the button, and then select an area on the diagram to zoom in).
	Pan (click the button then select a node on the diagram and drag it holding the left mouse button).
	Print Preview; a Preview pop-up box appears.
	Export as an image (exports a snapshot of the currently visible area into a JPG or PNG bitmap file which then can be imported into MS Word or any other application that supports that format).
	Flow direction (click the button then select an area on the diagram; the nodes and links included in the selection will be highlighted).
	Return to the previously viewed Process, sub-process or sub-tree.

The **Sub-Processes** folder of the **Process Activities** tree contains the sub-process activities as a separate tree. Click the sub-process item in the **Sub-Processes** folder to view the **sub-process** diagram in the **Graphic Diagram** pane. These sub-processes are read-only here; they can edit them by finding their metadata objects in **Processes** folder in Navigation Pane.

Similarly, the **Local Sub-Flows** folder of the **Process Activities** tree contains the sub-flow activities as a separate tree. The **Local Sub-flows** folder only stores local sub-flows. You can add a sub-flow by right-clicking the **Local Sub-Flows** folder and then select **Add Subflow** at the pop-up menu. To view or edit existing sub-flows in this Process, simply click the sub-flow item in the **Local Sub-Flows** folder to view and edit the **sub-flow** diagram in the **Graphic Diagram** pane.

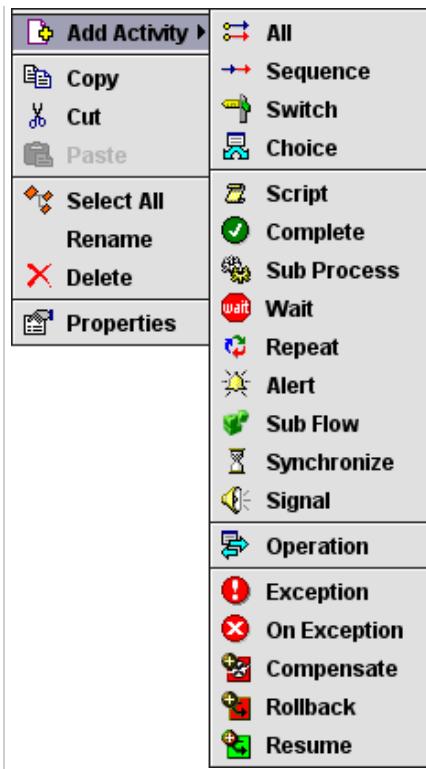
The **Sub-Flows** folder stores top-level sub-flows. Its behaviour is similar to the **Sub-Processes** folder.



## Diagram Elements

To add the first process activity node to the diagram, right-click the **Process root** folder. A right-click pop-up menu will appear where a process activity can be selected from the **Add Activity** sub-menu. A dialog will open where the activity properties can be defined and an activity node will be added in the **Graphic Diagram** pane.

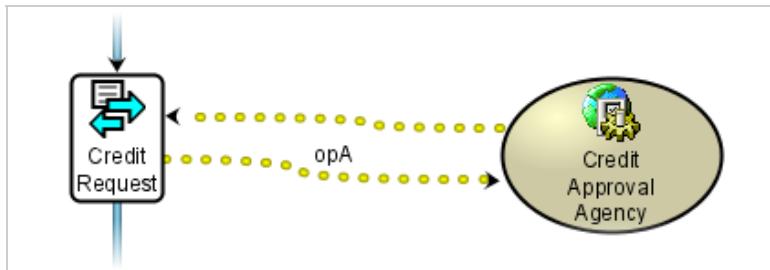
Once the top level process activity node has been added to the diagram, additional child level process activity nodes can be added by right-clicking the node in the **Process Activities** tree or in the **Graphic Diagram** pane. A pop-up menu will appear. The set of options in the menu will vary depending on the actions available for selected activity node. The table below describes the common pop-up menu items.



Menu Item	Description
Add Activity	Available for activities that have children activities. A list of allowed children activities is shown as a sub menu.
Copy	Copies the activity into the internal buffer.
Cut	Removes the activity from its current position in the tree after the paste operation.
Paste	Activated once <b>Copy</b> or <b>Cut</b> is selected.
Select All	Available if the activity has children activities. Highlights all children activities.
Rename	Rename the activity.
Delete	Deletes the activity.
Properties	Used to view and edit the activity properties. When selected, a dialog opens with the <a href="#">activity properties</a> .
Display	Available for sub-process and sub-flow activities only. Shows the sub-process or sub-flow activity as a separate diagram.
Edit	Available for sub-process and sub-flow activities only. Will select the corresponding Process in the objects tree in the <b>Navigation</b> pane of Velocity Studio. The Process properties will open that will allow editing of the diagram.

The diagram elements that can be added include the following:

As an Activity is added to the **Graphic Diagram** pane, an **Activity Node** is presented as an icon with an image corresponding to the [Activity type](#) and label. **Activity Links** are presented by solid lines ended with an arrow to link and indicate flow between activities. In addition, if an *Operation* Activity is added, a **Participant Node** is added in addition to the Operation Node. The Participation Node is presented as an oval icon with the participant name, with dotted yellow lines with arrow(s) indicating the message passing of the Operation between the Process runtime and Participant.



## Actions Tab

The **Actions** tab centralizes all scripts of the Processes.

The screenshot shows the 'Actions' tab in Velocity Studio. At the top, there are tabs for General, Diagram, Actions (which is selected), Subflows, and Migration. On the left, a tree view shows 'TestUserProcess' expanded, with 'myProcessActions' selected. The main area contains the following fields:

- Name:** myProcessActions
- Description:** (empty)
- Action Category:** (empty)
- Parameters:** A table with columns Name, Type, and Description. It currently has one row with an empty Name field and a 'New Row' button on the right.
- Script:** A code editor containing the script: 

```
function myProcessActions() {
```

 and a closing brace at the bottom.
- Return:** com.conceptwave.system.Void
- Return Description:** (empty)

On the right side of the main area, there are several small icons for managing the script, including arrows for sorting, a plus sign for adding, and a magnifying glass for search.

The **Actions** tab follows Velocity Studio's convention of a [Scripting Interface](#); please refer to it for details. There are no pre-defined scripts in Process Actions.

### Important Note

The number of scripts that can be run within a business process is limited to 200,000. The system will provide a warning message when the system has approached 90% of the limit.

Notably, the *Process* object is accessed by using *this* keyword in script, which gives access to all types of process data, and allows the performing of different process functions (such as access to process Document, process Order).

**Note:** Refer to the [JavaScript Extensions HTML documentation: Process object](#) for reference.

## Subflows Tab

The **Subflows** tab shows a list of global subflows that are used within the workflow of the current process and the revision number to be used for the subflow. Initially, when you create a process, no subflows are shown in the **Subflows** tab, as the process does not have a workflow yet. As you add subflow activities to the workflow, if these subflow activites are associated with global subflows, the system automatically adds the global subflows to the **Subflows** tab.

To access the **Subflows** tab, select the process from the left pane and then click on the **Subflows** tab. As an illustration, the following figure shows two global subflows listed under the **Subflows** tab of a process.

The screenshot shows the AVM Metadata interface with the 'Subflows' tab selected. On the left, there's a tree view of metadata, including 'interfaceA' which has 'Business Processes', 'Participants', 'Processes' (which contains 'Subflow1' and 'Subflow2'), and 'TestUserProcess'. On the right, the 'Subflows' tab displays a table titled 'Subflow Revisions' with two rows:

Subflow	Revision Number
interfaceA.Subflow2	1
interfaceA.Subflow1	2

Columns in the **Subflows** tab:

- **Subflow**: Lists the global subflows used in the workflow for the current process
- **Revision Number**: Allows you to choose the revision number for the subflow

### Select Revision Number for a Subflow

To select the revision number for a subflow listed in the **Subflows** tab, do the following:

1. Under the Revision Number column, click in the area for that subflow to display a drop-down list of revision numbers, as shown in the following figure.

The screenshot shows a dropdown menu for selecting a subflow revision. It lists three options: 2, 3, 2, and 1. The option '1' is highlighted, indicating it is the selected revision number.

2. Select the desired revision number.
3. Save changes to the metadata.

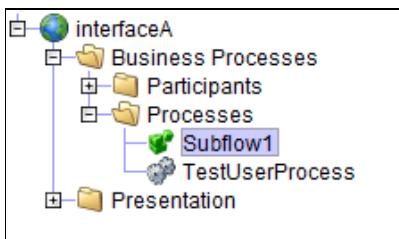
Once you select a subflow revision number for a subflow and save the metadata, all process subflow activities using this subflow start using that particular subflow revision.

**Note:** You can only change the subflow revision number for the latest process revision. Earlier process revisions are in read-only state and cannot be changed in any way.

### How Subflows Are Added to the Subflows Tab

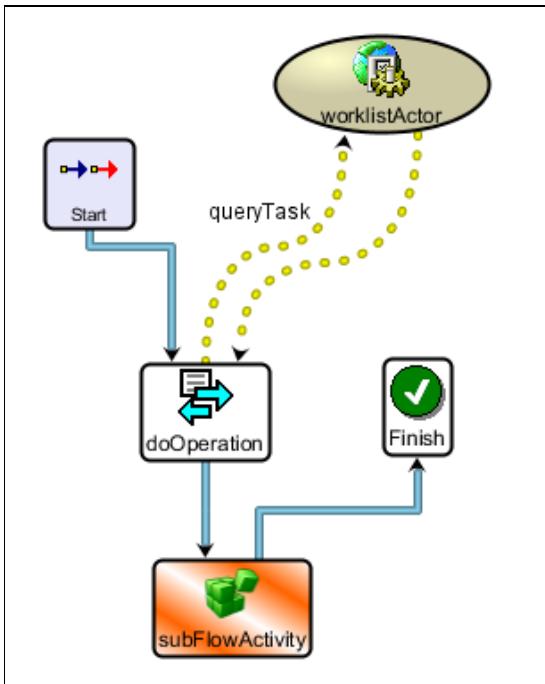
The system automatically adds subflows to the **Subflows** tab when you associate global subflows with subflow activities in the workflow of the process.

1. Add a global subflow. For more information, see [Subflow as a Top-level Process](#). The following figure shows a global subflow that has been added.

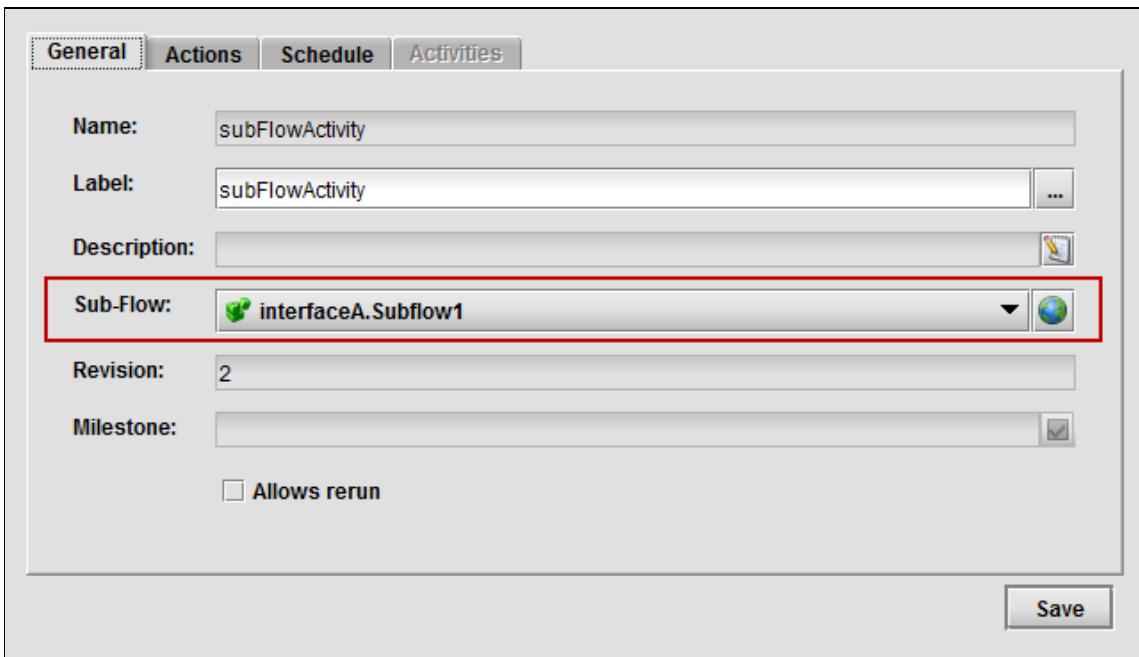


2. Click the **Diagram** tab for the subflow and add the workflow for the subflow.
3. Select the process from the left pane and click the **Diagram** tab for the process.

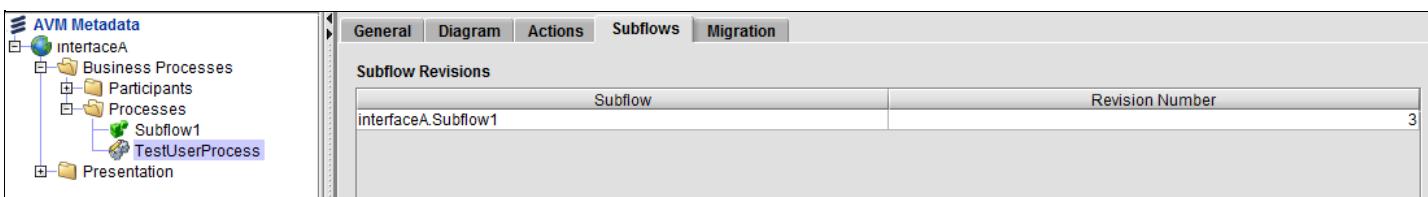
4. Add a subflow activity to the workflow of the process. A sample workflow is shown in the following figure.



5. Right-click the subflow activity and click **Properties** from the menu that appears to open the properties window for the activity. Under the **General** tab, select the global subflow created earlier in the **Sub-Flow** field, as shown in the following figure.



6. Click the **Save** button to close the dialog. The system automatically adds the global subflow to the **Subflows** tab for the process, as shown in the following figure.



7. Save changes to the metadata.

**Note:** When a subflow generates a new revision, a dialog appears, which displays a list of processes that use this subflow in the previous revision (that is, the revision before the newly generated revision). You can select the processes from this list that need to have their subflows re-adjusted to use the latest

subflow. After clicking the **OK** button, the selected process references are automatically re-adjusted. By default, this feature is available with the **Prompt to propagate subflow revision** setting. To change this setting, you can delect this option by clicking **File > Preferences** from the Velocity Studio menu bar, and then selecting **Setting**.

## Migration Tab

The **Migration** tab defines *migration records* of the Process, where each record specifies how to migrate running process instances of a specific Process Revision to the current Process (that is, the "latest" Process Revision).

You are encouraged to consult [Process Revisioning](#) before managing migration records.

Migration records of a Process are optional metadata. If a migration record does not exist for a particular Process Revision, those running process instances shall continue to execute by that Process Revision. This is known as *grand-parenting*. Thus, you would only define a migration record when there are (uncompleted) process instances for a Process Revision in the target environment, and you do not want grand-parenting to occur.

The screenshot shows the 'Migration' tab selected in a process editor. At the top, there are tabs for General, Diagram, Actions, and Migration. Below the tabs is a section titled 'Migration Records' containing a table with four columns: Revision, Migration Action, and Script. The table has four rows with the following data:

Revision	Migration Action	Script
2	<Migrate>	interfaceA.orderProcess.myProcessActions
1	<Suspend>	<None>
3	<Terminate>	<None>
5	<Suspend>	<None>

At the bottom of the 'Migration Records' section, there are three buttons: 'Add Migration', 'for revision 1', and 'Delete Migration'.

Field	Description
Revision	The Process Revision number to migrate from. This is a fixed field that identifies the migration record, and thus cannot be changed after the record is added.
Migration Action	Specifies how this Process revision should be migrated. Can have one of the following values: <ul style="list-style-type: none"><li><b>Suspend</b>: Suspends the process instance; sets the process instance's <a href="#">status</a> to <i>Suspended</i>, and executes the migration <b>Script</b> if any.</li><li><b>Terminate</b>: Terminates the process instance; sets the process instance's <a href="#">status</a> to <i>Terminated</i>, and executes the migration <b>Script</b> if any.</li><li><b>Migrate</b>: Executes the migration <b>Script</b> if any. Then change the process instance's revision number to the current revision. Continue to execute the instance, based on the current Process revision.</li></ul>
Script	Specifies a script that will be executed during process migration. The choice of scripts are selected from the Process' Actions (see <a href="#">Actions tab</a> ). This migration script runs with ' <i>this</i> ' object being the current process instance, which notably includes access to the Process Document and Process Order. The metadata object of <i>this</i> is the process metadata, not the revision metadata. The migration script receives one parameter that is the process instance revision number.

**Note:** Migration converts Process Revision to Process (and not revision to revision). Because Process Revision metadata objects are independent and not related to each other, there is no derivation, and thus re-use, in properties and scripts among Process Revisions. Indeed, migration scripts in different

Process Revisions may be completely different even with the same names.

## Common Migration Scenarios

There are several common migration scenarios. The following describes how each scenario can be accomplished in this **Migration** tab:

Scenario	Migration Action	Script
<b>Terminate process instance and start new one.</b>	<b>Terminate</b>	<p>Analyze the process instance, and then changes the order content and creates and initializes the process document for a new process. At the end the script starts a new process. This is the migration scenario when the process structure is changed.</p> <p>The latest Process Revision has to be able to skip (or compensate) the already performed activities and start from the right point in the new process instance. This approach, a <i>controlled restart</i>, needs a special design for the new process version. For instance, migration of a relatively small and linear process can be done with the following:</p> <ul style="list-style-type: none"> <li>• Create a new Process Revision that conditionally executes all major steps of the process. The conditions should be based on the process state computed from the process document and/or process order.</li> <li>• Create migration record that terminates the current process instance and starts the latest Process Revision with the corresponding changes in the process document or/and order. The changes are computed based on the current process state and the state of some process instance activities.</li> </ul> <p>As an illustration,</p> <p>the new instance starts from the beginning and skips some activities depending on the condition which is calculated entirely based on the process state. This should be done using the activity condition scripts that are available for all activities except the case activities and the operations of the choice activity. Note that if a group activity such as Sequence or All is skipped by the condition script all its children (the whole sub-tree starting at the activity) are skipped too.</p>
<b>Suspend process instance until the decision is made to how to migrate it.</b>	<b>Suspend</b>	<p><i>None.</i> Useful in the situation when there are errors in the process that can be corrected only manually. After the corrections are done, the processes should be manually resumed.</p>
<b>Migrate process instance to the latest revision.</b>	<b>Translate</b>	<p>Analyze the process instance and roll back some actions (change the process document or the process order). The process engine will continue to run the process instance with the process object metadata and the latest revision number. This is the typical scenario for simple migration, where the logic changes are minimal and the activity indexes are not changed. For example, this may be used when the process scripts are changed but the structure of the process is the same.</p>

<b>Stop process instance and do nothing.</b>	<i>Terminate</i>	<i>None</i> . Used when very wrong process revision that can be handled only manually.
--	------------------	--

## Subflow Process Migration

---

When migrating subflows from version 5.x, each process or revising that has subflow activities is updated to create a list of subflow references. These subflow references have the following format:

| <subflow full name>-<subflow revision number>

In version 5.x, a revision's subflow activity always references a subflow revision. As a result, at migration, you know the subflow revision that this workflow is using. A (top-level) process's subflow activity always references a (top-level) subflow process. At migration, the revision number for the top-level subflow process is always calculated.

Finally, while migrating subflows, all subflow activities are updated to reference the subflow process instead of the revision. At compilation, the process replaces its subflow activities with an actual copy of the subflow workflow.

Once you have migrated your metadata, no backward compatibility issues should exist, as all existing processes and revisions behave as in version 5.x. However, the subflow revision mechanism assumes slightly different behaviour, so revisions created in this release follow different rules.

## Subflow as a Top-level Process

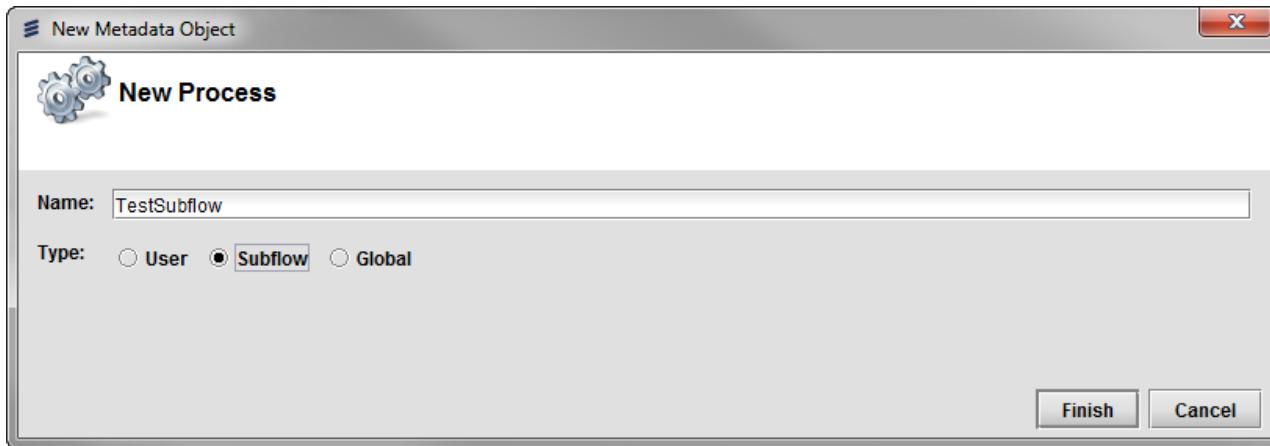
This release supports global subflow processes in addition to local subflows. This feature affects the following areas:

- [Metadata elements and Velocity Studio](#)
- [Runtime process engine](#)
- [Revision mechanism](#)
- [Migration](#)

**Note:** Global subflows are used for backward compatibility only. Creating new global subflows is not recommended. Instead, it is recommended that you either use synchronized subprocesses or local subflows.

### Metadata Elements and Velocity Studio

When [creating a process](#), you have the option of selecting **Subflow** for the process **Type** field, in addition to selecting the **User** or **Global** types.



In this release, there are restrictions imposed on how different process types are used. Only subflow processes can be used as subflows, and only user processes can be used as runnable processes or subprocesses.

The following important points require noting while working with subflow processes in Velocity Studio:

- Process and decision tree subflow activities must be able to use local and global subflows. These activities cannot use user processes as subflows.
- Default Exception Handler, in the Exception metadata element, cannot use subflow processes.

### Runtime Process Engine

Both global and local subflows are compiled into the main process (that is, subflow activity is replaced with the copy of the subflow workflow). As a result, the subflow becomes a part of the main process.

**Note:** A subflow process cannot be started as an independent process neither through process activities, nor through API methods.

### Revision Mechanism

Revisions for global subflows affect these areas:

- [At deployment](#)
- [Manual creation of new revision from Velocity Studio](#)

#### At Deployment

As subflow is considered to be part of the main process, any changes to subflow are counted as changes to the main process.

**Note:** The latest process object always uses latest subflow process. Process revisions always use subflow revisions.

**Example:** Suppose a main process P (in revision 2) uses a global subflow process S (in revision 1). The following table shows four different cases and their results.

Case	Result
<b>Case 1:</b> Both P (revision 2) and S (revision 1) remain unchanged since their last revisions	After deployment, no new revisions are generated, and P (revision 2) continues to use S (revision 1).
<b>Case 2:</b> P (revision 2) is changed while S (revision 1) remains unchanged	After deployment, P is in revision 3 and uses S (revision 1). The new revision object Rev_P (revision 3) is generated and uses revision object Rev_S (revision 1).
<b>Case 3:</b> P (revision 2) remains	After deployment, S is in revision 2 and the new revision object Rev_S (revision 2) is created. Since S is a part of main process P, P is

unchanged while S (revision 1) is changed	also changed and goes into revision 3. The new revision object Rev_P (revision 3) is generated. As a result, P (revision 3) uses S (revision 2), and Rev_P (revision 3) uses Rev_S (revision 2).
<b>Case 4:</b> P (revision 2) and S (revision 1) changed	The results are the same as in case 3.

#### Manual creation of new revision from Velocity Studio

The following points must be kept in mind when manually creating new revisions in Velocity Studio:

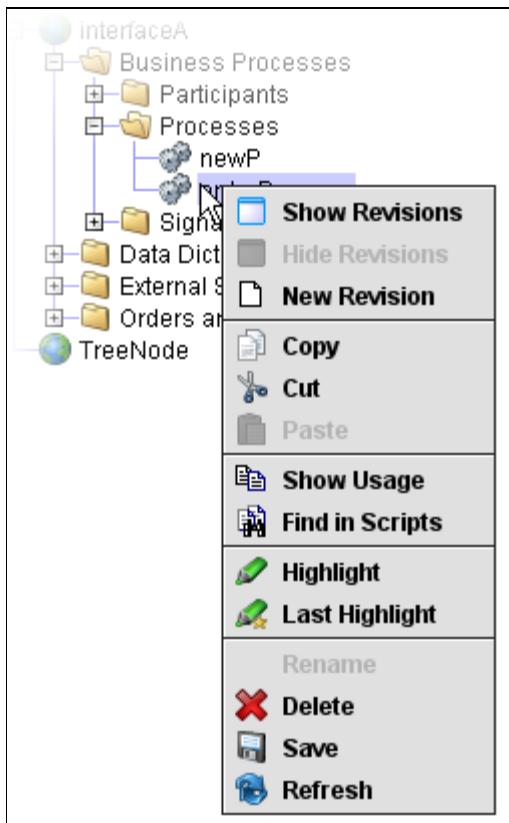
- You cannot manually generate subflow revisions. Creating new subflow revision assumes changes to all processes that use this subflow, which cannot be enforced without user consent.
- A new process revision is generated regardless of its subflow changes.

#### Migration

All version 4.2 subflows migrate into global subflows and references to these subflows are preserved accordingly. See the [Subflow process migration](#) section for details.

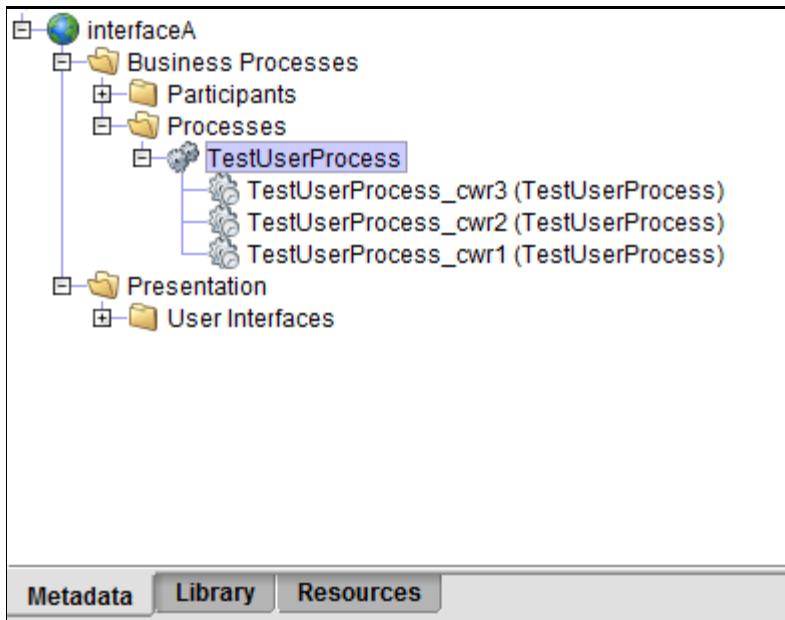
## Process Operations

The right-click pop-up menu for an individual Process metadata has several items in addition to the common right-click menu items.



### Show Revisions

The **Show Revisions** menu item (and then expand using the **expand** icon in Process node) shows all existing [Process Revisions](#) of the Process, under the Process node in the **Navigation** pane.



Selecting a Process Revision node shows its content in **Details** pane. All content of Process Revision is read-only, including the notable **Revision** number.

Equivalent to **Show Revisions** from right-click pop-up menu is double-clicking the Process node in **Navigation** pane.

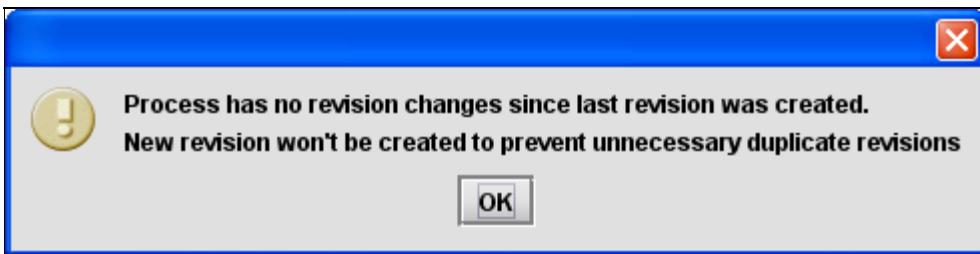
## Hide Revisions

The **Hide Revisions** menu item does the reverse of **Show Revisions**, which hides all Process Revisions under the Process node.

## New Revision

The **New Revision** menu item allows to explicitly create a new [Process Revision](#) of the Process, if the Process metadata has been changed from its latest Process Revision.

A new Process Revision can only be created if the Process metadata has been changed according to the [Process Revision change definition](#). The reason is to minimize the number of Process Revisions in a Process over time. If not, the Process Revision creation is disallowed with the following pop-up prompt:



If [new Process Revision](#) is successfully created, it will show up under the Process (use [Show Revisions](#) if Process Revisions are not shown in Process), with the name of the Process Revision appended with its revision number.

**Note:** This command should be reserved for occasions when there is specific justification to do so. This is to avoid polluting the metadata with unnecessary Process Revisions. For ordinary circumstances, up-to-date Process Revisions of all Processes are created at [Packaging](#) step of deployment, and thus this command is not necessary.

## Delete Revision

The **Delete Revision** menu item appears on the right-click menu item of a Process Revision node (as opposed to Process node), which removes the Process Revision from project directory. The Process Revision will be forever lost in design time as well as any [WAR file](#) that is built afterwards, which implies its runtime will no longer have this metadata to run.

As such, **use this command only when you are sure that it is safe to do so**. Refer to [removing Process Revisions](#) for details. A confirmation prompt appears upon invoking this command:



**Note:** This command is disabled on the latest Process Revision node, because it is a required metadata to compare with Process metadata to decide whether the Process is changed.

# Execution of Process and Process Activities

Global and interface processes are executed by Process Engines. Which Process Engine will execute a particular type of global or interface process (and how many instances), depends on the [Process Engine configuration](#) in the Configuration application.

User processes are typically started through the user interface, but other [starting methods](#) exist. When a request to start a specific process type is submitted, it is stored as an unassigned process in the database with a certain process ID. The process ID determines the user process *bucket*, that defines which Process Engine instance is responsible for its execution. See [Assignment of Workload to Buckets](#) for details.

The CWPRODUCTPROPERTIES table contains the CW-SMLST-PID data value, which specifies the smallest process ID. This value must be larger than 250000.

Once execution is started by a Process Engine, the bucket number is assigned and stored in the process record in the database. This way, there is no need to reassign the running processes when a PE instance is restarted.

Each process contains a [process priority](#) (as defined for this process type), so processes with the highest priority will be addressed first.

Each Process Engine monitors its resource utilization (such as memory), and will not run more processes than its calculated limit. This is done to guarantee adequate performance, and to prevent memory allocation problems.

Any new sub-processes started by the [sub-process \(spawn\) activity](#) of a parent process are always executed by the same Process Engine that executes the parent process.

During process execution, the Process Manager framework stores processes and process-activity status in the process database. This includes process Documents, process flow status, activity-status, activity time constrains, etc. The frequency of these database updates depends on the *speed mode* of the Process Engine (configurable in [System Parameters](#) page in Configuration application). If the Process Engine restarts in whatever reason (because of server or Java VM shutdown), processes will start executing from the point reached prior to Process Engine failure.

## Process Status

---

Every process instance maintains a process status that can have one of the following values:

- *Executing* (1): The process is running (executing).
- *Suspended* (2): The process is suspended. Only user processes can be suspended. Processes can be suspended and restarted through the [System Administration application](#).
- *Completed* (3): The process has completed executing successfully.
- *Aborted* (4): The process is aborted. The process receives an aborted status when the Process Manager framework detects a failure in the Process Engine that executes this process. Aborted processes will return to executing status once restarted.
- *Error* (5): The process is in error. The process receives an error status when an error occurs during process activity execution (such as a JavaScript error or a database error). The Process Engine will stop executing a process that receives an error status. Such processes can be restarted (after the problem is corrected) through the [System Administration application](#).
- *Terminated* (6): The process is terminated. The process receives a terminated status when a terminate request is sent to the process from the [System Administration application](#) or from migration record. The process will stop immediately. Terminated processes cannot be resumed.
- *Cancelled* (7): The process starts a subprocess in synchronous mode, with the subprocess activity having the **synchronization** checkbox selected. The process waits in this activity for the child process to complete. If the child process throws an exception, but does not have a handler to process this exception, it goes into Cancelled status and does not run other activities. The exception is then sent to the parent process.

# Setting Process Priority

---

The Process Management product allows priorities to be assigned to each process type (global or user). Priority values range from 1 to 15, with priority value 1 being the highest priority.

To be able to assign priority to the process, a priority enumeration data type must be defined in the metadata. This data type is based on the *Priority* data type, in the System namespace (that is, *cwf.priority*). The priority enumeration can be customized by overriding *cwf.allPriorities* Data Type (and extending from *cwf.priority*), and contain any number of the priority values (all of them between 1 and 15) and not necessarily all 15. For example, a priority enumeration can contain only three values: 1 - High, 2 - Medium and 3 - Low.

If no process priority enumeration data type is defined, all processes are considered to have an equal priority. If a process priority data type is defined, every process definition must have a priority assigned.

The following sections outline process priority implications:

## Process Activities

When the Process Engine executes process activities, activities with the highest priority are executed first. The Process Engine employs several activity execution threads (10 or more). Each of these threads will pick the highest priority activity awaiting execution. Activities with the same priority are executed based on a FIFO (first in first out) algorithm.

All activities for a specific process instance are executed by the same Process Engine. Prioritization of the process activities is done within each Process Engine separately, and not across all of the running Process Engines.

It is conceivable that processes with a lower priority will receive very little or no execution time. This can be solved by starting additional Process Engines and/or by introducing more powerful computers.

## External Service Participant

When a process needs to invoke an external service Operation, the request is delegated to the interface process that services this Interface. The interface process will perform the request with the highest priority first and use a FIFO (first in first out) algorithm for requests that carry an equal priority.

The interface process is a system process and is not exposed in the metadata. This process is in charge of converting data message objects from their string format to XML objects, so the actual message being sent is a Java XML object and not a simple string.

At the interface/operation level you are given the option to Invoke an interface Directly, thus bypassing this interface process and sending the XML message but as a string of characters.

## Global Process Participant

The priority of a global process does not affect the order in which messages are consumed by the participant thread.

## User Participant

When a process sends a request for a manual task to a User Participant, tasks with higher priority will appear at the top of the user worklist.

## Start User Process

When the user interface submits a request to start a specific user process type, requests for the processes with the highest priority (defined for this process type) will be processed first.

## Starting User Processes

---

The following options can be used to activate a user process:

- *Menu*: A Data Object menu item, with the **Type** of process defined on the [General tab of Menu](#), is created in Velocity Studio and can be used to start the selected process at runtime.
- *Decision Tree*: A Decision Tree is created in Velocity Studio with the required process added to the Decision Tree diagram as a [Sub-Process activity](#). The process will be started when this activity is reached in the Decision Tree.
- *Script*: A process can be started from a script (for example, `startProcess("namespace.processname", pdoc)` in the Process object of the [JavaScript Extensions](#) documentation).

In contrast, a *Global Process* is started automatically by the Process Engine at runtime.

### Process Engine Skips a New Process

As indicated previously, the product has script methods to create new processes, such as `startProcess()` and `startSubProcess()`. A new process in the database is created at the end of the current database transaction. It is recommended that your application not include additional database activities in this transaction, to make it process as quickly as possible.

If you find that the transaction is taking too long, a possible situation occurs in which the process engine skips a new process. To resolve this problem, the process engine reads processes by overlapping the creation time. Use the [New processes read overlap](#) configuration parameter in the Configuration application, under **Processes > General**. By default, its value is 10 seconds. However, you can increase this time when a process engine skips new processes, as required.

You can [restart new user processes from that JMX console](#) that may have been skipped for some reason when starting up the process engine or a standby PE. This feature resets the current reading time and allows the process engine to go back and read skipped processes.

## Assignment of Workload to Buckets

---

The following explains the inner workings of how the workload of a process type are assigned to buckets in bucket span.

### User Process Type

The user processes bucket numbers are predefined as follows:

1. 0 - unassigned processes (new and running waiting for restart).
2. 1-20 - user processes assigned to buckets 1 - 20 respectively.

User processes bucket number  $P$  contains

- all processes already assigned to bucket  $P$ , and
- all unassigned processes that satisfy:  $PID \bmod S = P - 1$ , where  $PID$  is the process ID, and  $S$  is the user processes bucket span, and  $\bmod$  is the modulo operation (that is, remainder of division).

User processes keep record to which bucket they belong. Once assigned to a bucket, they stay in this bucket even if bucket span  $S$  is changed afterwards.

When the user processes bucket span is reduced, all processes belonging to buckets above the new bucket span are "unassigned" - moved to bucket 0. Thus, these processes are distributed to existing buckets according to the modulo arithmetic.

### Global and Interface Process Type

The global processes do not have bucket numbers. Instead, they are assigned to PE ids. The PE\_ID column of the CWGLOBALPROCESS table can have the following values:

1. 0 - unassigned global processes. Any PE can pick it if needed.
2. 1-20 - global processes assigned to PE with PE ids 1 - 20.

Global or interface processes bucket number  $P$  contains

- all global participant messages for the given metadata type that satisfy:  $SID \bmod S = P - 1$ , where  $SID$  is the global participant message serialization ID, and  $S$  is the corresponding type's bucket span.

Global or interface participant messages do not keep the bucket number; their bucket is always calculated by the serialization ID. At runtime, the PE instance only process messages that belong to its buckets.

# Activities Execution

---

An instance of the Process Engine is able to run multiple process instances at the same time. The number of process instances to run is configurable as **Number of processes** in [Config application's PE Processes](#) page. The highest priority activity awaiting execution is picked up first by these threads, but no more than one activity of a specific process instance is executed at any given time. The maximum activity size is 4000 for one process. The system will provide a warning message when the system has approached 90% of the limit.

The Process Engine performs the following tasks during execution of each process activity:

## Alert

- Executes **Before** script for *email* and *pager* Alerts only. The script result is used as the message content.
- Sends the [Alert](#).
- Queues the next sub-activity for execution.

## All

- Executes **Before** script.
- Queues all sub-activities for execution.
- Waits for all sub-activities to end.
- Executes **After** script.

## Case

- Queues sub-activity for execution.

## Choice

- Executes **Before** script.
- Waits until it receives one of the children operations (consume).
- Queues *Operation* activity received for execution.
- Terminates all other pending operations.
- Upon time expiration, process can resume using the *Timeout* activity (if present).
- *Request-response* Operation types are unsupported.
- Validation warning occurs when the Choice activity has two child activities of type Operation that reference exactly the same operation.

## Compensate

- Executes **Before** script.
- Queues a sub-activity for execution.

## Complete

- Executes **Before** script
- Completes the process. All activities that are queued for execution are ignored.
- The process stops with a completed status.

## Exception (Fault)

- Executes **Before** script.
- If **Before** script returns null, finds and queues proper *On Exception* activity for execution (see [exception handling processing](#)).
- If **Before** script returns a number or process object, sends this exception to the specified process (see [exception handling processing](#)).

## Wait (Join)

- Waits for all sub-process(s) of specified type to complete.

- Executes **After** script.
- Queues next sub-activity for execution.

### On Exception

- Executes **Before** script (see [on exception activity execution](#)).
- Queues a sub-activity for execution.

### Operation

- Creates input message (only for *one-way* or *request-response* Operation types).
- Executes **Before** script with input message as a Document parameter (only for *one-way* or *request-response* Operation types).
- Performs Operation (produce, consume or both).
- Executes **After** script with the output message as a Document parameter (only for *request-response* or *notification* Operation types).
- Queues next sub-activity for execution.

### Repeat

- Executes **Before** script.
- Queues *Repeat* activity for execution.

### Resume

- Executes **Before** script.
- Resumes previously suspended activities by the *On Exception activity*.
- Queues next sub-activity for execution.

### Rollback

- Executes **Before** script.
- Queues all appropriate *Compensate* activities for execution.
- Waits for all *Compensate* activities to complete (see [rollback activity execution](#) for details).
- Executes **After** script.
- Queues next sub-activity for execution.

### Script

- Executes **Before** script.
- Queues next sub-activity for execution.

### Sequence

- Executes **Before** script.
- Queues first sub-activity for execution.
- Waits for all sub-activities to end.
- Executes **After** script.

### Signal

- Executes **Before** script.
- If **Before** script returns null, raises Signal within current process.
- If **Before** script returns a number or process object, sends this Signal to the specified process.

### Sub-process (Spawn)

- Creates the new sub-process data Document.
- Executes **Before** script. The Document parameter is the new sub-process Document (script may initialize this Document with proper values).
- Spawns new sub-process with data Document.
- Queues next sub-activity for execution.

### Sub-Flow

- Executes **Before** script.
- Queues first sub-activity for execution.
- Waits for all sub-activities to end.
- Executes **After** script.

## Switch

- Executes **Before** script.
- If exclusive, queues the first *Case* activity with a true condition (or default case if none found) for execution.
- If inclusive, queues all *Case* activities with true condition (or default case if none found) for execution.

## Synchronize

- Waits for a specified Signal to arrive.
- Executes **After** script.
- Queues next sub-activity for execution.

## Timeout

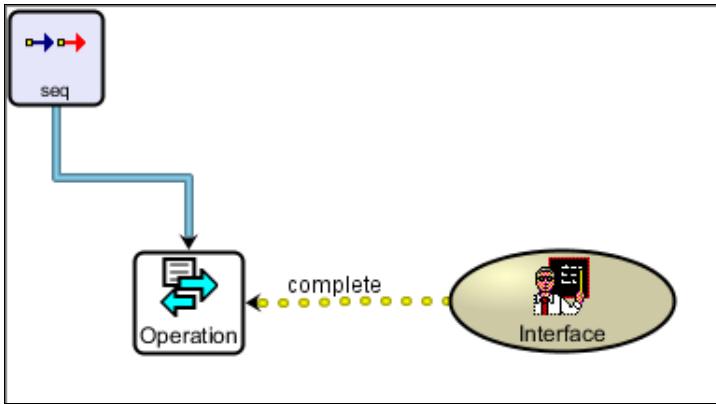
- Applicable only as child of a *Choice* activity
- Waits for duration of *Choice* to expire
- If duration expires, executes **After** script and child activities

## Delay Notification Activities

Operation Activities of the Type = *Notification* allows notifications or messages to be queued before processing. When sending a message, the earliestConsumeTime and a priority can be specified to determine when the message will be processed. The Operation Activity calls the sendMessageToProcess() method to send the message. To make changes to the existing messages, the delayMessageToProcess method is called in the After Script of the Operation Activity.

### Operation Activity

The following business process diagram displays the Operation Activity of the type *Notification*. The Process diagram displays the Activity Operation and Participant associated with the Operation.



To call the sendMessageToProcess method, the Properties of the Operation must contain a Participant and define the Operation of type *Notification* (defined in Business Processes > External Services > Interface > Operation).

Operation activity: 'Start.creditAnalysis.CADecision.reject'

General		Actions		Schedule		Activities	
Name:	reject						
Label:	reject						
Description:							
Participant:	<input type="button" value="orderManagement.creditAnalysisParticipant"/> <input type="button" value="..."/> <input type="button" value=""/>						
Operation:	<input type="button" value="orderManagement.creditAnalysisInterface.rejectCredit - Notification"/> <input type="button" value="..."/>						
Participant instance:							
Milestone:	<input checked="" type="checkbox"/>						
<input type="checkbox"/> Allows rerun <input type="checkbox"/> Invoke directly							
<input type="button" value="Save"/>							

## sendMessageToProcess

A message can be sent through the sendMessageToProcess from any script. To receive the message, you need to configure the Operation activity. This method passes the *earliestConsumeTime* and *priority* optional parameters.

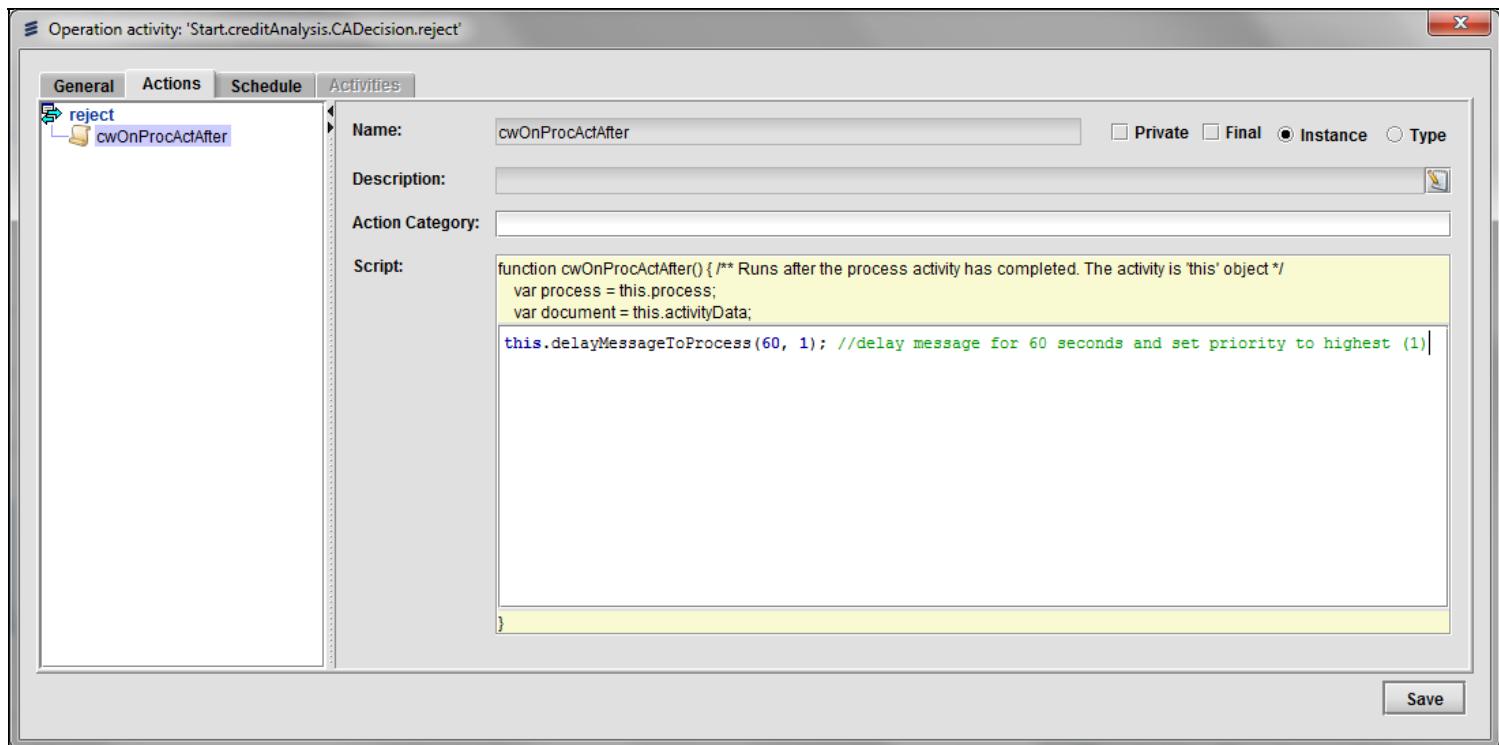
The **earliestConsumeTime** is either a time or date stamp, or an integer value that determines the earliest possible time that the message can be processed. The integer value acts as a delay time in seconds from the current moment; however, it cannot exceed 6 months (190 days). The time or date stamp must be a future date. An exception message occurs if the integer value exceeds 190 days, or the time or date is a past date. If the parameter value is null or zero (0), the system assumes no delay. If the delay exceeds the allowed delay, the process goes into an error state.

The **priority** determines the importance of the message for being processed. A value of 1 has the highest priority. The acceptable values of this parameter range from 1 to 15. The system will assign a value of eight (8) when a zero (0) or null value passed in this parameter. Any other number that is out of range will result in an exception message being created.

**Note:** The process engine does not guarantee that messages sent to a process's Operation activities are consumed in the same order in which they were sent.

## delayMessageToProcess

The delayMessageToProcess() method is called in only the *After* script of the Operation Activity and passes only the *earliestConsumeTime* (time/date stamp or integer value) and *priority* (integer value). This method updates the message record with the new values that are passed and return the message to the queue. In the following example, the message will be processed no earlier than 60 seconds and in the highest priority. If one of these parameters is not passed - then that value will not be updated.



# Worklists and User Tasks at Runtime

A Worklist displays a list of all user tasks that are assigned to a user.

A User Task is an item of work that needs to be acted on by a User Participant. A new user task is created by either a menu item action or a process activity. Once the task is created, it is assigned to a user using the specified [participant distribution method](#) (Note: Manual and shared methods require user interaction). Tasks that are assigned to a user are then available in the user's worklist until a task action is performed, which moves the task to the worklist archive.

## Customizing Worklists

The worklist may be customized in the following ways:

- Custom attributes may be added to user tasks. These changes are done system-wide.
- The worklist table display may be customized (column headers, ordering, width, styles, etc). This customization is on a per-application menu basis.

You can customize attributes by referring to the Worklist documents contained in the **Library tab > cwf\_pm (Process Management Product) > Finders**. The worklist display is customized by creating a user-defined Finder and selecting it in the Worklist Finder property of the User Interface application menu (the built-in Finder is Worklist in the **Process Management Product** namespace).

**Note:** To create and use custom worklist finders, your customized finder user interface must extend `cwf_pm.BaseWorklistFinder.UserInterface`. Otherwise, you cannot use it as a worklist finder.

## Search Behaviour

The current Worklist Finder search behavior is as follows:

When defining the Worklist Finder in the metadata:

- The page size value is set in the metadata to the larger of the **Page size** and **Max row** fields defined on the **Advanced** tab of the Finder properties (refer to section on Finder Advanced Properties).
- At runtime, the Worklist Finder does not display more records than the value of the page size. For example, if the Worklist Finder is defined as **Page size** = 200 and **Max row** = 300 in the metadata, after a search in the User Interface at runtime the Worklist Finder displays up to 300 items on the page, regardless of whether the number of tasks in the database exceeds this number.

When logging into the User Interface at runtime:

- There is no **Next** or **Back** button shown as in other types of Finders; all search results are displayed on one page according to the page size definition.
- Users can create a custom view (refer to section on Finder View Properties) and re-define the **Max row** value. However, the system only takes the value that is less than the page size defined in the metadata. If the new value is larger than the metadata value, the system ignores it and continues to use the old value. For example, using the Worklist Finder in example 1, a custom view is defined as **Max row** = 200, after a search in the User Interface the Worklist Finder displays up to 200 items on the page, regardless of whether the number of tasks in the database exceeds this number. However if a custom view is defined as **Max row** = 400, after a search in the User Interface the Worklist Finder displays up to 300 items on the page, regardless of whether the number of tasks in the database exceeds this number.

The Finder icon in the User Interface shows the number of records that are displayed on the page.

## Notation Unification and Data Storage in Worklist Table

In an effort to unify : and . notation, use only the . notation to ensure that the worklist table properly stores your data. In case the application hardcodes either the participant type or operation for the worklist in the metadata, run the following SQL:

```
UPDATE CWPWORKLIST
SET PARTICIPANT_TYPE = REPLACE(PARTICIPANT_TYPE, ':', '.')
WHERE PARTICIPANT_TYPE LIKE '%:%';

UPDATE CWPWORKLISTARCHIVE
SET PARTICIPANT_TYPE = REPLACE(PARTICIPANT_TYPE, ':', '.')
WHERE PARTICIPANT_TYPE LIKE '%:%';

UPDATE CWPWORKLIST
```

```
SET OPERATION = REPLACE(OPERATION , ':', '.')  
WHERE OPERATION LIKE '%:%';  
  
UPDATE CWPWORKLISTARCHIVE  
SET OPERATION = REPLACE(OPERATION, ':', '.')  
WHERE OPERATION LIKE '%:%';  
  
COMMIT;
```

For more information about worklists and user task, see the following pages:

- [Participants](#)
- [Introduction to Business Processes](#)
- [Configuring Worklist User Interfaces](#)
- [Configuring Worklist Manager](#)
- [Using Worklist Manager](#)
- [User Worklists](#)

## Configuring Worklist User Interface

To display the Worklist at runtime, you must create a User Interface that extends from the *com.conceptwave.system.Application* User Interface (this is used to specify *Worklist*' Finder for the application).

The screenshot shows the 'General' tab of a User Interface configuration dialog. The 'Name' field contains 'Worklist\_Interface'. The 'Label' field contains 'Worklist\_Interface'. The 'Description' field is empty. The 'Extends' field is set to 'com.conceptwave.system.Application'. The 'Help' field is empty. The 'Worklist' dropdown menu is open, showing 'cwf\_pm.BaseWorklistFinder (Worklist)' as the selected item. The 'Timer duration (minutes)' input field is also circled in red.

The Worklist drop-down list displays the system-defined worklist (*cwf\_pm.BaseWorklistFinder (Worklist)*) and any user-defined worklist that were previously created when defining Finders.

The Time duration (minutes) field enables you to specify an interval where the timer calls the *onTimer* method each time an interval elapses. The default interval is specified on the Configuration page >System > Framework parameters > Worklist poll time (minutes). Using the Time duration (minutes) setting overrides the interval set on the Configuration page.

### Worklist Variables

There are system variables that are available to configure the Worklist.

Variable	Description
workListFinder	Type User Interface. Displays the Work List finder if the user has correct permissions.
workListImage	Type String. The image that displays (available/unavailable) for the user who has permissions for the work list.

See [User Interface Variables](#) for a complete list of variables and definitions.

### Worklist Method

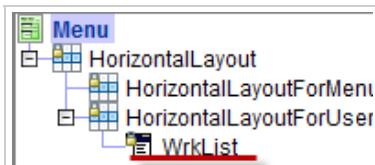
There are system methods that are available to configure the Worklist.

Method	Description
onTimer	The method that is invoked periodically for every <b>Time duration</b> defined in User Interface <b>General</b> tab since the creation of this User Interface, if it is defined. By default, this Method is empty.
hasWorklist	This method checks if the worklist is created for a user based on permissions.
openWorklist	This method opens the worklist when user permissions match permissions for the Worklist.
permWorklist	This method returns the worklist when the permissions set are satisfied.

See [User Interface Methods](#) for a complete list of methods and definitions.

### Menu Form

When you create a User Interface under the Presentation node, by default, several forms are created. The Menu form contains several Form elements that assist you with the layout of your display at runtime.



The WrkList menu contains properties for the configuration of the Worklist menu. The menu creates an icon when clicked opens the WorkList pane which displays the results of the Finder.

createOrder createProcess Admin ▾ Distribution Tasks ▾ WL Alert emailAlert ValidStateForTech Advanced Tech ▾ jsCreateTask

1

WorkList

!	Due	Task	Details	Assigned	Started
	06/17/2010 13:08 13:0...	createTask	custOrder		06/15/2010 13:08 13:0...

For information about configuring the Menu Form element properties, see [Menu](#).

**Important:** Worklists do not display if the user does not have Participant [privileges](#).

## User Worklist

Based on configured processes, the user worklist displays a list of all user tasks that are assigned to the current user. The worklist user interface is instantiated using the **BaseWorklistFinder (Worklist)** found in the **Templates** tab > **cwf\_pm (Process Management Product)** > **Finders** > **Document Finders** folder. If a user has worklist tasks assigned to him or her, the application menu (top right corner) displays the number of worklist tasks and an icon indicating whether user is Available or Unavailable. Clicking the icon or the task number opens the Worklist Finder specified on the Application User Interface element.

createOrder createProcess Admin ▾ Distribution Tasks ▾ WL Alert emailAlert ValidStateForTech Advanced Tech ▾ jsCreateTask							1	
WorkList							1	
!	Due	Task	Details	Assigned	Started			
▶	06/17/2010 13:08 13:0...	createTask	custOrder	06/15/2010 13:08 13:0...				

The columns of the worklist displays specific data about each item.

Column	Description
	The first column allows you to use sort the column in the table. This functionality is similar to the table functionality in <a href="#">Finders</a> .
(priority - sorted first)	Indicates the item with an icon as follows: Alarm Task
Due (sorted second)	Date and time that the task is due to be completed
Task	Task description
Details	Task details
Assigned	Date and time assigned to the current user
Started	Date and time started by the current user

### Displaying Details

You can display the details for a worklist item by doing one of the following actions:

- double-click the action in the worklist row.
- click the arrow icon located at the beginning of the row for the action that you want to view the details.

The worklist user interface has a **Task** drop-down menu that contains actions that correspond to data displayed in the Worklist.

### Working With Tasks

When a process sends a task to a User Participant, these tasks are added to the global worklist queue, but not assigned to any specific user. The User Participant properties have a distribution type attribute which specifies how manual tasks should be assigned to users. This is defined by the **Distribution type** field on the **General** tab.

There are several task settings available in the user interface in a submenu titled **Task**. This submenu contains a list of all *notificationOperations* defined for each participant. Clicking a task action sends a message to the Process Engine and moves the item into the worklist archive (the user no longer sees this item in the worklist). For example, when a user B takes a task from user A, the acknowledge task appears in the user interface for user A to acknowledge this change.

**Note:** Errors are generated when an action cannot be completed.

Action	Description
<b>Start Work</b>	The default the task that displays in the list. When this task action is selected, the current date and time appears in the Started column. The task is not bolded.  <b>Note:</b> The Start Work action is only available if selected work items have not been started.
<b>Get Available</b>	This task displays when the user has the privileges to at least one participant that has shared distribution. This task allows the user

	to retrieve one or more items from the shared task list. If the user does not have the Workgroup Select privilege, the next unassigned task is assigned to the current user and displayed in the worklist (the tasks are chosen by highest priority and earliest due date). If no tasks are available, a warning dialog displays.
<b>Take From User</b>	Displays if the current user has the privilege to at least one participant that has shared distribution. This task allows the user to take tasks assigned to other users.
<b>Delegate Task(s)</b>	Displays if the current user has the privilege to at least one participant that has shared distribution. This task allows the user to give one or more items from the shared task list to another user who has not reached their maximum task capacity.
<b>Acknowledge</b>	This task displays when a task has been taken or delegated to a user.
<b>Assign Tasks</b>	This task is shown if the current user has the Workgroup Select privilege. It opens a dialog that allows the user to change the user assigned to the selected task (allowed only if selected task is for a participant with Shared distribution).
<b>Return to Queue</b>	Returns the task to the participant's queue for the process.
<b>Complete</b>	The task is removed from the task list.

**Note:** All worklist task actions allow for invalid objects.

Tasks displayed in the worklist table are highlighted as follows:

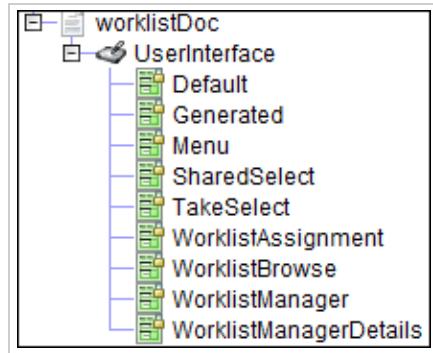
- New tasks: Text is shown in **bold** until the user selects the task.
- Late tasks: Text is shown in **red** after the due date has passed.
- Taken tasks: Text is shown in **red**.

Distribution of tasks are controlled by the configuration of Participants at design time. The Distribution type drop-down field on the Participants general tab contains three choices that allows you to specify how a task is routed. For details, see [User Participant](#).

Work item actions displayed in the **Actions** menu support translation labels.

## Configuring Worklist Manager

At design time, you can configure the Worklist Manager by creating a Document that extends using the `cwf_pm.BaseWorklist (Worklist Document)`.



By default, the `WorklistManagerDetail` Form displays the Worklist Manager fields at runtime when the **Worklist** option is selected in either the Profile Administration at runtime (a user needs to have the correct permissions to be able to view the Profile Administration). By customizing metadata options in Velocity Studio, the Worklist Manager can display in an application menu.

Due	Participant type	User ID	Task	Details	Created	Assigned	Started
-----	------------------	---------	------	---------	---------	----------	---------

Use the existing system-defined document variables and methods to configure the Document Form.

For more information about Worklist Manager runtime options, see [Using Worklist Manager](#).

# Worklist Manager

The Worklist Manager report displays a list of all open worklist tasks that require user action. An administrator having access to Worklist Manager can perform basic changes to worklist tasks, such as view task details, change a task's due date and re-assign task owners.

The screenshot shows the 'Worklist Management' interface with the 'Worklist Manager' tab selected. On the left, there is a search panel with fields for Priority, User ID, Participant, Order ID, Type, Process ID, and dates for Created Date, Assigned Date, Start Date, Due Date. A 'Search' button is at the bottom of the search panel. To the right, a table titled 'Worklist Manager Results (0)' is displayed with columns: Due Date, Participant, User ID, Task, Label, Create Date, Assigned Date, and Start Date. A message 'No items to show.' is centered below the table. At the top right of the interface, there are links for 'Tasks (0)', 'Help', and 'upadmin'.

You can perform the following actions in Worklist Manager:

- **[Access Worklist Manager](#)**

This page describes how to access and use the Worklist Manager to view tasks that require user action.

- **[View task details](#)**

The Task Details page allows you to view specific information about a task, including who is assigned to the task and when the task was created.

- **[Assign a task](#)**

This page describes how to assign a task to a user.

- **[Unassign a task](#)**

This page shows how to unassign a task.

- **[Change a task](#)**

This page allows you to change the priority and due date of a selected task,

- **[View and make changes to an order associated with a task](#)**

This page allows you to view an order associated with a task and make any pertinent changes to the order.

- **[View the change history of a task](#)**

This page shows how you can access the history log, to view changes made to tasks.

## Access Worklist Manager

The Worklist Manager is accessible to users having one of the following privileges in the User Profile Management application:

- Worklist Administrator
- Process Manager Administrator

By default, the Worklist Manager appears when you log in to the Worklist Management application. You can also access the Worklist Manager bar by clicking **Manage > Worklist Manager** and from the User Worklist by clicking **Tasks > Manage**.

To use the Worklist Manager, complete the steps:

1. From the Worklist Manager page, do one of the following:
  - o Specify the following search criteria, and then click the **Search** button to view a list of worklist users:

Field	Description
<b>Priority</b>	Click the drop-down menu and select the task priority from the list.
<b>User ID</b>	Select the user assigned to tasks. This field is a reference field, meaning that you can enter the username in this field or select a user by clicking the <b>Finder</b> icon and selecting a user from the list.
<b>Participant</b>	Click the drop-down menu and select the participant assigned to the tasks.
<b>Order ID</b>	Specify the order associated with the tasks.
<b>Type</b>	Click the drop-down menu, and select whether to display regular tasks or alerts.
<b>Process ID</b>	Specify the process that created the tasks that you want to display.
<b>Created Date</b>	Specify the time and date when task creation occurred. There are two fields that you can specify: <ul style="list-style-type: none"><li>■ <b>From</b> field Click the <b>Calendar</b> icon and select the date from which you want to search. You can also specify the time by using the <b>Hour</b>, <b>Minute</b>, and <b>Second</b> drop-down fields.</li><li>■ <b>To</b> field Click the <b>Calendar</b> icon and select the date to which you want to search. You can also specify the time by using the <b>Hour</b>, <b>Minute</b>, and <b>Second</b> drop-down fields.</li></ul> To search by either a before or after date, specify the corresponding <b>From</b> or <b>To</b> field, and leave the other field blank.
<b>Assigned Date</b>	Specify the time and date when tasks were assigned to a user. There are two fields that you can specify: <ul style="list-style-type: none"><li>■ <b>From</b> field Click the <b>Calendar</b> icon and select the date from which you want to search. You can also specify the time by using the <b>Hour</b>, <b>Minute</b>, and <b>Second</b> drop-down fields.</li><li>■ <b>To</b> field Click the <b>Calendar</b> icon and select the date to which you want to search. You can also specify the time by using the <b>Hour</b>, <b>Minute</b>, and <b>Second</b> drop-down fields.</li></ul> To search by either a before or after date, specify the corresponding <b>From</b> or <b>To</b> field, and leave the other field blank.
<b>Start Date</b>	Specify the date and time when tasks were started. There are two fields that you can specify: <ul style="list-style-type: none"><li>■ <b>From</b> field Click the <b>Calendar</b> icon and select the date from which you want to search. You can also specify the time by using the <b>Hour</b>, <b>Minute</b>, and <b>Second</b> drop-down fields.</li><li>■ <b>To</b> field Click the <b>Calendar</b> icon and select the date to which you want to search. You can also specify the time by using the <b>Hour</b>, <b>Minute</b>, and <b>Second</b> drop-down fields.</li></ul> To search by either a before or after date, specify the corresponding <b>From</b> or <b>To</b> field, and leave the other field blank.
<b>Due Date</b>	Specify the date and time when tasks are expected to be completed. There are two fields that you can specify: <ul style="list-style-type: none"><li>■ <b>From</b> field Click the <b>Calendar</b> icon and select the date from which you want to search. You can also specify the time by using the <b>Hour</b>, <b>Minute</b>, and <b>Second</b> drop-down fields.</li><li>■ <b>To</b> field Click the <b>Calendar</b> icon and select the date to which you want to search. You can also specify the time by using the <b>Hour</b>, <b>Minute</b>, and <b>Second</b> drop-down fields.</li></ul> To search by either a before or after date, specify the corresponding <b>From</b> or <b>To</b> field, and leave the other field blank.

- o Click the **Search** button to view your results.

Create   Manage   View

Worklist Manager

## Worklist Manager

Search

Priority

User ID   

Participant   

Order ID   

Type

Process ID   

Created Date  From  To

Assigned Date  From  To

Start Date  From  To

Due Date  From  To

Worklist Manager Results (0) 

Assign   Unassign   Modify   Order   History

View ▾

	Due Date	Participant	User ID	Task	Label	Create Date	Assigned Date	Start Date
No items to show.								

2. The Worklist Manager search results contain the following fields:

Field	Description												
Icon Column	<p>This field indicates the task type:</p> <table border="1"> <thead> <tr> <th>Icon</th> <th>Task Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td></td> <td>Regular task</td> <td>A (regular) task represents a unit of work, which is a typical task that is created, assigned, and then worked on by users. Users must perform task actions to complete the task. Participant operations specify the task actions and may include such actions as reject, validate, and accept.</td> </tr> <tr> <td></td> <td>Alert</td> <td>An alert notifies you of an event that requires your attention (for example, when a user process that created the task goes into error state, the active task gets closed, and an alert is created). The user must <i>acknowledge</i> the alert to indicate that it was read and can be closed.</td> </tr> <tr> <td></td> <td>Taken task</td> <td>A taken task notifies the user that his task was reassigned to another user. The user must <i>acknowledge</i> the taken task to indicate that it was read and can be closed.</td> </tr> </tbody> </table> <p>The style of the task record indicates whether the task has been started yet (bold) or is overdue (red), or a combination of the two. The task record styles as well as the task icons indicate whether a task is an alert or a regular task. Alerns appear in blue font and taken tasks are greyed out.</p>	Icon	Task Type	Description		Regular task	A (regular) task represents a unit of work, which is a typical task that is created, assigned, and then worked on by users. Users must perform task actions to complete the task. Participant operations specify the task actions and may include such actions as reject, validate, and accept.		Alert	An alert notifies you of an event that requires your attention (for example, when a user process that created the task goes into error state, the active task gets closed, and an alert is created). The user must <i>acknowledge</i> the alert to indicate that it was read and can be closed.		Taken task	A taken task notifies the user that his task was reassigned to another user. The user must <i>acknowledge</i> the taken task to indicate that it was read and can be closed.
Icon	Task Type	Description											
	Regular task	A (regular) task represents a unit of work, which is a typical task that is created, assigned, and then worked on by users. Users must perform task actions to complete the task. Participant operations specify the task actions and may include such actions as reject, validate, and accept.											
	Alert	An alert notifies you of an event that requires your attention (for example, when a user process that created the task goes into error state, the active task gets closed, and an alert is created). The user must <i>acknowledge</i> the alert to indicate that it was read and can be closed.											
	Taken task	A taken task notifies the user that his task was reassigned to another user. The user must <i>acknowledge</i> the taken task to indicate that it was read and can be closed.											
Due Date	This field contains the date and time when the worklist task is due.												
Participant	This field denotes the participant assigned to the task.												
User ID	This field represents the ID of the user assigned to the task.												
Task	Description of the task.												
Label	The metadata label name of the task.												
Create Date	Date and time when the task was created.												
Assigned Date	Date and time when the task was assigned.												
Start Date	Date and time when the task was started.												

For more information about user tasks, see [Worklist Users](#). The Worklist Manager can be configured to display in the Administration application or in an application provided that you have the correct permissions.

From the search results, you can perform the following tasks:

- [View task details](#)
- [Assign a task](#)
- [Unassign a task](#)
- [Change a task](#)
- [View and make changes to an order associated with a task](#)
- [View the change history of a task](#)

## View Task Details

To view task details, do the following:

- From the Worklist Manager search results, click the **Details** icon next to the task that you want to view its details.

The screenshot shows the Worklist Manager interface. At the top, there are tabs for CREATE, MANAGE, and VIEW. A success message 'Assign User(s) completed successfully' is displayed. Below the tabs is a search bar with fields for Priority, Participant, Type, Created Date, and Start Date, each with 'From' and 'To' date pickers. To the right of these fields are buttons for User ID, Order ID, Process ID, Assigned Date, Due Date, and another set of date pickers. Below the search bar is a 'Search' button. The main area contains a grid of task records with columns: Due Date, Participant, User ID, Task, Label, Create Date, Assigned Date, and Start Date. One row in the grid has a magnifying glass icon in the first column, indicating it is selected. The bottom of the screen shows pagination controls for 'Rows per Page' (set to 10), '1 - 5 of 5', and navigation arrows for 'Previous' and 'Next'.

- The Task Details screen appears.

The screenshot shows the Task Details screen. At the top, there are tabs for CREATE, MANAGE, and VIEW. Below the tabs is a 'Task Details' section with various input fields: Priority (Normal), Due Date (03/28/2013 07:52), Order Item ID (0), Participant (tech(Shared)), Task (worklistNS.techInterface/createTask), Assigned Date (03/25/2013 07:52), and Details (empty). To the right of these fields are corresponding output fields: User ID (tt), Sender ID (1548332), Order ID (12655225), Type (Task), Create Date (03/25/2013 07:52), and Start Date (empty). The bottom right of the screen shows navigation links for 'Previous' and 'Next'.

S

The screen contains the following fields:

Field	Description

<b>Priority</b>	This field represents the task's priority.
<b>User ID</b>	This field denotes the ID of the user assigned to the worklist task.
<b>Due Date</b>	This field indicates the date and time when the task is due.
<b>Sender ID</b>	This field contains the ID of the sender process.
<b>Order ID</b>	This field represents the ID of the order.
<b>Order Item ID</b>	This field denotes the ID of the order item.
<b>Participant</b>	This field indicates the participant assigned to the task.
<b>Type</b>	This field denotes the task type.
<b>Task</b>	This field contains the task's description.
<b>Create Date</b>	This field indicates the date and time when the task was created.
<b>Assigned Date</b>	This field denotes the date and time when the task was assigned.
<b>Start Date</b>	This field represents the date and time when the task was started.
<b>Details</b>	This field contains details pertaining to the task.

3. When you have finished viewing the task, click the **Back** button in the right corner to return to the Worklist Manager.

## Assign a Task

To assign a task, complete these steps:

- From the Worklist Manager search results, select the checkbox next to the task that you want to assign to a user, and then click the **Assign** button.

Worklist Manager

Priority	User ID	Participant	User ID	Task	Label	Create Date	Assigned Date	Start Date

No items to show.

Search

Worklist Manager Results (0)

Assign Unassign Modify Order History View ▾

Priority User ID Participant User ID Task Label Create Date Assigned Date Start Date

Participant Order ID Type Process ID Created Date Assigned Date Start Date Due Date

From To From To From To From To From To

Search

- The Worklist Users page appears. Select the **User ID** that you want this task assigned to, and then click the **Select** button.

Worklist Users

User ID	Available	Active	Tasks	Overdue Tasks	Alerts

No items to show.

Search

Worklist User Results (0)

User Profile Tasks Available Unavailable View ▾

No. of Tasks User ID Available Active Tasks Overdue Tasks Alerts

User ID Participant Available Active Tasks Overdue Tasks Alerts

No. of Overdue Tasks Participant Available Active Tasks Overdue Tasks Alerts

No. of Alerts Alerts >= Available Active Tasks Overdue Tasks Alerts

Alerts <= Available Active Tasks Overdue Tasks Alerts

Include Summary Available Active Tasks Overdue Tasks Alerts

Active Available Active Tasks Overdue Tasks Alerts

Search

- The Worklist Manager reappears. A confirmation dialog appears, indicating that selected task has been assigned to the specified user.

Assign User(s) completed successfully

**Worklist Manager**

Priority

User ID



Participant

Order ID



Type

Process ID



Created Date

 From  To 

Assigned Date

 From  To 

Start Date

 From  To 

Due Date

 From  To 

 Assign
 Unassign
 Modify
 Order
 History
 Export ▾
 View ▾

	Due Date	Participant	User ID	Task	Label	Create Date	Assigned Date	Start Date
<input type="checkbox"/>	03/20/2013 14:54:59	partA		worklistNS.techInt	Alert operation No F	03/20/2013 14:54:59	03/20/2013 14:54:59	
<input type="checkbox"/>	03/23/2013 14:52:40	tech(Shared)	tt	worklistNS.techInt	custOrder	03/20/2013 14:52:39	03/20/2013 15:43:21	
<input type="checkbox"/>	03/23/2013 14:52:40	tech(Shared)		worklistNS.techInt	custOrder	03/20/2013 14:52:39		
<input type="checkbox"/>	03/21/2013 14:54:34	partA	wl	WorklistTMPL.intP	wl_100	03/20/2013 14:54:34	03/20/2013 14:54:35	
<input type="checkbox"/>	03/21/2013 14:54:44	partA		WorklistTMPL.intP	wl_10	03/20/2013 14:54:44		

Rows per Page

10



1 - 5 of 5

 Previous
 Next

## Unassign a Task

To unassign a task, complete these steps:

1. From the Worklist Manager search results, select the checkbox next to the task that you want to unassign to a user, and then click the **Unassign** button.

The screenshot shows the Worklist Manager interface. At the top, there are three tabs: CREATE, MANAGE, and VIEW. Below the tabs is a search bar labeled "Worklist Manager" with various filters: Priority, Participant, Type, Created Date, Start Date, User ID, Order ID, Process ID, Assigned Date, Due Date, and a Search button. The main area is a grid of task data. The columns are: Due Date, Participant, User ID, Task, Label, Create Date, Assigned Date, and Start Date. The first row has tasks assigned to "partA". The second row has tasks assigned to "tech(Shared)". The third row has tasks assigned to "tech(Shared)". The fourth row has tasks assigned to "partA". The fifth row has tasks assigned to "partA". The "Assigned Date" column for the second row is highlighted in blue. The "Unassign" button, located in the toolbar above the grid, is highlighted with a red box. The grid also includes icons for Assign, Modify, Order, History, Export, and View. At the bottom, there are buttons for "Rows per Page" (set to 10), "0 - 0 of 0", and navigation links for "Previous | Next".

	Due Date	Participant	User ID	Task	Label	Create Date	Assigned Date	Start Date
<input type="checkbox"/>	03/20/2013 14:54:59	partA		Alert operation No F	wl_202	03/20/2013 14:54:59		
<input checked="" type="checkbox"/>	03/23/2013 14:52:40	tech(Shared)	tt	worklistNS.techInt	custOrder	03/20/2013 14:52:39	03/20/2013 15:43:21	
<input type="checkbox"/>	03/23/2013 14:52:40	tech(Shared)		worklistNS.techInt	custOrder	03/20/2013 14:52:39		
<input type="checkbox"/>	03/21/2013 14:54:34	partA	wl	WorklistTMPL.intP	wl_100	03/20/2013 14:54:34	03/20/2013 14:54:35	
<input type="checkbox"/>	03/21/2013 14:54:44	partA		WorklistTMPL.intP	wl_10	03/20/2013 14:54:44		

2. The Worklist Manager refreshes. A confirmation message appears, indicating that the user is no longer assigned to the specified task.

## Change a Task

You can change the priority and due date of a selected task by completing these steps:

- From the Worklist Manager search results, select the checkbox next to the task that you want to change, and then click the **Modify** button.

The screenshot shows the Worklist Manager interface. At the top, there are three tabs: CREATE, MANAGE, and VIEW. Below the tabs is a search bar labeled "Worklist Manager" with several dropdown and input fields for filtering results. A "Search" button is located below the search bar. The main area displays a grid of task data. The columns include: Due Date, Participant, User ID, Task, Label, Create Date, Assigned Date, and Start Date. One row in the grid has a checked checkbox in the first column. To the right of the grid are buttons for Assign, Unassign, Modify, Order, History, Export, and View. The "Modify" button is highlighted with a red box. At the bottom of the grid, there are buttons for Previous and Next, and a "Rows per Page" dropdown set to 10.

- The Change Task dialog appears, which allows you to change the **Priority** and **Due** date fields.

The screenshot shows the "Change Task" dialog box. It contains two main fields: "Priority" with a dropdown menu showing "Normal" and "Due" with a date and time input field showing "03/23/2013 14:52". At the bottom left is a "Save" button.

The following table describes the fields that are available from the Modify Task dialog:

Field	Description
Priority	Click the drop-down menu and select from one of the priority settings.
Due	Indicate the date and time that the task is due.

Make any pertinent changes, and then click the **Save** button.

3. The Worklist Manager reappears. You can verify that your changes have been made by locating your task from the Worklist Manager and reviewing the task's details.

## View and Change an Order

The Order dialog displays the order associated with the selected task. To access the Order dialog, complete these steps:

1. From the Worklist Manager search results, select the checkbox next to the task that you want, and then click the **Order** button.

The screenshot shows the Worklist Manager interface. At the top, there are tabs for CREATE, MANAGE, and VIEW. Below the tabs is a search bar labeled "Worklist Manager" with various filters: Priority, Participant, Type, Created Date, Start Date, User ID, Order ID, Process ID, Assigned Date, Due Date, and a Search button. The main area is a grid of tasks with columns: Due Date, Participant, User ID, Task, Label, Create Date, Assigned Date, and Start Date. One row in the grid has a checked checkbox in the first column. A red box highlights the "Order" button in the toolbar above the grid. The toolbar also includes Assign, Unassign, Modify, History, Export, and View buttons. At the bottom of the grid, there are pagination controls for Rows per Page (set to 10), a page number (0 - 0 of 0), and navigation links for Previous and Next.

	Due Date	Participant	User ID	Task	Label	Create Date	Assigned Date	Start Date
<input type="checkbox"/>	03/20/2013 14:54:59	partA		Alert operation No F	wl_202	03/20/2013 14:54:59		
<input checked="" type="checkbox"/>	03/23/2013 14:52:40	tech(Shared)	tt	worklistNS.techInt	custOrder	03/20/2013 14:52:39	03/20/2013 15:43:21	
<input type="checkbox"/>	03/23/2013 14:52:40	tech(Shared)		worklistNS.techInt	custOrder	03/20/2013 14:52:39		
<input type="checkbox"/>	03/21/2013 14:54:34	partA	wl	WorklistTMPL.intP	wl_100	03/20/2013 14:54:34	03/20/2013 14:54:35	
<input type="checkbox"/>	03/21/2013 14:54:44	partA		WorklistTMPL.intP	wl_10	03/20/2013 14:54:44		

2. The Order dialog appears, which displays order details that are associated with the selected task.

custOrder

**Customer**

firstName *	lastName
Tech	Smith
address	TN
9051234567	
trigger	
<input type="checkbox"/>	

**Actions:**

Order   Save

3. Clicking the **Order** button allows you to perform the following actions:

- o Select the **Validate** option to determine whether your order is valid. A dialog appears, confirming whether your order is valid. Click the **OK** button to return to the Order dialog.
- o Select the **Save All** option to save all changes that you have made.
- o Select the **Tasks** option to open the Task dialog, which allows you to search and select tasks for the selected order.
- o Select the **Task History** option to open the Worklist History dialog and view a log of changes that have been made to the selected order.

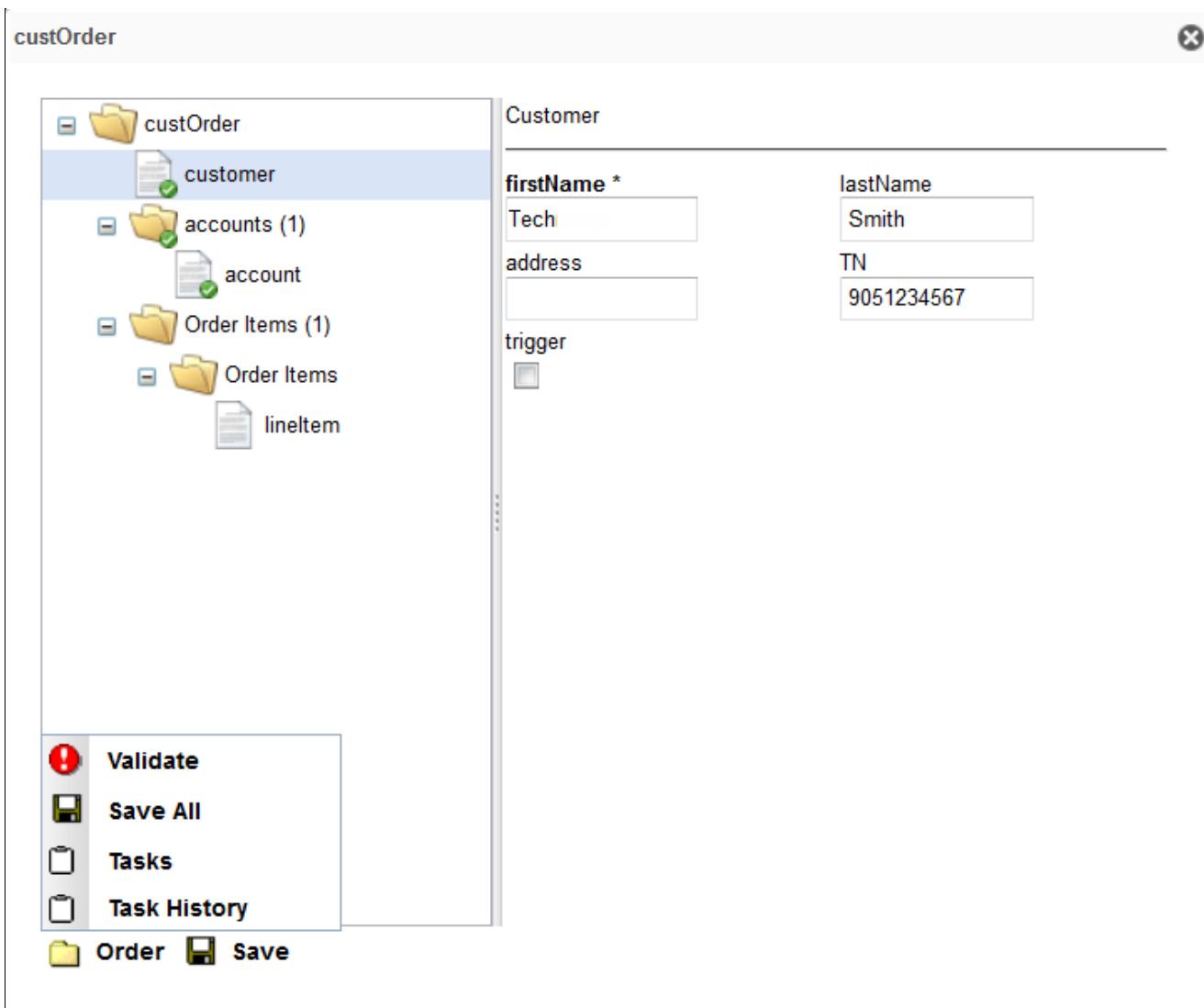
custOrder

**Customer**

firstName *	lastName
Tech	Smith
address	TN
9051234567	
trigger	

**Actions**

-  **Validate**
-  **Save All**
-  **Tasks**
-  **Task History**
-  **Order**     **Save**



4. You can make any pertinent changes to your order. When you have finished making your changes, click the **Save** button.

## View the Change History of a Task

The Task History dialog displays a log of changes that have been made to the selected task. To access the History dialog, complete these steps:

- From the Worklist Manager search results, select the checkbox next to the task that you want, and then click the **History** button.

The screenshot shows the Worklist Manager search interface. At the top, there are tabs for CREATE, MANAGE, and VIEW. Below the tabs is a search form with fields for Priority, Participant, Type, Created Date, Start Date, User ID, Order ID, Process ID, Assigned Date, and Due Date. A 'Search' button is located at the bottom left of the search form. Below the search form is a table of search results. The table has columns: Due Date, Participant, User ID, Task, Label, Create Date, Assigned Date, and Start Date. The 'History' button in the toolbar above the table is highlighted with a red box. The table contains several rows of data, with the second row being highlighted in blue and containing a checked checkbox in the first column.

Due Date	Participant	User ID	Task	Label	Create Date	Assigned Date	Start Date
03/20/2013 14:54:59	partA		Alert operation No F	wl_202	03/20/2013 14:54:59		
03/23/2013 14:52:40	tech(Shared)	tt	worklistNS.techInt	custOrder	03/20/2013 14:52:39	03/20/2013 15:43:21	
03/23/2013 14:52:40	tech(Shared)		worklistNS.techInt	custOrder	03/20/2013 14:52:39		
03/21/2013 14:54:34	partA	wl	WorklistTMPL.intP	wl_100	03/20/2013 14:54:34	03/20/2013 14:54:35	
03/21/2013 14:54:44	partA		WorklistTMPL.intP	wl_10	03/20/2013 14:54:44		

- The History Task dialog appears, which allows you to review all changes made to the task.

The screenshot shows the Task History dialog. At the top, it says 'Task History'. Below that is a table with a single row of data. The table has columns: Action, Performed By, Changed On, User Id, Due Date, Priority, and Reason. The data in the table is as follows:

Action	Performed By	Changed On	User Id	Due Date	Priority	Reason
Created	<System>	03/20/2013 14:52:39		03/23/2013 14:52:40		

At the bottom of the dialog, there are buttons for 'Rows per Page' (set to 10), 'Previous | Next', and 'Export'.

The dialog contains the following fields:

Field	Description
Action	This field contains a brief description of the change (for example, creating the task).

<b>Performed By</b>	This field indicates the name of the user who made changes to the task.
<b>Changed On</b>	This field denotes the date and time when the change was made.
<b>User ID</b>	This field indicates the new task's user ID value.
<b>Due Date</b>	This field contains the task's due date.
<b>Priority</b>	This field represents the task's priority.
<b>Reason</b>	This field contains an explanation for the change.

- When you have finished reviewing the task history, click the **Close** button in the top right corner to close the dialog and return to the Worklist Manager.

## Worklist Users

The Worklist Users report displays a list of active users. Based on configured processes, the Worklist Users report displays a list of all user tasks that are assigned to the current user. The worklist user interface is instantiated using the *BaseWorklistFinder (Worklist)* found in Velocity Studio's Library tab, under **System > cwf\_pm (Process Management Product) > Finders > Document Finders**.

There are two ways to access the Worklist Users report:

- By clicking **Manage > Worklist Users** from the menu bar
- When reassigning or delegating a task to another user either from the Worklist Manager or the User Worklist

When accessed as a standalone finder, the Worklist Users report lists all users in the system. When you invoke Worklist Users from the context of reassigning a task's owner, only those users who have the proper privilege for the particular tasks are listed.

To access the Worklist Users report, complete these steps:

1. From the menu bar, click **Manage > Worklist Users**.
2. The Worklist Users screen appears. Click the **Search** button to view a list of worklist users.

User ID	Available	Active	Tasks	Overdue Tasks	Alerts
upadmin	Available	Active	0	0	0

You can also specify your search criteria in the fields provided and then click the **Search** button to view your results.

The Worklist Users search results contain the following fields:

Field	Description
<b>User ID</b>	This field contains the ID of the user.
<b>Available</b>	This field indicates whether the user is available for task assignment.
<b>Active</b>	This field denotes whether the user ID is active.
<b>Tasks</b>	This field represents the number of tasks assigned to the user.
<b>Overdue</b>	This field contains the number of overdue tasks assigned to the user.
<b>Alerts</b>	This field indicates the number of alerts assigned to the user.

From the Worklist Users search results, you can perform the following tasks:

- [View user summary details](#)
- [View user profile details](#)
- [View a list of tasks assigned to a user](#)

- [Set a user's availability for additional tasks](#)

## View User Summary Details

To view user summary details, do the following:

- From the Worklist Manager search results, click the **Details** icon next to the user that you want to view its details.

The screenshot shows the 'Worklist Users' search interface. On the left, there is a search form with various filters: 'User ID' (set to 'upadmin'), 'Available' (checkbox checked), 'Active' (checkbox checked), and 'Tasks' (set to '0'). On the right, the 'Worklist User Results (1)' table displays one row for 'upadmin'. The table columns are 'User ID', 'Available', 'Active', 'Tasks', 'Overdue Tasks', and 'Alerts'. The 'Available' and 'Active' columns both have green dots, indicating they are checked. The 'Tasks', 'Overdue Tasks', and 'Alerts' columns all show '0'.

User ID	Available	Active	Tasks	Overdue Tasks	Alerts
upadmin	●	●	0	0	0

- The User Summary Details page appears.

The screenshot shows the 'Worklist Users / Details' page. It displays a summary for the user 'upadmin'. The fields shown are: User ID (upadmin), Tasks (0), Overdue Tasks (0), Alerts (0), Available (checked), and Active (checked).

User ID	upadmin	Tasks	0
Overdue Tasks	0	Alerts	0
Available	<input checked="" type="checkbox"/>	Active	<input checked="" type="checkbox"/>

The page contains the following fields:

Field	Description
<b>User ID</b>	This field contains the ID of the user.
<b>Tasks</b>	This field represents the number of tasks assigned to the user.
<b>Overdue Tasks</b>	This field contains the number of overdue tasks assigned to the user.
<b>Alerts</b>	This field indicates the number of alerts assigned to the user.
<b>Available</b>	This field indicates whether the user is available for task assignment.
<b>Active</b>	This field denotes whether the user ID is active.

- When you have finished viewing the task, click the **Back** button in the right corner to return to the Worklist Users page.

## View User Profile Details

To view specific information about a user, follow these steps:

- From the Worklist Users search results, select the checkbox next to the **User ID** that you want, and then click the **User Profile** button.

The screenshot shows the 'Worklist Users' search interface. On the left, there are various search filters: 'No. of Tasks' (two input fields), 'User ID' (input field with icons for edit, search, and delete), 'No. of Overdue Tasks' (two input fields), 'Participant' (dropdown menu), 'No. of Alerts' (two input fields for alerts >= and alerts <=), 'Available' (dropdown menu), 'Include Summary' (checkbox), 'Active' (dropdown menu), and a 'Search' button. On the right, the results are displayed under 'Worklist User Results (1)'. A red box highlights the 'User Profile' button. Below it, a table shows one result: 'User ID' (upadmin), 'Available' (green dot), 'Active' (green dot), 'Tasks' (0), 'Overdue Tasks' (0), and 'Alerts' (0). At the top of the results table, there are tabs for 'User Profile' (which is selected), 'Tasks' (with 'Available' and 'Unavailable' sub-options), 'Export' (dropdown), and 'View' (dropdown).

- The User Summary Details page appears.

The screenshot shows the 'User Profile Details' page. At the top, there are navigation links: 'CREATE', 'MANAGE', and 'VIEW'. The main area contains fields for 'User ID' (manager), 'Active\*' (checkbox checked), 'Minimum Effort' (input field 3), 'Email' (input field), 'Name' (input field manager), 'Available\*' (checkbox checked), 'Maximum Effort' (input field 10), and 'Org Chart Position' (input field with icons for edit, search, and delete). At the top right, there are 'Previous' and 'Next' buttons, and at the bottom right, there is a 'Back' button.

The User Profile Details page contains the following fields:

Field	Description
<b>User ID</b>	This field denotes the ID of the selected worklist user.
<b>Name</b>	This field represents the name of the selected user.
<b>Active</b>	This field indicates whether the selected user is active or not.
<b>Available</b>	This field indicates whether the selected user is available or not.
<b>Minimum Efforts</b>	This field contains the smallest number of tasks that the selected user can have assigned.
<b>Maximum Efforts</b>	This field denotes the largest number of tasks that the selected user can have assigned.
<b>E-mail</b>	This field represents the e-mail address of the selected user.
<b>Org Chart Position</b>	This field indicates the organizational chart to which the selected user belongs.

- When you have finished viewing the user profile details, click the **Back** button in the right corner to return to the Worklist Users page.

## View a List of Tasks Assigned to a User

To view a list of tasks assigned to a user, follow these steps:

- From the Worklist Users search results, select the checkbox next to the **User ID** that you want, and then click the **Tasks** button.

The screenshot shows the 'Worklist Users' search interface. On the left, there is a search panel with fields for 'No. of Tasks', 'User ID', 'No. of Overdue Tasks', 'Participant', 'No. of Alerts', 'Available', 'Include Summary', and 'Active'. On the right, the 'Worklist User Results (0)' section displays a table header with columns: Available, Active, Tasks, Overdue Tasks, and Alerts. Below the header, it says 'No items to show.'

- The User Tasks dialog appears, showing a list of tasks that are assigned to the selected user.

The screenshot shows the 'User Tasks' dialog. It contains a grid with one row of data. The columns are: Task ID (checkbox), Task Name (checkbox, magnifying glass, document icon), Due Date (03/21/2013 14:54:34), Task Description (WorklistTMPL.intPartA/createTask1PartA), Details (wl\_100), Assigned Date (03/20/2013 14:54:35), and Start Date (empty). The grid has standard export and view buttons at the top right. At the bottom, there are buttons for 'Rows per Page' (set to 10), 'Previous | Next', and a 'Close' button.

The User Tasks contains the following fields:

Field	Description
Due Date	This field denotes the date and time when the worklist task is due.
Task	This field contains the description of the task.
Details	This field indicates the order details for the task.
Assigned Date	This field represents the date and time when the task was assigned.
Started Date	This field denotes the date and time when the task was started.

- When you have finished viewing the list of tasks assigned to the selected user, click the **Close** button in the right corner to return to the Worklist

Users page.

## Set a User's Availability for Additional Tasks

To set a user's availability for additional worklist tasks, complete these steps:

- From the Worklist Users search results, select the checkbox next to the **User ID** that is set to **Available**, and then click the **Unavailable** button.

Screenshot of the Worklist Users search interface. The search filters are set to find users with 0 tasks, no alerts, and the 'Available' status is selected. The 'Unavailable' button is highlighted with a red box.

User ID	Available	Active	Tasks	Overdue Tasks	Alerts
upadmin	<input checked="" type="radio"/>	<input checked="" type="radio"/>	0	0	0

- A confirmation message appears on the page, indicating that the selected user is unavailable for task assignment.

Screenshot of the Worklist Users search interface after setting the user 'upadmin' to 'Unavailable'. A green message box at the top right says 'Update User(s) completed successfully'. The 'Available' radio button for 'upadmin' is now highlighted with a red box.

User ID	Available	Active	Tasks	Overdue Tasks	Alerts
upadmin	<input type="radio"/>	<input checked="" type="radio"/>	0	0	0

- Similarly, to set a user as available for additional tasks, select the checkbox next to the **User ID** that is set to **Unavailable**, and then click the **Available** button.

## Worklist Participants

The Worklist Participants report displays a list of participants with pending tasks and alerts. A user participant represents a person who performs manual tasks within a process.

To access the Worklist Participants report, complete these steps:

1. From the menu bar, click **Manage > Worklist Participants**.
2. The Worklist Participants screen appears, which allows you to view a list of participants with pending tasks.

Participant	Alerts	Assigned Tasks	Unassigned Tasks	Overdue Tasks	Total
partA	1	1	2	0	2
tech(Shared)	0	0	2	0	2

The Worklist Participants report contains the following fields:

Field	Description
<b>Participant</b>	This field contains the participant name.
<b>Alerts</b>	This field denotes the number of alerts assigned to the participant.
<b>Assigned Tasks</b>	This field represents the number of tasks assigned to the participant.
<b>Unassigned Tasks</b>	This field indicates the number of tasks assigned to the participant, but not assigned to any users.
<b>Overdue Tasks</b>	This field contains the number of overdue tasks assigned to the participant.
<b>Total</b>	This field denotes the total number of tasks for each participant.

From the Worklist Participants page, you can perform the following tasks:

- [View a list of users associated with a participant](#)
- [View a list of tasks associated with a participant](#)

### View a List of Users Associated with a Participant

To view a list of users that are associated with a participant, follow these steps:

1. From the Worklist Participants page, select a **Participant** and then click the **Users** button.

CREATE   MANAGE   VIEW

### Worklist Participants

Participant	Alerts	Assigned Tasks	Unassigned Tasks	Overdue Tasks	Total
partA	1	1	2	0	2
tech(Shared)	0	0	2	0	2

Rows per Page: 10 | 1 - 2 of 2 | Previous | Next

2. The Worklist Users dialog appears, showing a list of users that are associated with the selected participant.

### Worklist Users

**Worklist Users**

No. of Tasks	<input type="text"/>	<input type="text"/>	User ID	<input type="text"/>					
No. of Overdue Tasks	<input type="text"/>	<input type="text"/>	Participant	<input type="text"/> tech(Shared)					
No. of Alerts	<input type="text"/>	<input type="text"/>	Available	<input type="text"/>					
Include Summary	<input type="checkbox"/>		Active	<input type="text"/>					

Search

<input type="checkbox"/>	User ID	Available	Active	Tasks	Overdue Tasks	Alerts
<input type="checkbox"/>	tt			0	0	0
<input type="checkbox"/>	t			0	0	0

Rows per Page: 10 | 1 - 2 of 2 | Previous | Next

### View a List of Tasks Associated with a Participant

To view a list of tasks that are associated with a participant, follow these steps:

- From the Worklist Participants page, select a **Participant** and then click the **Tasks** button.

Worklist Participants					
Participant	Alerts	Assigned Tasks	Unassigned Tasks	Overdue Tasks	Total
partA	1	1	2	0	2
tech(Shared)	0	0	2	0	2

Rows per Page  1 - 2 of 2 Previous | Next

2. The Worklist Manager dialog appears, showing a list of tasks that are associated with the selected participant.

Worklist Manager																							
Worklist Manager		User ID			Order ID			Process ID															
Priority		<input type="text"/>			<input type="text"/>			<input type="text"/>															
Participant		<input type="text" value="partA"/>			<input type="text"/>			<input type="text"/>															
Type		<input type="text"/>			<input type="text"/>			<input type="text"/>															
Created Date		<input type="text"/> From		<input type="button" value="Calendar"/>	<input type="text"/> To		<input type="button" value="Calendar"/>	<input type="text"/> Assigned Date		<input type="button" value="Calendar"/>													
Start Date		<input type="text"/> From		<input type="button" value="Calendar"/>	<input type="text"/> To		<input type="button" value="Calendar"/>	<input type="text"/> Due Date		<input type="button" value="Calendar"/>													
<input type="button" value="Search"/>																							
<input type="button" value="Assign"/> <input type="button" value="Unassign"/> <input type="button" value="Modify"/> <input type="button" value="Order"/> <input type="button" value="History"/> <input type="button" value="Export"/> <input type="button" value="View"/>																							
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Due Date <input type="text"/> Participant <input type="text"/> User ID <input type="text"/> Task <input type="text"/> Label <input type="text"/> Create Date <input type="text"/> Assigned Date <input type="text"/> Start Date																							
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	03/20/2013 14:54:59	partA		Alert operation No / wl_202	03/20/2013 14:54:59																
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	03/21/2013 14:54:34	partA	wl	WorklistTMPL.intf wl_100	03/20/2013 14:54:34	03/20/2013 14:54:35															
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	03/21/2013 14:54:44	partA		WorklistTMPL.intf wl_10	03/20/2013 14:54:44																

Rows per Page  1 - 3 of 3 Previous | Next

# Order Printing

You may use the product UI to capture an order at its current state into a PDF document. This process allows an order to be saved or printed for future use.

## Overview

---

There are two elements that take part in PDF generation in the product UI:

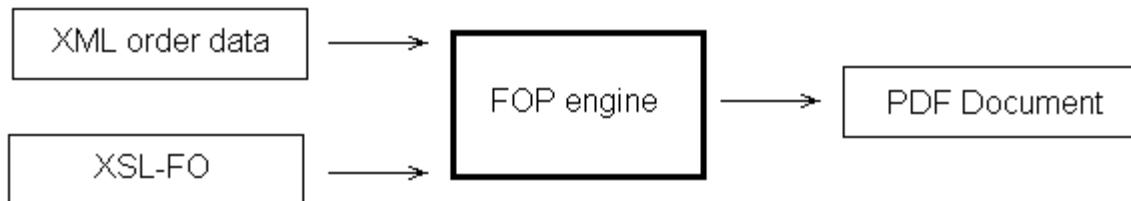
- XML data, which comes from the current order.
- XSL-FO library, which is user-created and comes from the system.

Each order using the order printing feature requires a unique XSL-FO library.

## FOP (Formatting Objects Processor)

---

The engine compiling the PDF document is called an FOP (Formatting Objects Processor). The FOP uses the XSL-FO library and XML order data as input parameters.



For detailed information about FOP, refer to the Apache XML Graphics project documentation (<http://xmlgraphics.apache.org/fop>).

# Configuring PDF Generation

---

For every order using this feature, there are four steps:

1. Generate an XML view of the order using the product UI.
2. Use the XML view to generate an XSD (schema definition).
3. Use the XSD to design an XSL-FO library.
4. Define the XSL-FO as a CW system resource.

There are several XML-designing and style-sheet designing tools available to create the XSD and XSL-FO. However, for the remainder of this document, we will refer to Altova XMLSpy and Altova StyleVision (<http://www.altova.com/>).

**Note:** Currently, the XSLT 1.0 standard is supported.

## Capturing XML Order Data

The XML order view generated from CW UI is the base of this operation. The easiest method to capture this raw XML data for an order is to create a `<script>` menu in the UI for that order:

1. In Velocity Studio, create a menu of category `<object menu>` with the order as the object.
2. Create a sub-menu of type `<script>` with the following script:

```
var currentOrder = menuOwner;
var xmlData = currentOrder ? currentOrder.toXML() : null;
if (xmlData) {
    Global._showModelessDialog(xmlData, 60, 30, 'text/xml');
```

See Velocity Studio User Guide's section on Menus for more information. Activate the metadata and use the UI to save the XML order data.

## Creating XSD Schema

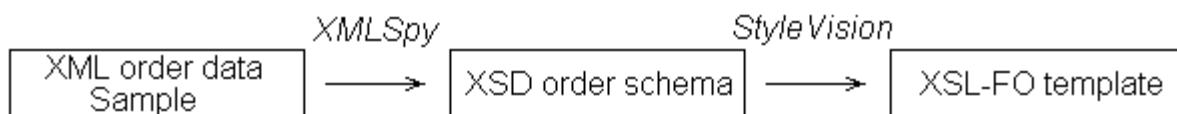
The XML data saved in the previous step will provide a basis for the XSD. The XSD is a skeleton document which gives the set of rules which validate an XML document. XMLSpy has a feature to auto-generate the XSD for any XML document.

1. Open XMLSpy and load the XML order data.
2. Choose DTD/Schema > Generate DTD/Schema, and agree with the default options.
3. The XSD will auto-generate; save it to disk.
4. Select yes when asked to apply the XSD to the working XML file, save it as well.

## Designing XSL-FO library

The XSD generated in the previous step will provide the basis for our XSL-FO library. The XSL-FO library will extract and apply formatting rules to the XML order data. When applying these rules, however, the XSL-FO relies on the general skeletal structure of the XML document, as defined by the XSD.

For example, an order can list 2 *Node* instances while another order lists 5. Rather than basing the XSL-FO on a hard-coded XML indicating either of 2 or 5 instances, we base it on the less rigid XSD. This allows us to handle all instances of *Node* the same way, no matter the number.



1. Launch StyleVision and open the XSD.
2. *Optional:* To preview the PDF in StyleVision, an XML data file and FOP must be assigned. FOP can be downloaded from any of the mirrors found at <http://www.apache.org/dyn/closer.cgi/xmlgraphics/fop>.
  - o Select **File > Assign working XML file** and load the XML order data.
  - o Select **Tools > Options > XSL-FO > FO Processor** and browse to fop.bat.
3. Design the style sheet. (For full details please refer to the Altova documentation).
  - o Drag nodes from the schema tree pane (left) into the content pane (right).
  - o Upon drop you will be asked what type of field you want to create, input field will suffice, as it will be treated as static text when the PDF is compiled.
  - o In the right pane, you can organize fields into tables, add static text labels, and insert images. There are also options for check boxes, drop down lists and expressions.
  - o *Images:* The CW framework will require images to be loaded as resources, and not fixed references. For now when inserting images, enable the *Absolute Path* option and in a later step, we will manually indicate the framework URL.
  - o If you have performed the FOP configuration step, you can preview the output using the PDF Preview tab.
4. Export the XSL-FO library: Select **File > Save Generated Files > Save Generated XSL-FO File**.

After creating the XSL-FO file, a **manual edit** must be made to the document header to ensure full browser and Web server compatibility. Change:

```
<xsl:output version="1.0" method="html" encoding="UTF-8" indent="no" />
```

to:

```
<xsl:output version="1.0" encoding="UTF-8" indent="no" omit-xml-declaration="no" media-type="text/html" />
```

where:

<b>Version</b>	specifies the version of the output method
<b>Encoding</b>	specifies the preferred character encoding that the XSLT processor should use to encode sequences of characters as sequences of bytes
<b>Indent (boolean)</b>	specifies whether the XSLT processor may add additional whitespace when outputting the result tree
<b>Omit-xml-declaration (boolean)</b>	specifies whether the XSLT processor should output an XML declaration
<b>Media-type</b>	specifies the media type (MIME content type) of the data that results from outputting

The XSL-FO file is now ready for PDF generation.

## Special Handling for Images

**Note:** For displaying images inside the PDF file, the the **HTTP port** attribute must match the Application Server port (for example, Tomcat 8080 or WebLogic 7001). This attribute is set in the Configuration application, under **System > System Parameters**.

In the previous step, we indicated images with an absolute path, but this will not suffice for the runtime environment. Thus, we manually edit the XSL-FO to get images as resources from the CW framework.

Look for all instances of the `<fo:external-graphic>` tag, and change the image references to follow this format:

```
cwf:///<subfolder>/filename.ext
```

For example:

```
<fo:external-graphic>
  <xsl:attribute name="src">url('<xsl:text disable-output-escaping="yes">file:///C:\fop_demo\banner.gif</xsl:text>')
  </xsl:attribute>
</fo:external-graphic>
```

Should be changed to:

```
<fo:external-graphic>
  <xsl:attribute name="src">url('<xsl:text disable-output-
escaping="yes">cwf:///docdemo/banner.gif</xsl:text>')
  </xsl:attribute>
</fo:external-graphic>
```

Where:

- **banner.gif** is the image defined as a resource (that is, in the Resources tab of Velocity Studio)
- **cwf://** is the CW protocol for resolving images
- **/docdemo** is the /<subfolder>, if such a subfolder has been defined

**Note:** **cwf** is the default name for the path. It can be renamed during system deployment. So check with the system administrator to get the WAR file name in your configuration.

## XSL-FO Font Customization

Custom fonts are described in an XML file named *userconfig.xml* which is loaded as an /FOP/ resource in Velocity Studio. This file is a master font configuration for all applications in an environment. That is, only one *userconfig.xml* file is required for all applications in that environment.

**Important:** The font and font-metrics files (mentioned below) must reside on the Web server machine. On the Windows platform, all fonts must go in the *\Windows\fonts* folder.

In the case of multiple Web servers, the font-metrics file must be in the same file path as mentioned in *userconfig.xml*. For example, if *userconfig.xml* points to *D:\proj\fonts\FONTXML\HelveticaLT53Extended.xml*, this file must exist in this fixed path on all Web server machines.

**STEP 1:** The file *userconfig.xml* uses the containment structure shown in the figure below. The table below describes the function of each XML tag.

```
<! --<!DOCTYPE configuration SYSTEM "config.dtd"\>-->
<configuration>
  <font>
    <font-metrics-file=d:\proj\fonts\FONTXML\HelveticaLT53Extended.xml" embed-
file="C:\WINDOWS\Fonts\lte50699.ttf" kerning="yes">
      <font-triplet name="HLT53Ext" style="normal" weight="normal"/>
      <font-triplet name="HLT53Ext" style="normal" weight="bold"/>
    </font>
  </font>
</configuration>
```

XML Tag	Description	Parameters
<b>configuration</b>	Bounds the font configuration file.	None
<b>fonts</b>	Bounds all fonts.	None
<b>font</b>	One tag required for each font group.	<b>metrics-file</b> - points to XML file defined by FOP. See FOP spec for more details (URL provided below) <b>embed-file</b> - points to location of .TTF font file. <b>kerning</b> - enables font kerning (even spacing).
<b>font-triplet</b>	One tag required for each font variant. See FOP spec for more details (URL provided below).	<b>name</b> - name by which font will be referenced in XSL-FO stylesheet. <b>style</b> - font style ( <i>normal</i> , <i>italic</i> , <i>oblique</i> ). See CSS or HTML spec for more details. <b>weight</b> - font weight ( <i>normal</i> , <i>bold</i> , <i>bolder</i> , <i>lighter</i> ). See CSS or HTML spec for details.

**Note:** More information on the font-metrics file can be found at <http://xmlgraphics.apache.org/fop/faq.html>.

## Change the Configuration File and the XSL File

Since the FOP library was updated, changes must also be performed in the configuration and XSL file.

## Configuration File

In the configuration file, you can either install your .ttf file under the windows/fonts folder or an arbitrary folder on your computer, and then specify the location of the .ttf file with directory tag in userconfig.xml, as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<fop version="1.0">
  <renderers>
    <renderer mime="application/pdf">
      <fonts>
        <directory>C:\MyFonts1</directory>
      </fonts>
    </renderer>
  </renderers>
</fop>
```

## XSL File

In the XSL file, the Font-family attribute must be defined for the content that contains foreign characters that cannot be displayed properly. Additionally, the value of that attribute must be the font name that appears after having installed it under windows/fonts.

**Example:** With a font file called TIMES.ttf, after it is installed in our operating system, the name of that font is Times New Roman (TrueType).

In the XSL file, the XML is as follows:

```
<fo:inline font-family="Times New Roman" font-size="10pt" font-style="normal" font-weight="normal">
  <xsl:text>SaygĂ±larĂ±mĂ±zla.</xsl:text>
</fo:inline>
```

Then, see the steps in the configuration file.

**STEP 2:** Once the file *userconfig.xml* has been defined, it must be loaded as a resource in Velocity Studio.

**STEP 3:** The XSL-FO references the custom fonts using the nameparameter from the font-triplettag in *userconfig.xml*. An example follows below. The figure below shows the end result of the custom font after PDF compilation.

```
<fo:inline font-family="HLT53Ext">This is HelveticaNeueLT53Ex font 11 </fo:inline>
<fo:block>
  <fo:leader leader-pattern="space" />
</fo:block>
```

# This is HelveticaNeueLT53Ex font 11

## Loading XSL-FO

To load the XSL-FO, complete these steps:

1. For internalization, rename the XSL to order-*en*-xx.xsl.

**Note:** For multiple language support, replace *en* with the appropriate ISO 639 language code (for example, order-it-xx.xsl for Italian).

2. To add a new language resource, copy the XSL file to your <root\_metadata\_directory>\languages folder. You can then view your file in the **Resources** folder of Velocity Studio.

## Notes:

- If the resource name or its path is incorrect, it will not be found at runtime.
- When a language resource is not found with either an expected or appropriate language code suffix, a default file is used.
- You can use the `order.generatePDF()` method to call your Order object. The following is an example:

```
Order.generatePDF("/XSL-DO/gpipPq.xslt", xmlData);
```

**Note:** Order is a data order object.

- Depending on the current language (for example, fr-ca, de-xx, etc.), the PDF is automatically generated using the existing file (that is, gpipPq-fr-ca.xslt, gpipPq-de-xx.xslt, etc.), respectively, and gpipPq.xslt by default.
- There is also multilanguage support for `Global.getResource()` and `Global.generatePdfFile()` methods.

## Loading Images in Velocity Studio

You can load images in Velocity Studio by copying your image files to your `<root_metadata_directory>\languages` folder. You can then view your file in the **Resources** folder of Velocity Studio. Refer to the Resources pane section of the Velocity Studio User Guide for more information.

### Specifying Path for Block Background Images

You can specify both absolute and relative paths for a block background image in your .fo file.

The following is an example of using an absolute path to your block background image:

```
<fo:block border="0.5pt solid silver"
  background-image="url('cwf:///images/block_logo_128x26.gif')">
  Absolute path to image works
</fo:block>
```

Similarly, you can specify a relative path to your block background image:

```
<fo:block border="0.5pt solid silver"
  background-image="url('images/block_logo_128x26.gif')">
  Relative path to image works
</fo:block>
```

## Support for Relative Font Path for userconfig.xml

You can use relative font paths in your `userconfig.xml` file. The font paths are relative to your metadata's resources folder.

The following is an example of using absolute paths in your `userconfig.xml` file:

```
<font metrics-url="file:///C:/ordering/resources/fonts/msgothic.xml" kerning="yes" embed-
url="file:///C:/ordering/resources/fonts/msgothic.ttf">
  <font-triplet name="MS Gothic" style="normal" weight="normal" />
</font>
```

You can use relative paths as follows:

```
<font metrics-url="file:///fonts/msgothic.xml" kerning="yes" embed-url="file:///fonts/msgothic.ttf">
  <font-triplet name="MS Gothic" style="normal" weight="normal" />
</font>
```

## Memory Considerations

To successfully generate your PDF, it is recommended that you increase the memory size on your application server. The following is an example of increasing the amount of memory required on a Tomcat server to successfully generate a PDF:

```
CATALINA_OPTS=-Xms1024m -Xmx1024m -XX:+UseParNewGC -XX:+CMSPermGenSweepingEnabled -  
Djava.awt.headless=true
```

**Notes:**

- It is recommended that you add the `-Djava.awt.headless=true` JVM option manually to the process engine script. This option allows you to run the process engine in headless mode to perform Abstract Windowing Toolkit (AWT) operations.
- The default [log level](#) for FOP/XMLGraphic is ERROR. Changing this log level may render PDF generation slower than expected.

## Test Cases

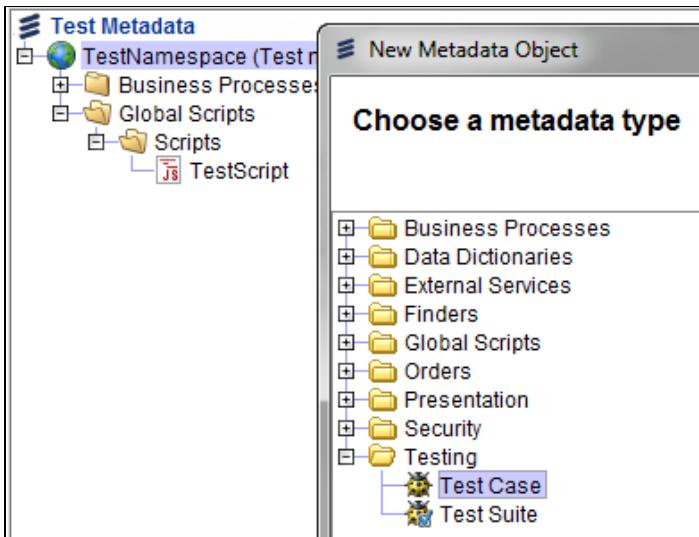
**Test Cases** facilitate unit-test development to the metadata. A test case provides the ability to:

- Store unit-tests functions as JavaScripts
- Specify unit-test *setup* and *teardown* activities using JavaScript or SQL, or both
- Specify expected duration of the test case processing

### Create a Test Case

To create a Test Case, complete these steps:

1. Right-click your namespace and select **New** from the menu.
2. From the **Testing** folder, select **Test Cases** and then click the **Next** button.



3. A new Test Case form displaying the test case properties appears.

### Test Case Properties

The following figure shows the general properties of a Test Case.

Name:	testEquipment	<input type="checkbox"/> Private	<input type="checkbox"/> Restricted	<input type="checkbox"/> Deprecated
Label:	Test Equipment	...		
Description:	This unit test tests the correctness of equipment related functionalities.			
Duration:	<input type="text"/>	Hour(s) <input type="text"/>	Minute(s) <input type="text"/>	Second(s) <input type="text"/>
Setup SQL:	<input type="button" value="..."/>			
Setup Javascript:	Global.logDebug("Test Equipment TC Setup Starts"); <input type="button" value="..."/>			
Functions:	<div style="background-color: #ffffcc; padding: 5px;">         Function Setup SQL          Function Setup Javascript  <b>TestFindEquipment</b> </div> <div style="background-color: #ffffcc; padding: 5px; margin-top: 10px;">         Function Teardown Javascript          Function Teardown SQL       </div>			
	<input type="button" value="Add"/> <input type="button" value="Remove"/> <input type="button" value="Properties"/>			
Teardown Javascript:	Global.logDebug("Test Equipment TC Teardown"); <input type="button" value="..."/>			
Teardown SQL:	<input type="button" value="..."/>			

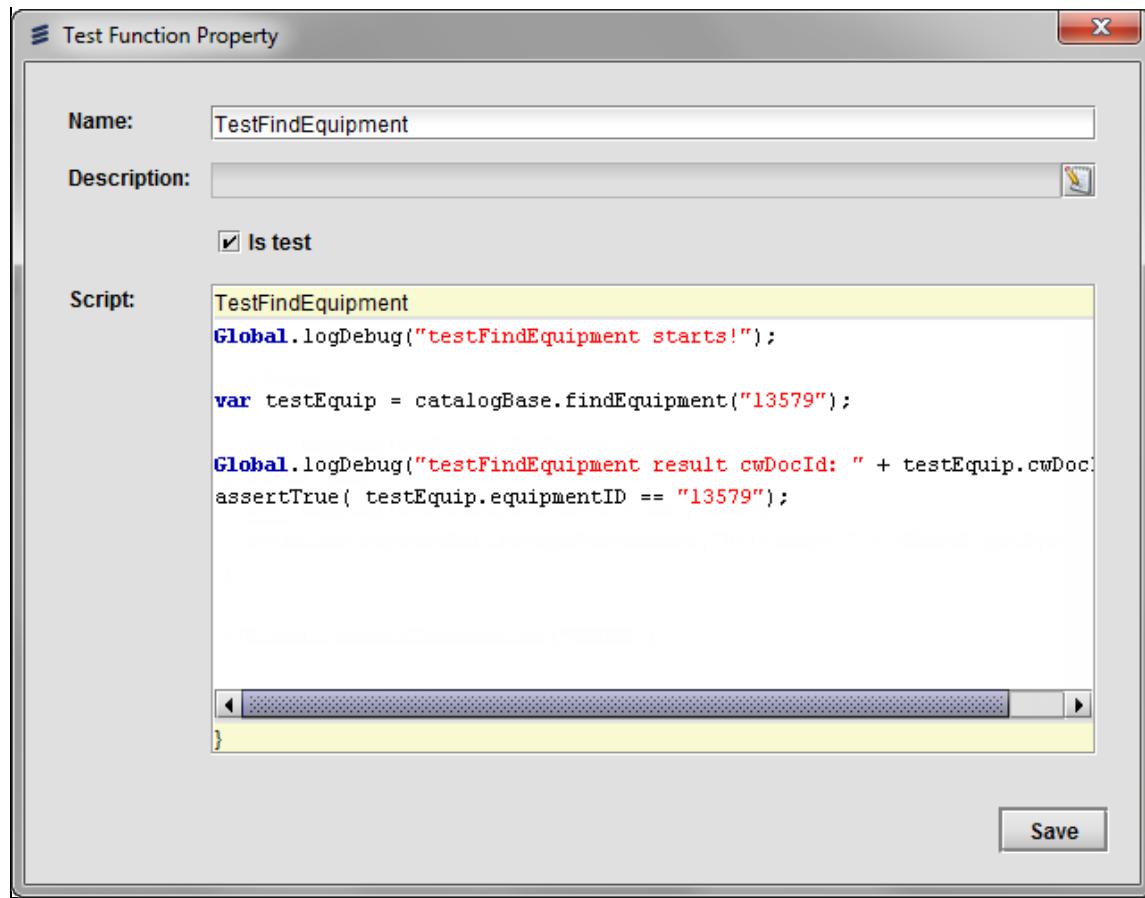
The following table contains a description of the Test Case fields.

Property	Mandatory/Optional	Description
<b>Name</b>	Mandatory	Unique name for the Test Case.
<b>Label</b>	Mandatory	Visible name for the Test Case.
<b>Description</b>	Optional	Descriptive text for documentation.
<b>Duration</b>	Optional	The maximum allowed time of the Test Case execution before the test case results in a failure; it can be specified in combination of hours, minutes, and seconds.
<b>Setup SQL</b>	Optional	Text area that contains the SQL script to be processed before all Test Functions' execution. It is processed as a procedure, thus the SQL script must be contained within <b>BEGIN</b> and <b>END</b> ; keywords.
<b>Setup JavaScript</b>	Optional	Text area that contains the JavaScript to be processed before all Test Functions' execution.
<b>Functions</b>	Optional	Contains list of Test Functions that defines the Test Case. The top two functions are always the Setup SQL and Setup JavaScript, and the bottom two functions are always Teardown JavaScript and Teardown SQL. The label of Setup and Teardown Functions becomes blue and bold whenever the corresponding Function is populated.
<b>Teardown JavaScript</b>	Optional	Text area that contains the JavaScript to be processed after all Test Functions' execution.
<b>Teardown SQL</b>	Optional	Text area that contains the SQL script to be processed after all Test Functions' execution. It is processed as a procedure, thus the SQL script must be contained within <b>BEGIN</b> and <b>END</b> ; keywords.

## Test Functions

To create or edit a Test Function, do the following:

1. To **create** a Test Function:
  - o From the general properties, click the **Add** button in the **Functions** field.
2. To **edit** an existing Test Function, including Setup/Teardown scripts:
  - o From the general properties, select the Test Function to be changed and click the **Properties** button.
3. The Test Function property pop-up box appears (see figure that follows). Enter or change a test function by changing the fields in the Test Function Property form. The table following contains a description of these fields.



Field	Description
Name	Mandatory. Unique name for the Test Function.
Description	Descriptive text for documentation purposes.
Is test	Indicates whether the function is an active test or not. If unchecked, the Test Function is not processed by the Test Case. See the example that follows on how selecting this property displays in the <b>Functions</b> field.
Parameters	Available only when <i>Is Test</i> is unchecked. Configures list of parameters for the non-test Function. When the non-test Function is invoked, the ordering of parameters follows the positions listed here, from top to bottom. To sort the ordering among parameters, use the Up and Down Arrow buttons to move a selected parameter up or down the list.
Script	Text area that contains the test function. The Test script is in JavaScript language.

If your function has the **Is test** property checked, the function appears in bold font in the **Functions** field. Otherwise, when this property is not selected, the function appears in normal font, as shown in the following example:

Name: sampleTestCase  Private  Restricted  Deprecated

Label: Sample test case

Description:

Duration:  Hour(s)  Minute(s)  Second(s)

Setup SQL:

Setup Javascript: `this.setProperty("selenium",sel.login("upadmin","upadmin", "label=" ));`

Functions:

- Function Setup SQL
- Function Setup Javascript
- tmd.sampleTestCase.sampleTestFunctionIsTest**
- tmd.sampleTestCase.TestFunctionNoTest

Function Teardown Javascript

Function Teardown SQL

Teardown Javascript: `this.getProperty("selenium").stop();`

Teardown SQL:

**Test Function Property**

Name: sampleTestFunctionIsTest

Description:

Is test

For all Test Functions, except for Setup/Teardown SQL, the Script field can be opened using the JavaScript Editor, by right-clicking the **Script** area, and select the **Editor** command.

For Setup or Teardown SQL, the Script field can be opened using the SQL Editor, by right-clicking the **Script** area, and select the **Editor** command.

To sort the ordering among Functions, use the Up and Down Arrow buttons to move a selected Function up or down the list.

To remove a Test Function, click the existing Test Function in the **Functions** field and click the **Remove** button.

Following regular Unit Test practices, a Test Function typically includes assertion statements as the means to validate the intended behaviour of a component. See the [JavaScript Documentation](#)'s CwAbstractTest class for a list of specific assertion methods. Since the TestCase class extends the CwAbstractTest class, in any test case created in Velocity Studio, all assertion methods can be used directly.

## Sequence of Test Function Processing

Optionally, Setup and Teardown scripts and functions can be provided to support the pre- and post-activities of the Test Case (for example, to create a dummy test entity before test, and to delete the entity after test). The Setup and Teardown scripts, at the Test Case level, are different from the Setup and Teardown Functions, within the Function level. The sequence of processing is as follows:

1. Setup SQL script
2. Setup JavaScript
3. Setup SQL Function
4. Setup JavaScript Function
5. Test Function
6. Teardown Javascript Function
7. Teardown SQL Function
8. Teardown JavaScript
9. Teardown SQL script

Note that the Setup and Teardown Function runs for every Test Function, but the Setup and Teardown script runs only once for each Test Case. For example, if there are two Test Functions in the Test Case (F1, F2), then the sequence of processing is as follows:

1. Setup SQL script
2. Setup JavaScript
3. Setup SQL Function
4. Setup JavaScript Function
5. F1
6. Teardown JavaScript Function
7. Teardown SQL Function
8. Setup SQL Function
9. Setup JavaScript Function
10. F2
11. Teardown JavaScript Function
12. Teardown SQL Function
13. Teardown JavaScript
14. Teardown SQL script

Among Test Functions, the order of which Test Function runs first is determined by top-to-bottom positions among the Test Functions in the Functions field; that is, the top Test Function runs first among all Test Functions.

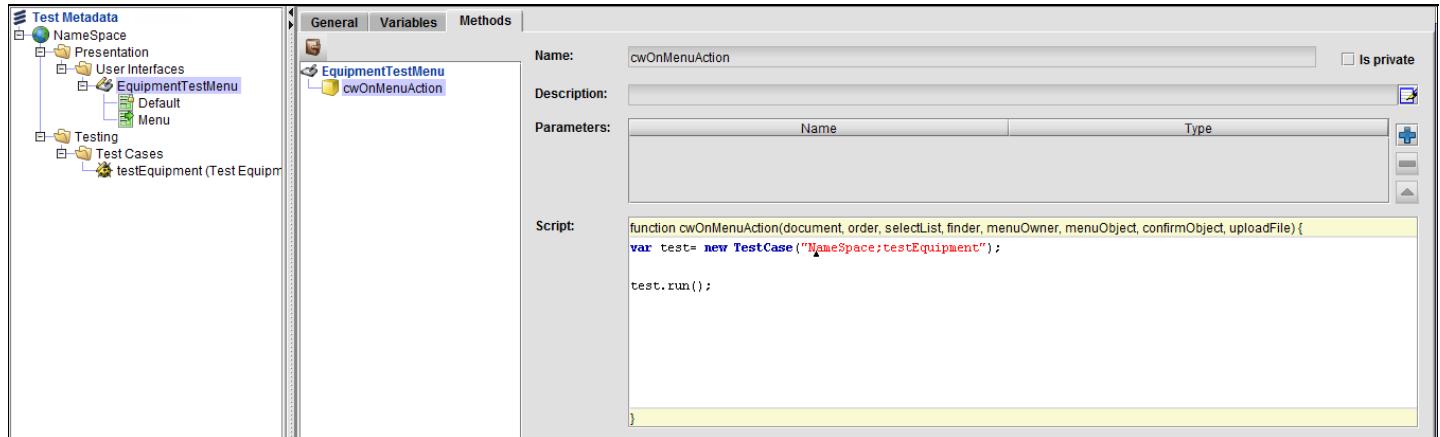
## Run a Test Case

Upon completion of all desired Test Functions and Setup/Teardown scripts in a Test Case, the Test Case can be executed by the following script:

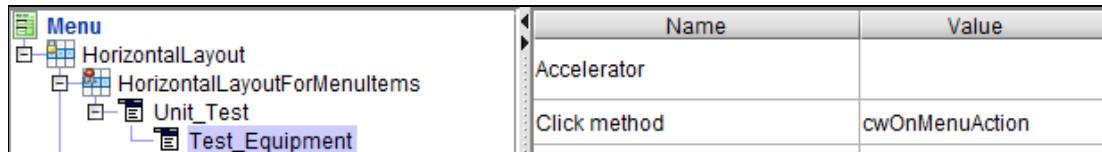
```
var test = new TestCase("namespace_name: testcase_name");
test.run();
```

Where *namespace\_name* is the name of the Test Case's Namespace, and *testcase\_name* is the name of the Test Case.

A recommended practice of Unit Test development is to develop a Unit Test drop-down menu. For each written Test Case, create a new User Action method *cwOnMenuAction* that under the Unit Test dropdown menu that invokes the Test Case using the previous script (see the figure that follows).

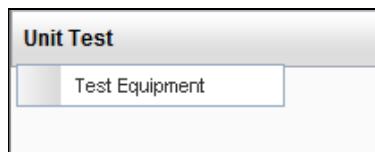


The User Action method - *cwOnMenuAction* is assigned to the **Click Method** property of the dropdown menu.

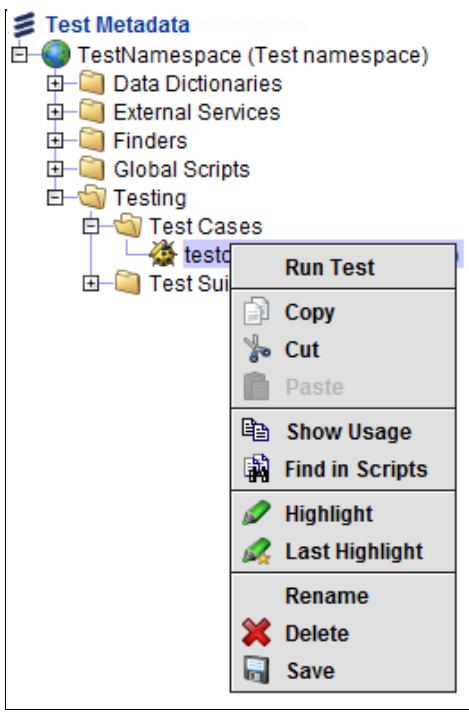


The Test Case can then be run by starting and logging into the application and clicking on the menu item (see the figure that follows).

**Note:** A Test Case can also be run as part of a Test Suite. See the next section for details.



You can also right-click your test case object in Velocity Studio's left pane and select **Run Test**.



After the Test Case is run, go to application's Event Log to observe the outcome (see the figure that follows). By default, the Test Case execution generates Test Case Start information log at the beginning, and Test Case End information log with Test Case result summary at the end.

The screenshot shows the 'Event Log' interface. At the top, there is a search and filter panel with fields for:

- Id
- NODE ID
- Transaction ID
- Date >=
- Date <=

Below the search panel is a table with columns:

- User
- Module
- Severity
- Code
- Process ID

At the bottom is a detailed log table with columns:

- Id
- NODE ID
- Date
- Module
- Severity
- Code
- User ID
- Transaction ID
- Description

For any assertion statements that have failed in any Test Function, the exception is shown in the event log as well. The Test Case continues to execute other Test Functions in the event of a Test Function failure.

## Enable and Disable Test Functions

The **Is test** checkbox, when selected, indicates that the test case runs the test function. You can also select or deselect this property by following these steps:

- From your test case, select the function that you want and right-click it to display the menu.

Name:	TestCase1	<input type="checkbox"/> Private	<input type="checkbox"/> Restricted	<input type="checkbox"/> Deprecated
Label:	TestCase1	...		
Description:				
Duration:	1	Hour(s)	Minute(s)	Second(s)
Setup SQL:				
Setup Javascript:				
Functions:	Function Setup SQL Function Setup Javascript <b>TestNamespace.TestCase1.login</b> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  Copy   Enable Test   Disable Test   Paste       </div>			
	Function Teardown Javascript Function Teardown SQL			
	<input type="button" value="Add"/> <input type="button" value="Remove"/> <input type="button" value="Properties"/>			
Teardown Javascript:				
Teardown SQL:				

2. To enable the selected test function, select **Enable Test**, which selects the test function's **Is test** option. Similarly, to disable the selected test function, select **Disable Test**, which deselects the test function's **Is test** option.
3. The right-click menu also allows you to select a test function and **Copy** its contents. You can then select **Paste** from the menu and create a new function with the same functionality as its original copied test function.
4. When you have completed your changes, save your metadata.

### Set a BreakPoint on a Test Case Object

You can set a breakpoint on a Test Case object by right-clicking a function and selecting **Break on first line** from the menu.

<b>SDE200 Training Metadata</b> <ul style="list-style-type: none"> <li><input type="checkbox"/> applicationUI (Application UI)</li> <li><input type="checkbox"/> billingClient (Billing Client)</li> <li><input type="checkbox"/> billingServer (Billing Server)</li> <li><input type="checkbox"/> customerManagement (Customer Management)</li> <li><input type="checkbox"/> orderManagement (Order Management)</li> <li><input type="checkbox"/> test           <ul style="list-style-type: none"> <li><input type="checkbox"/> Data Dictionaries</li> <li><input type="checkbox"/> Testing               <ul style="list-style-type: none"> <li><input type="checkbox"/> Test Cases                   <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> testCase1 (Test Case 1)</li> </ul> </li> </ul> </li> </ul> </li> <li><input type="checkbox"/> upStuff (UP Stuff)</li> </ul>	<p><b>Name:</b> testCase1</p> <p><b>Label:</b> Test Case 1</p> <p><b>Description:</b></p> <p><b>Duration:</b> <input type="text"/> Hour(s) <input type="text"/></p> <p><b>Setup SQL:</b> <input type="text"/></p> <p><b>Setup Javascript:</b> <input selenium\",sel.login(\"upadmin\",\"upadmin\"))"="" type="text" value="this.setProperty(\"/></p> <p><b>Functions:</b></p> <pre>Function Setup SQL Function Setup Javascript <b>test.testCase1.testScript</b></pre> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <span style="border: 1px solid red; padding: 2px;">Break on first line</span>   <span> Copy</span> <span> Enable Test</span> <span> Disable Test</span> <span> Paste</span> </div>
---	---

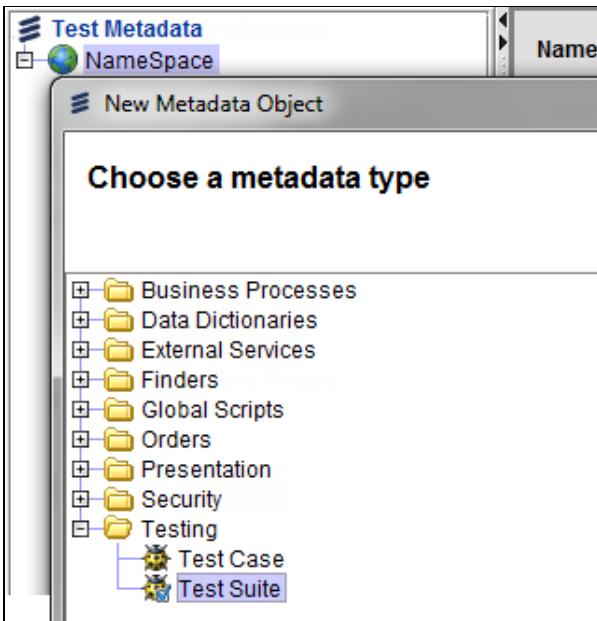
## Test Suites

A Test Suite is a collection of Test Cases. A Test Suite may also contain other Test Suites. Running a Test Suite processes all its Test Cases contained within the hierarchy.

### Creating a Test Suite

Test Cases should have been created before creating a Test Suite. To create a Test Suite, perform these steps:

1. Right-click the namespace and select **New**.
2. From the **Testing** folder, select the **Test Cases** menu option and then click the **Next** button.



3. A new Test Suite form displaying the test case properties appears.

### Test Suite General Properties

The following figure shows the general properties of a Test Suite. The table that follows contains a description of the Test Suite fields.

Name:	testSuiteEquipment
Label:	Test Suite Equipment
Description:	
Setup SQL:	
Setup Javascript:	Global.logDebug("Test Suite Equipment setup");
Tests:	<p>NameSpace.testEquipment</p> <p> NameSpace.testEquipment (Test Equipment) <span style="float: right;">▼</span> <span style="float: right;">Add</span> <span style="float: right;">Remove</span></p>
Teardown Javascript:	Global.logDebug("Test Suite Equipment teardown");
Teardown SQL:	

Property	Mandatory/Optional	Description
Name	Mandatory	Unique name for the Test Suite.
Label	Mandatory	Visible name for the Test Suite.
Description	Optional	Descriptive text for documentation purposes.
Setup SQL	Optional	Text area that contains the SQL script to be processed before all Test Cases' execution. It is processed as a procedure. As a result, the SQL script must be contained within <b>BEGIN</b> and <b>END;</b> keywords.
Setup JavaScript	Optional	Text area that contains the JavaScript to be processed before all Test Cases' execution.
Tests	Optional	Contains a list of Test Cases and Test Suites that defines the Test Suite.
Teardown JavaScript	Optional	Text area that contains the JavaScript to be processed after all Test Cases' execution.
Teardown SQL	Optional	Text area that contains the SQL script to be processed after all Test Cases' execution. It is processed as a procedure. As a result, the SQL script must be contained within <b>BEGIN</b> and <b>END;</b> keywords.

To add a Test Case into the Test Suite, select the Test Case in the dropdown box and then click the **Add** button in the **Tests** field.

To edit a Setup or Teardown script, click the corresponding ellipsis button. For Setup or Teardown JavaScript, the Script field can then be opened using the JavaScript Editor, by right-clicking the text area, and selecting the **Editor** command.

For Setup or Teardown SQL, the Script field can be opened using the SQL Editor, by right-clicking the text area, and select the **Editor** command.

To sort the ordering among Tests, use the Up and Down Arrow buttons to move a selected Test up or down the list.

To remove a Test, click the existing Test in the **Tests** field and click the **Remove** button.

### Sequence of Test Function execution

Optionally, Setup and Teardown scripts can be provided to support the pre- and post-activities of the Test Suite (for example, to create a dummy test entity before test, and to delete the entity after test). The sequence of processing is as follows:

1. Setup SQL
2. Setup JavaScript
3. Tests
4. Teardown JavaScript
5. Teardown SQL

The setup and teardown runs only once for *all* Tests in the Test Suite. For example, if there are two Test Cases in the Test Suite (TC1, TC2), then the sequence of processing is Setup SQL, Setup JavaScript, TC1, TC2, Teardown JavaScript, Teardown SQL.

Among Tests, the order of which Test runs first is determined by top-to-bottom positions among the Tests in the **Tests** field; that is, the top Test runs first among all Tests.

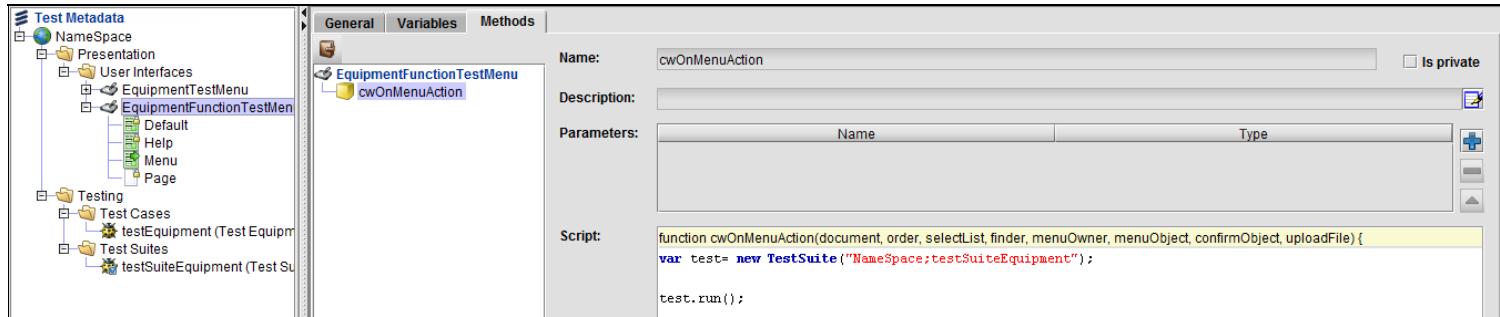
## Run a Test Suite

The Test Suite can be run by the following script:

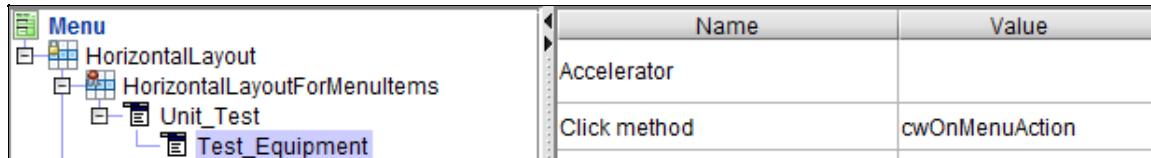
```
var test = new TestSuite( "namespace_name:testsuite_name" );
test.run();
```

where *namespace\_name* is the name of the Test Suite's Namespace, and *testsuite\_name* is the name of the Test Suite.

A recommended practice of Unit Test development is to develop a Unit Test drop-down menu. For each written Test Suite, create a new menu item under the Unit Test dropdown item. Create a new User Action method, *cwOnMenuItem* using the previous script (see the figure that follows) and assign this method to the Menu's **Click Method** property.



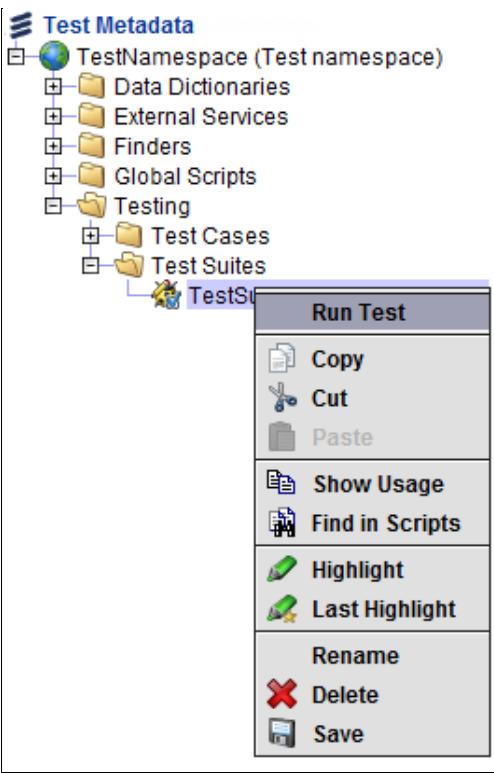
The User Action method - *cwOnMenuItemAction* is assigned to the **Click Method** property of the drop-down menu.



The Test Suite can then be run by simply starting and logging into the application, and then clicking the menu item (see the figure that follows).



You can also right-click your test suite object in Velocity Studio's left pane and select **Run Test**.



After the Test Suite is run, go to application Event Log to observe the outcome. By default, the Test Suite execution generates *Test Suite Start* information log at the beginning, and *Test Suite End* information log with Test Suite result summary at the end.

The screenshot shows the 'Event Log' interface. At the top, there is a toolbar with icons for PEs/AVMs, Interfaces, User Profile Manager, Tools, Favorites, and Providers. Below the toolbar, there is a search bar with dropdowns for Id, NODE ID, Transaction ID, Date >=, and Date <=. There are also filters for User, Module, Severity, Code, and Process ID.

The main area displays a grid of event log entries with columns: Id, NODE ID, Date, Module, Severity, Code, User ID, Transaction ID, and Description.

If no errors occurred, the event log only shows the start of the outermost Test Suite and the ending of the outermost Test Suite. If errors occurred (for example, failed assertion statements in a Test Case), the event log shows the error in the Test Case, and continues to run subsequent Test Cases. At the end, it shows how many Test Cases were successful in the Test Suite.

## Internationalization

---

Applications built by Velocity Studio enables clients to work simultaneously in different languages. The elements of Velocity Studio applications that can be internationalized are set by establishing a *Translation* per language, which are Excel files that can be imported and exported of the following types:

- Attributes of the metadata elements such as label, help, etc.
- Messages in validation rules.
- Messages for applications, the system and users.

The first two types of Translation are user-defined, which means a Translation record can be added or removed, and can be translated using Translation. Conversely, the last type is system-defined, which can be translated using Translation but the translation record cannot be added or removed.

Translation supports the internationalization of the following elements:

- The label and help of the data types, documents, orders, menus, forms and dashboards.
- The labels of the interfaces, operations, exceptions, signals, participants, alerts, processes, finders and navigation trees.
- The messages of validation rules in the data types, documents and orders.
- The enumeration code descriptions in the data types.
- The labels of the leaves in the documents.
- The label, help, range axis and domain axis of the chart items in the finders.
- The application, system and user messages.

At the top-level application metadata object, the [Languages tab](#) defines all languages that shall be supported by the application runtime, as well as a default language that shall be used when browser requests have no language specified. All Translations can only be made to this list of languages defined by the application metadata.

## Client Browser Language Settings

To see the runtime Web application in the desired language, the following steps should be taken:

1. The language specific Translation should be created using [Translation Import](#) menu command.
2. The browser language should be set to the same language. For example, in Internet Explorer, do the following steps to set up the language:
  - Select **Internet Options...** from the Tools menu. The Internet Options dialog opens.
  - Click the **Languages...** button at the bottom of the Internet Options dialog. The Language Preference dialog opens.
  - If you do not see your language in the Language list of the dialog, click the **Add...** button, select your language, and then click the **OK** button. The language is added to the Language list.
  - Select your language in the Language list and use the **Move Up** button to move the selected row to the first position in the list.
  - Click the **OK** button in the Language Preference dialog, and then click the **OK** button from the Internet Options dialog.
3. If the character set of the selected language is not installed on your computer, you are prompted to install it when running the Web framework for the first time. You must have privileges to install programs on your computer to run the installation.
4. The Web framework uses the browser language setting to select the language resource according to the following rules:
  - The language code is taken from the browser request (for example, Ir-cc).
  - If language resource Ir-cc exists, it will be used.
  - If language resource Ir-cc does not exist, but the language resource Ir-xx exists, the latter will be used.
  - If language resource Ir-xx does not exist, the default language resource specified in the [top-level Metadata's Language tab](#) will be used.

**Note:** You can sort language resource files by keyword (for example, name) after saving all metadata.

## Translation Fallback for Labels, and System and User Messages

Velocity Studio uses a system default language. By default, it uses EN-XX.

The fallback for translations for labels, and system and user messages is as follows:

- If the language requested is the system default language, it will not fall back to any other language.
- If the language requested is in the same family as the system default language, it will fall back to the system default language if either one of the following conditions are met:

- o The language requested is not defined in the metadata
- o The language requested has no value for the requested code
- If the language requested is a language is not in the same family (see the definition that follows the table) as the system default language:
  - o If the language requested is defined in the metadata, but does not contain the code requested, it falls back to the system default language. If the language requested is not defined in the metadata, it falls back to the –xx version of its language. Failing that, it falls back to the next language in the same family. Failing that, it ultimately falls back to the system default language.

As an example, the language fallback priority is as follows:

1. DE-CH falls back to DE-XX.
2. If DE-XX is not defined, it falls back to whichever DE- it finds first that is defined in the metadata.
3. If there are no other DE-XX languages defined, it falls back to the system default language (EN-XX).

The following table summarizes the language requested, whether the language is defined in the metadata or code, whether the language in the same language family has been defined in either the metadata or code, and the final fallback result.

Language requested	Has language requested been defined in metadata?	Code is defined in language requested?	Has language in same family* been defined in metadata?	Code is defined in language in same family?	Fallback result
System default (EN-XX)	Not applicable	Not applicable	Not applicable	Not applicable	System default (EN-XX)
System default family (EN-CA)	Yes	Yes	Not applicable	Not applicable	System default family (EN-CA)
	No	Not applicable	Not applicable	Not applicable	System default (EN-XX)
	No	Not applicable	Not applicable	Not applicable	System default (EN-XX)
Non-system language (DE-something)	Yes	Yes	Not applicable	Not applicable	Non-system language (DE-something)
		No	Not applicable	Not applicable	System default (EN-XX)
	No	Not applicable	Yes	Yes	Language in same family (for example, DE-XX)
				No	System default (EN-XX)
				No	System default (EN-XX)

\* Language in the same family means languages that have the same prefix.

For instance, these languages belong to the same family:

- EN-XX
- EN-CA
- EN-US

Another example of languages belonging to the same family is as follows:

- DE-XX
- DE-CH
- DE-AT

**Note:** This fallback strategy is primarily based on whether the possible fallback language is defined in the metadata, and is not based on whether the possible fallback language actually has the code requested. Ultimately, the last fallback language is always the system default language. As a result, it is recommended that you ensure that any message code has a value in the system default language.

## Translation Finder

The translation finder popup uses the following system event:

Event	Returns	Description
UI_GET_TRANSLATION_FINDER	This event returns an instance of either the translation finder or the metadata name.	<p>This event contains the following parameters:</p> <ul style="list-style-type: none"><li>• targetObj This parameter contains either the target document or data structure</li><li>• varName This parameter denotes the variable name</li><li>• parent This parameter contains the parent UI object</li></ul>

The default event handler, located in the UI Common library, returns a string with the finder's metadata name, ui\_common.translationFinder, for any target object and variable of translation type.

### Use the Translation Finder User Interface

To be able to pop up the translation finder, the translation field needs to contain a value. When the translation field is empty, the reference button is disabled.

Once you have text in the translation field and you save your changes (the entered value saved for the current language being used), the reference button is enabled. Enabling this button allows you to add and edit the translation contained in this field.

Translation text can be added for any available language. To manage your available languages, select your project's metadata header and click the Languages tab in Velocity Studio.

**Note:** When adding a new translation, if you select a language that already has a translation record, the existing value is replaced with the new translation value.

The following is an example of how to set up your metadata to use the translation finder user interface:

1. Create a document that contains some test variables, including a variable of com.conceptwave.system.Translation type.
2. Create a form for the document. Proceed to add a reference field for the translation variable.
3. Add menu actions or use a finder for the following actions:
  - Create a new document
  - Open a saved document
  - Open a saved document that is read-only (you can read the document from the database by setting `document.readonly = true;`)
4. Map the document to the database.

## Best Practices for Internationalization

---

When considering internationalization, follow these design points:

- Minimize duplication in labels to reduce translation effort.
- Allow labels to be inherited from the variable level rather than specifying them at the form level.
- Use access rules to hide or enable fields rather than new forms to reduce the translation effort.
- Avoid the use of plain text in rules. Instead, use code that can be translated.
- Use substitution parameters in rules to reduce the number of different messages.
- Create a central registry of error messages using the resource editor, to ensure that messages are reused.
- If appropriate, define translations for the general language (for example, en-xx) and omit translations for the specific language (for example, en-ca, which is not appropriate for all languages).
- Model the solution in the primary or most common language for the solution.
- Where template customizations consist primarily in relabeling fields, consider using the Resources tab to change the labels, rather than overriding or changing the underlying template.

## Error Messages

Error messages are defined by text or by code. While in either case, the resulting message can be internationalized, code based error messages are preferable.

- The Resources tab provides a central location to manage all error messages, rather than navigating to all data element, document, and order validation rules.
- The use of code provides an opportunity to reuse of messages, thereby reducing the translation effort.
- The use of code allows the message parameter substitution capability to be used, reducing the translation effort (for example, a message defined as No value in mandatory field {0} requires one translation for the message, with an additional one for each variable label, rather than one message and one label for each variable).

## Generated Text

Avoid using non-metadata-based text within the solution, such as rules that are generated. This text is not visible to the Resources tab and cannot be internationalized. Common situations include the following:

- Computed form fields
- Visual keys
- Browser window title (application name)

Where text is desired within these fields, these basic means exist to provide internationalization:

- The Resources tab's error code translation facility may be used to create an error code with the associated text. The script may generate the translated value directly by generating an error code and using `Global._translateText()` to return the translated text, which is the recommended mode when the resulting text is appended to other values or text.
- A data type may be defined containing an enumerated list of codes and labels. The product would then be used to translate the enumeration. The script may generate the translated value directly by generating a code, and using `Global._getCodeDescription()` to return the translated text for the code.
- Similar to the previous method, if the target field is defined with the enumeration as the base type, the translated description is automatically generated by the product, which is the recommended approach where the resulting text is the final text.

**Note:** The generated text should be transient in nature, which means it should be recalculated on each view. If it is persisted and not recalculated, a user from another locale will see the original translation and not the translation appropriate for the locale.

## Field Labels

Field labels within the product follow an inheritance. If the form label is not specified, the variable's visual label is used, which can be defined using the **Element Properties** field. To reduce the effort in translation, use the default label wherever possible.

- Specification of labels at the form level requires additional effort in the initial modelling, especially where multiple forms are defined that include the same variables.

- Changes to the labels assigned to the underlying variables are not automatically propagated to the forms.
- Internationalization is complicated as the translation effort is performed for each form, not only at the variable level.
- Even where form field labels are appropriate, consider using the variable's visual label for the most common label and specifying the form field labels only for the exceptions.

## **Business Data**

Where business data translation is required (for example, with reference tables), it is the responsibility of the designer to provide translation facilities and to invoke them.

- For example, a separate table may be created and managed to hold the translations by language (see Catalog Management or the core product's code tables).
- Catalog Management can be used to manage these tables, which is the recommended approach for dynamic data.
- Product's code tables can be used to manage these tables, which is the recommended approach for static data.

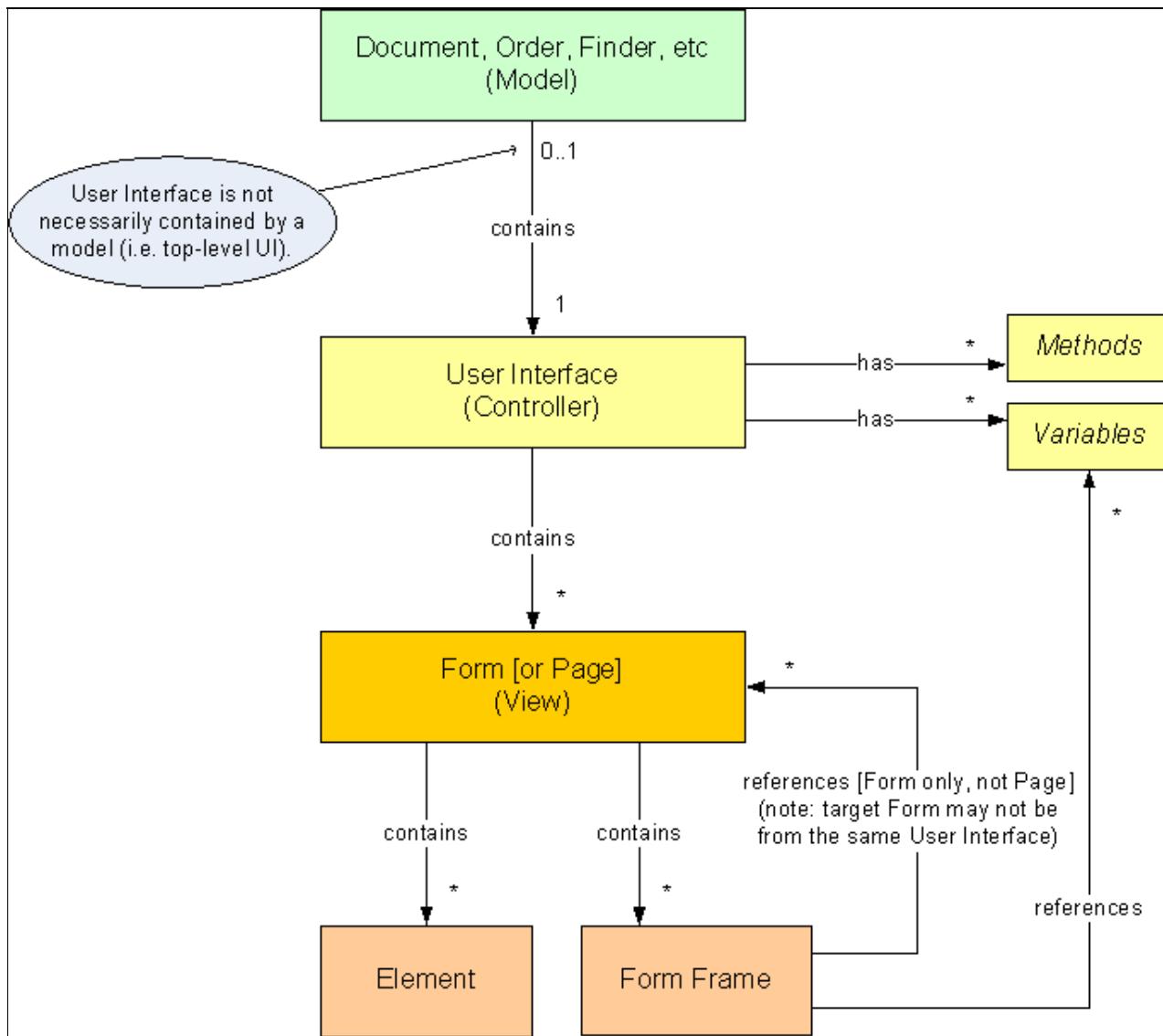
## User Interface Design

The presentation-oriented design uses [MVC architecture](#) patterning to configure and then display data at runtime. This is achieved by introducing a new metadata object type called the **User Interface**, that implements the *controller* of the architecture pattern. There are two different types of User Interface available:

- *Included User Interface*
- *Top-level User Interface*

Included User Interfaces are defined within a metadata object that may contain User Interface objects such as **Documents**, **Orders**, and **Finders**. With an included User Interface, the *model* in MVC pattern connects with the controller. Top-level User Interfaces are defined as a stand-alone metadata object, although their model is attached to the data source. Either type of User Interfaces provides data and business logic access to the **View** in the MVC pattern using **Variables** and **Methods**.

Views in the MVC pattern query the Model to generate an appropriate User Interface. A Form is the equivalent of a View in the MVC pattern and is the vehicle that queries the Model and defines a presentation fragment of a Web page. Forms are not rendered as stand-alone, but are always referenced, by a [Form Frame](#), or as part of another Form or **Page**. A User Interface can contain multiple Forms, each with its own purpose. For example, by default a Document Finder has multiple Forms within the Finder's User Interface; a parent Form that references a **Search** Form (search border) and a **Results** Form (result table). Another example is, two Forms can create a Document called **Product**; one for displaying basic information (product name, product code) and another for revealing all product details (description, features and specifications). Examples of the runtime presentation of a Form are: the presentation of a customer Document, or Web page header or footers of an Order Finder.



The **View** of the pattern is also constructed by creating a **Page** within User Interface. The **Page** is a top-level View that is very similar to the **Form**.

object, but unlike Forms, it is never referenced by another Form or Page. Only one Page can be defined for each User Interface object. The Page is the object that is referenced by the User Interface object and is displayed by the Web server.

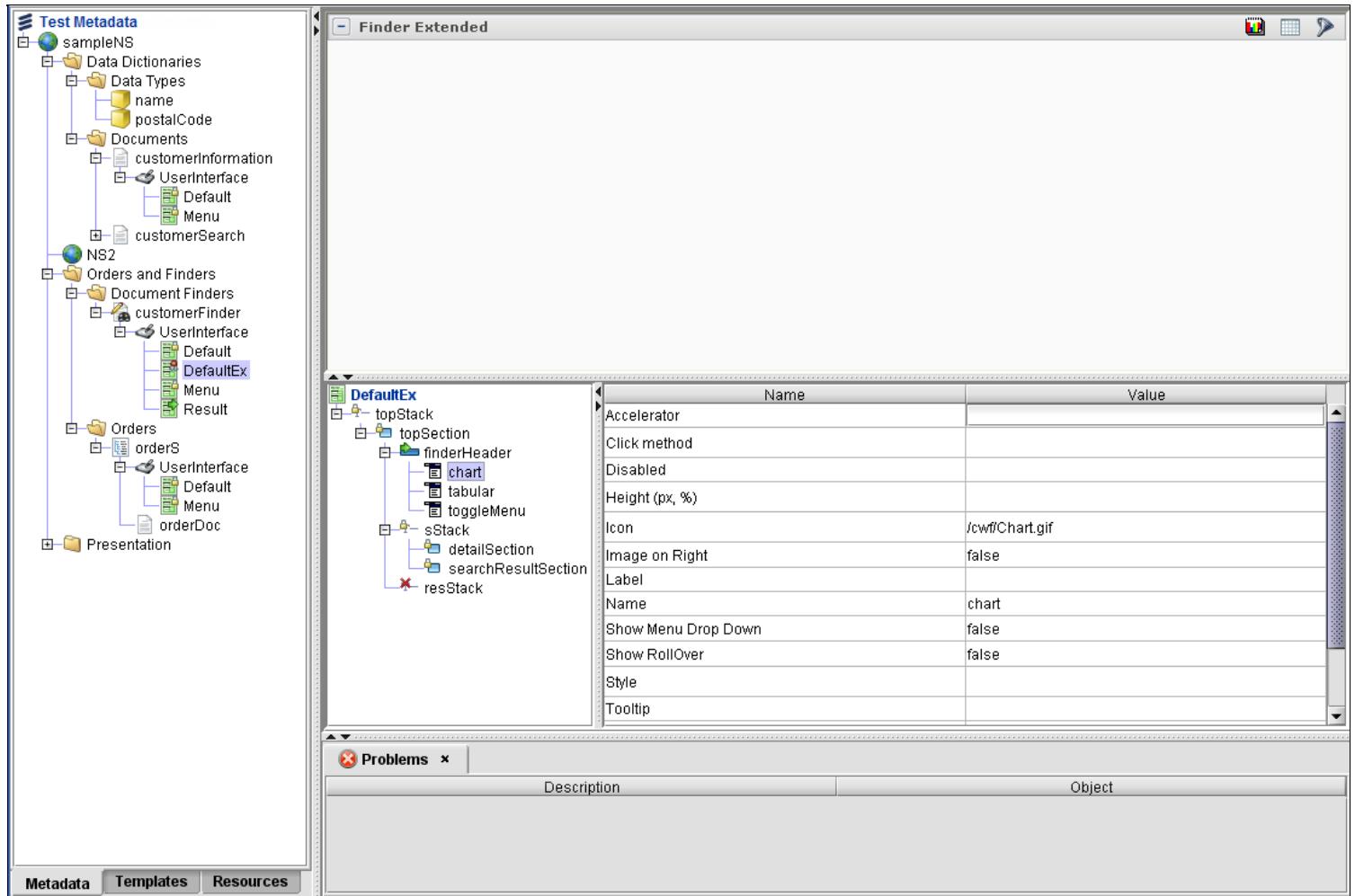
Whether it is a Form or a Page, it may contain many **Form Elements** that compose the Web page or Web fragment. These elements are assembled in a tree structure called the **Element Tree**. The **Preview Pane** in Velocity Studio, when editing a Form or Page, renders the UI view based on the Element Tree. Elements may be distinct objects such as Labels and Images, layout directives such as Vertical or Grid Layout, organizational directives such as Tabs, Tabs and Section Stacks, or User Interface elements such as Tables and Trees. See [List of Form Elements](#) for the complete list of Elements available.

## Re-use Common User Interface Objects

The **Form Frame** is an element that references other Forms, and enables a Page or a Form to include another Form as part of its view. In the MVC model context, the Form Frame's variable, which references a Form, enables the View to present the Model permitted by the Controller. Common User Interface objects such as application menus and banners are constructed as a Form, and then are included into a Page as Form Frame. These common objects can be reused by other Forms or Pages. The referenced Form and the Page that contains the Form Frame does not have to be in the same User Interface (this is typically the case). In addition, the Form Frame can assume a variable in the User Interface. A common example is to use a Form Frame to reference an Order Form that only exposes user-meaningful fields of the Order, and references an Order variable that is managed by User Interface's methods.

## Accelerate Development by Extending objects

One of the key advantages of Velocity Studio is its ability to accelerate project development by extending existing base metadata objects and making a set of functionality available. This reduces development time by refining and adding supplementary functionality in the extended metadata objects. Forms are a good example of extended metadata functionality. For example, a Form can be extended by another Form. This provides the ability to augment the presentation of metadata objects such as Document, as defined in existing base Forms. The extended Form is located in the same User Interface as the base Form, and therefore they share the same set of variables and methods provided by the User Interface. The extended Form is enhanced from the base Form by **adding**, **replacing**, and **deleting** Elements in the Form (however it cannot modify properties of existing Elements).



The metadata object that has a User Interface (that is., a Document) can be extended. In this case, the User Interface of the extended object is a copy of the base object's User Interface metadata, with all existing properties (that is, variables, methods, forms) fixed. Additional User Interface

variables, methods and forms are added to the extended object to configure it for the desired functionality. For example, when a variable is added at the extended object (not at the User Interface level, but at the object level itself), an extended Form can be added to the extended object's User Interface, based on one of the Forms in the base object.

In addition, the top-level User Interface object can be extended. Similarly, all existing properties of the base User Interface object is copied and fixed and additional variables, methods and forms can be added.

## User Interface Permissions

Consult the [Permissions](#) page for details.

# User Interface Objects Overview

The User Interface object controls the input and output data of a application and how it displays at runtime. The User Interface's primary function is to allow the user to control the display of data at runtime by configure variables and methods for Forms and Pages. Forms allow subsets of data to be displayed in different presentations and are used in different runtime conditions while a Page displays the layout of a page in a Web browser at runtime (Pages only exist in User Interfaces of type application).

User Interface variables are used to control specific aspects of the user interface's behaviour using Forms and Page. When the User Interface object is created, system variables are available to configure properties, but you can also create custom variables that can be used to store data or are referenced by a method. For more information see, [User Interface Variables](#).

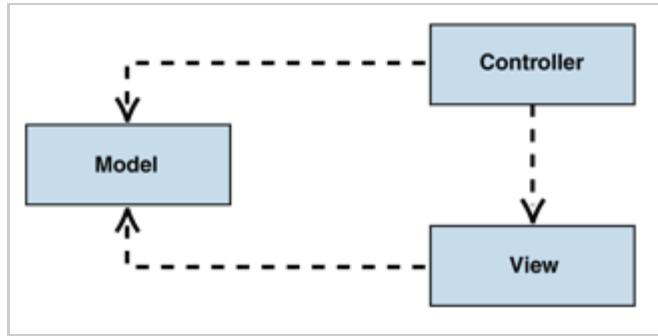
User Interface methods use a script to invoke parameters. A method can contain a script that defines an action or sets initial values for variables in the same User Interface object. For more information see, [User Interface Methods](#).

Using the metadata contained in a User Interface, you can configure properties and invoke an action on an object. The User Interface is available for several metadata objects such as:

- Documents
- Orders
- Finders

## Model-View-Controller Architecture

MVC is a software architecture design pattern for interactive applications. MVC organizes an application into three main components —**Controller, View and Model**. Controllers are used for handling events that affect the model or views, Views for displaying all or a portion of the data including the presentation and user input, and models are used for maintaining data. Both the View and the Controller depend on the Model, but the Model does not depend on either of these objects. This means that the model can be developed separately from the visual presentation.



The Controller receives input, such as an event, and initiates a response by making calls on the model object. It handles the user input and determines which model classes to call to handle the processing for it. Events typically cause a controller to change a model, or the view, and sometimes both. Whenever a Controller changes a model's data or properties, all dependent views are automatically updated. Similarly, whenever a Controller changes a view, the view gets data from the underlying model and refreshes.

A View queries the Model to generate an appropriate User Interface. The View gets its own data from the Model. The Model is a representation of the data upon which the application operates. When a Model changes its state, it notifies its associated views so they can be refreshed. Because Models, Views and the Controller are separate objects, multiple Views and Controllers can interface with the same Model.

## MVC and Velocity Studio

The presentation-oriented User Interface design in Velocity Studio separates the User Interface from its model-driven predecessor, with the use of [MVC architecture](#) patterning. The *User Interface* metadata object is considered the *Controller*, while the *View* is considered all of the *Form* and *Form elements*, and the *Model* is represented by the database.

Consider the behavior of a Form Frame Form element in an application. The Form Frame is a reference element that displays a Form.

Data is displayed in a Form Frame when a related action (such as a button click) triggers a method script to run. Each button represents an event source that spawns an action event each time that it is clicked. The button is connected to a listener that receives action events and routes them to an action listener. The action listener is an example of a controller. This controller then converts the event into the appropriate user action, one that the model can understand. The controller notifies the model of the user action (that the Form Frame will receive data). The Form Frame queries the model to generate an appropriate User Interface and the view retrieves data from the model.

## Create User Interface Objects

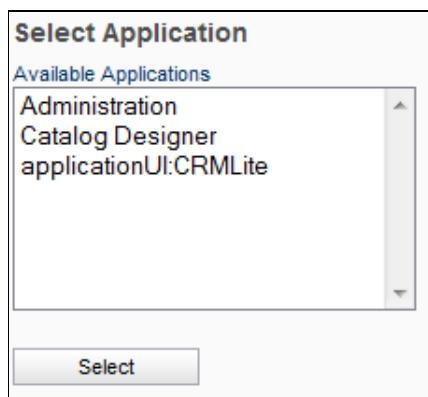
A top-level User Interface is not included within a model-based metadata object. You can create a top-level User Interface in the Presentation folder within the Navigation Pane. Although there are several different base User Interface objects (see [Table](#)) to choose from, only two top-level User Interface objects are available:

- `com.conceptwave.system.UserInterface`
- `com.conceptwave.system.Application`

The `com.conceptwave.system.UserInterface` object is a base metadata object used to extend the top-level User Interface object under the **Presentation** folder in the **Navigation Pane**. It contains child objects ( default, menu, page) that sets the runtime interfaces appearance. By default, this User Interface contains a [Page](#) object that may be presented to users as a Web page via URL mapping. The Forms and Page contain only Form elements and are for the most part, empty.

In the Page Form, the Form Frame element, which represents the dialog, displays the dialog in the User Interface. The dialog Form Frame allows objects to be created as dialog content. Without the Form Frame, objects continue to be created as dialog content, but they are not displayed. When the Page Form is copied and replaced, the Form element properties can be configured. When the dialog element's *Unmanaged* property is set to *false*, content does not display in a dialog. In addition, if the Unmanaged property is set to *false*, and the Variable property is bound to a Form Frame, the dialog content displays in the main dialog window, top and left, positioned as 0.

Similar to the `com.conceptwave.system.UserInterface` object is a metadata object that extends the top-level User Interface object. The `com.conceptwave.system.Application` object contains child objects (default, menu) that sets runtime interfaces appearance. Use the `system.Application` to add a Web application to the runtime. Once logged in to the application, select an application using the **Select Application** screen.

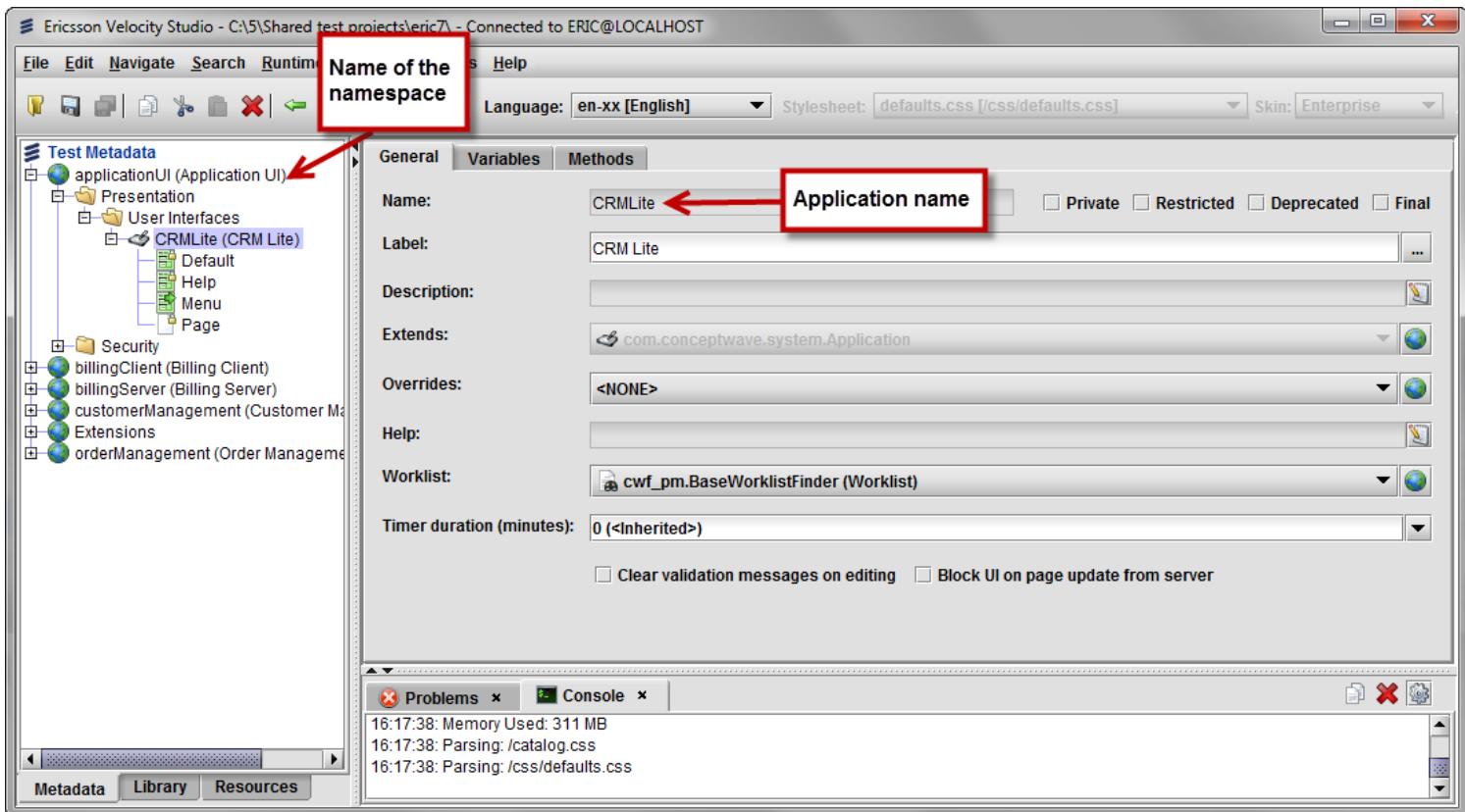


**Note:** The Select Application page is specialized. After it is created, it is set as the application, which overrides any call to `setApplication` within the `onInit()` function.

### Application Name under the Select Application dialog

The name of the application that appears in the Select Application screen has the syntax *namespace: application name*. The *namespace* represents the name of the namespace in which the application resides. The *application name* is the name defined for the application under its **General** tab.

In this example, the *namespace* is ns and the *application name* is OrderApplicationShort. As a result, the Select Application screen shows **ns:OrderApplicationShort** as an available application that you can select.



## Create a Top-level User Interface

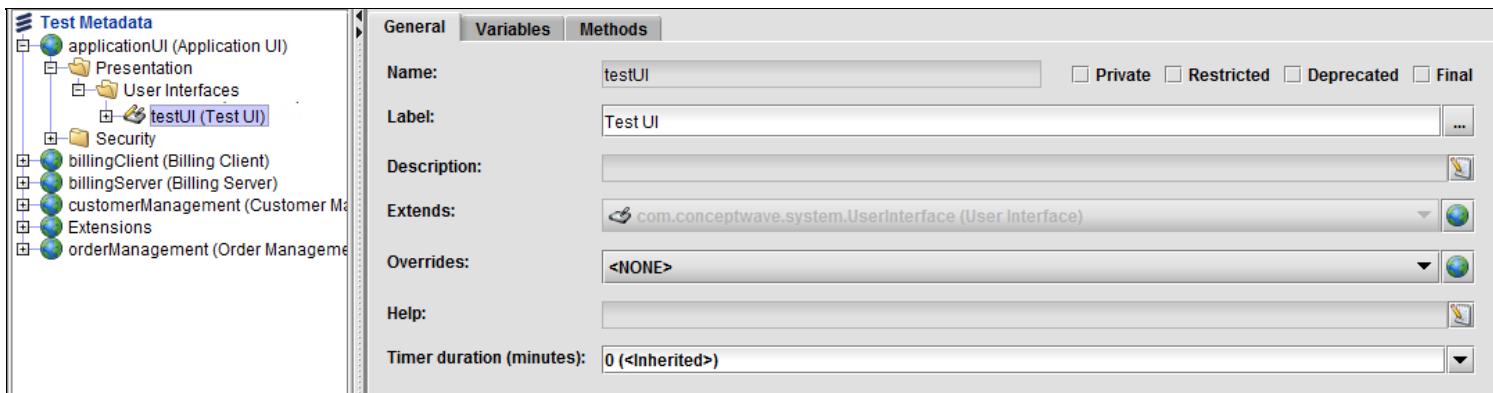
To create a top-level User Interface, complete these steps:

1. In the **Metadata tab of Navigation** pane right-click a Namespace.
2. Select **New**. The New Metadata Object wizard appears.
3. Expand the **Presentation** node, and select **User Interface**.
4. Click **Next**.
5. In the Name field, type a unique name for the User Interface (must conform to JavaScript naming conventions).
6. In the Label field, type a unique name for the User Interface (this is the name that appears in the user interface).
7. The **Extends** field, is defaulted to system-standard. You may choose a specific User Interface to extend from it.
8. Click **Finish**.

*Alternatively*

1. From **Namespace > Presentation**, right-click the **User Interfaces** folder.
2. Select **New User Interface**.
3. Continue by following steps 3-8 above.

After the User Interface is created, the node is added under the **User Interfaces** folder.



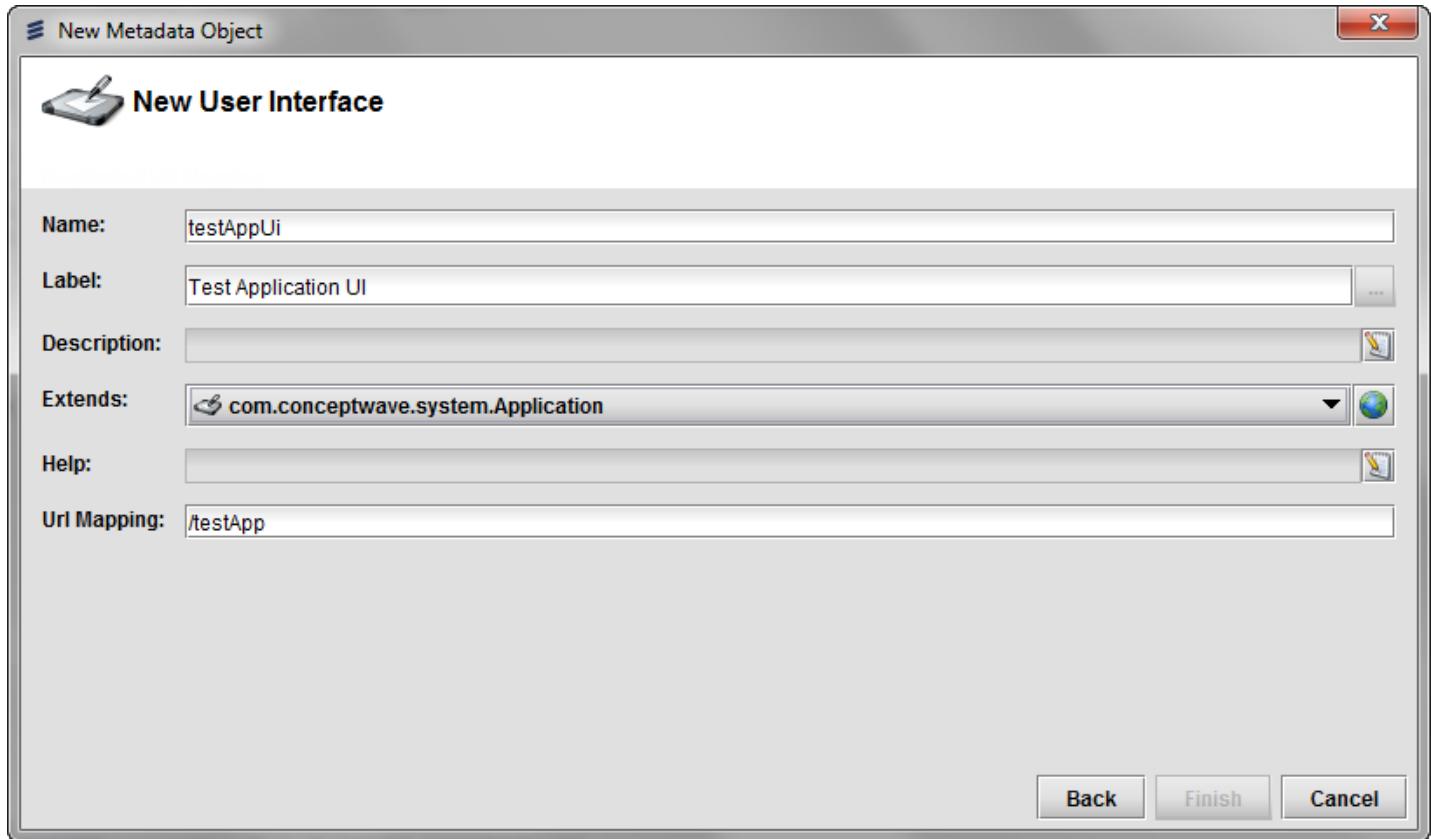
The User Interface product metadata objects are stored under the **Library** tab of the **Navigation Pane** under **com > conceptwave > System > Presentation > UserInterfaces > Application**.



## Specify URL Mapping at the Application Level

All applications that are opened or created in Velocity Studio have a URL mapping. When creating a new application in Velocity Studio, you have the ability to specify the URL mapping by completing these steps:

1. Create a [new user interface](#) to launch the New User Interface dialog.
2. For the **Extends** field, select **com.conceptwave.system.Application** to show the **URL Mapping** field.



3. In the **URL Mapping** field, specify the URL segment after `http://<host>:<port>`. In this example, the URL to access this application is `http://<host>:<port>/testApp`. Ensure that you enter the forward slash (/) before your URL segment in this field. By default, the URL mapped for each new application is the name of the application.
4. Click the **Finish** button to create your application's user interface.
5. You can verify that your URL mapping was successful by clicking your root metadata and selecting the Application Page tab.

**Test Metadata**

applicationUI (Application UI)

- Data Dictionaries
- Data Types
- Global Scripts
- Presentation
  - User Interfaces
    - CRM Lite (CRM Lite)
      - Default
      - Help
      - Menu
      - Page
- Security
- billingClient (Billing Client)
- billingServer (Billing Server)
- customerManagement (Customer Management)
- Extensions
- orderManagement (Order Management)

General Library Languages Url Mapping Aliases Metadata Replacement Application Page

Group Source: applicationUI.TestDataType

Show Application Page before Login  Classic Select Application

Show	Application	Security	URL	Login UI	Group	Source
Yes (<Default>)	cwf_pm.WorkflowAdministration		/WorkflowAdministration		Security	ConceptWave Metadat...
Yes (<Default>)	cwt_sof.SUSMenu (Old App Menu)		/SUSMenu			Service Orchestration...
Yes (<Default>)	applicationUI CRM Lite (CRM Lite)		/testApp		General	<Local>
Yes (<Default>)	cva_worklist.worklistManagement (Worklis...		/worklistMgr		Security	CWA - Worklist Manag...
Yes (<Default>)	cwa_config.systemConfiguration (System ...		/configApp		System	CWA - Config
Yes (<Default>)	cwa_admin.systemAdministration (System ...		/systemAdministration...		System	CWA - System Admini...
Yes (<Default>)	cwt_pc.catalogMain (Catalog Management ...		/catalogMain5 (<Defau...		Other	ConceptWave Metadata
No (<Default>)	ui_common.applicationPage		/selectedApp (<Default>)			UI - Common
No (<Default>)	ui_common.baseApplication		/baseApplication (<De...			UI - Common
No (<Default>)	ui_common.loginUI		/loginUI (<Default>)			UI - Common
No (<Default>)	ui_common.logoutUI		/logoutUI (<Default>)			UI - Common
Yes (<Default>)	ui_common.sessionErrorUI		/session_error (<Defa...			UI - Common
Yes (<Default>)	cwt_pc.lm.LifeCycleManagement		/LifeCycleManagement...			ConceptWave Metadata
Yes (<Default>)	cwt_pc.plnConfiguration		/plnConfiguration (<D...			ConceptWave Metadata
Yes (<Default>)	cwt_sof.softMenu (Service Orchestration)		/softMenu (<Default>)			Service Orchestration ...
Yes (<Default>)	cva_security.securityManagement (User Pr...		/securityManagement...			CWA - Security
Yes (<Default>)	cwt_pc.catalogMainApp (Catalog Managem...		/catalogMain (<Default...			ConceptWave Metadata

## User Interface General Properties

The User Interface's **General** tab allows you to view and change a number of general properties.

**Note:** You may see more fields on this tab depending upon which object is used to extend the User Interface.

The screenshot shows the 'General' tab of a User Interface configuration dialog. The 'Name' field is set to 'CRMLite'. The 'Label' field contains 'CRM Lite'. The 'Description' field is empty. The 'Extends' field shows 'com.conceptwave.system.Application'. The 'Overrides' field shows '<NONE>'. The 'Help' field is empty. The 'Worklist' field shows 'cwf\_pm.BaseWorklistFinder (Worklist)'. The 'Timer duration (minutes)' field is set to '0 (<Inherited>)'. The 'Url Mapping' field is set to '/CRMLite'. At the bottom, there are several checkboxes: 'Clear validation messages on editing', 'Show Errors Inline', 'Not Valid for Application Selection', and 'Block UI on page update from server'.

The following table describes the fields for the User Interface's General tab:

Field	Description
<b>Name</b>	Mandatory. Name of the metadata object (used in scripts; must conform to JavaScript naming conventions). The name is always <i>User Interface</i> unless you have created a top-level User Interface. In this case, the name is user-specified.
<b>Label</b>	Mandatory. Display label for the User Interface.
<b>Private</b>	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Virtual</b>	If checked, it provides the ability to create methods in template metadata that does not have implementation, but can be used in template script and be bound to UI elements. The implementation is done in user metadata (see <a href="#">Virtual Field</a> for more information).
<b>Final</b>	Metadata elements can be marked as final, which includes any Boolean final property that is available on any metadata element. Additionally, any metadata element that has the <b>Final</b> property set to true can neither be extended, nor overridden.
<b>Description</b>	Description of the metadata object for documentation.
<b>Extends</b>	Displays the base User Interface that is extended by this User Interface. A new included User Interface always extends from system default <code>com.conceptwave.system.&lt;UIObject of parent&gt;</code> . If a metadata object, that contains this User Interface, is a derived object, then this User Interface always extends from the base object's User Interface -- it is not possible to change this derivation. For a new top-level User Interface, any base User Interface can be selected to be extended. User Interfaces can be extended only at the top-level using the Wizard.
<b>Overrides</b>	Metadata object to override. Typically used to customize system or library metadata objects.
<b>Help</b>	Help content of the object.
<b>Worklist</b>	This option is only available if the User Interface is of type Application. This method is used to specify a <i>Worklist</i> Finder for the application. The Worklist Finder is either <code>cwf_pm.BaseWorklistFinder</code> or any Finder on which it is based. The default Worklist

(cwf\_pm.BaseWorklistFinder) has no time interval specified, but has an implemented onTimer method. To avoid refreshing this finder, override your worklist finder's onTimer() method and make it empty.

If a logged in user has worklist tasks assigned to him or her, then the application menu displays the number of user worklist tasks icon indicating whether user is *Available* or *Unavailable* (rightmost corner of menu). Clicking the icon beside the task number opens up the Worklist Finder specified on the Application User Interface element. Worklist's on Timer methods refreshes Worklist Finder (in case New Worklist Tasks arrived to the Worklist) and updates application icon, where number of worklist tasks is displayed. Worklist Timer polls Worklist Timer interval from Config, which contains a setting for Worklist Timer Interval (Worklist Timers are special case scenario).

<b>Timer Duration (minutes)</b>	Any User Interface element can have a <i>Timer</i> bound to it. If a timer interval is specified, the timer calls the <i>on Timer</i> method each time an interval elapses.  <b>Note:</b> The system keeps a list of timer objects and searches to remove those that are no longer in use.
<b>URL Mapping</b>	This field denotes the relative URL path that is binded to the application controller (for example, /testApp).
<b>Not Valid for Application Selection</b>	By default, this property is false. When you select this property, the application is not visible on the Select Application page. However, direct URL access to the application through its <b>URL Mapping</b> property is allowed at runtime. See the <a href="#">Metadata Root Application Page - Not Valid for Application Selection</a> for details.
<b>Show Errors Inline</b>	If checked, any validation error message is shown below the field element at runtime (with the validation error icon). If this setting is not selected, only validation error icon is shown and you need to point the mouse over the icon to view the error message.
<b>Block UI on Page Update from Server</b>	By default, the value for this property is false. If checked, this property blocks the UI (such as, typing or any mouse actions) when the page is being updated. A <i>wait</i> cursor remains on the UI while the page is waiting for the server response and updating the field values.

## User Interface Variables

The User Interface **Variables** tab is used to define variables. These variables help facilitate a Form's access to data within models.

The screenshot shows a software interface for managing User Interface Variables. At the top, there are three tabs: **General**, **Variables**, and **Methods**. The **Variables** tab is active, displaying a table of variables:

Name	Type	Methods	Vis	Opt	Edit
confirmObject	Object	<input type="checkbox"/>			
model	Model	<input type="checkbox"/>			
cwSelectCallback	String	<input type="checkbox"/>			
detail	User Interface	<input type="checkbox"/>			
cwShowHover	Boolean	<input type="checkbox"/>			
detailForm	String	<input type="checkbox"/>			
result	Document	<input type="checkbox"/>			
resultForm	String	<input type="checkbox"/>			
disableValidation	Boolean	<input type="checkbox"/>			
cwHasFavorite	Boolean	<input type="checkbox"/>			
chart	Boolean	<input type="checkbox"/>			
header	String	<input type="checkbox"/>			
tableState	String	<input type="checkbox"/>			
views	User Interface	<input type="checkbox"/>			
finder	Finder	<input type="checkbox"/>			
search	User Interface	<input type="checkbox"/>			

Below the table, the **General** tab is selected, showing detailed configuration for the variable **confirmObject**:

- Name:** confirmObject
- Description:** (empty)
- Data type:** com.conceptwave.system.Object
- Type error code:** (empty)
- Mandatory error code:** (empty)
- Element properties:** <Inherited>

A small table below shows element properties:

Name	Value
(empty)	(empty)

The **Variable** tab shows the list of Variables in the User Interface.

Field	Description
<b>Name</b>	Name of the variable (used in scripts; must conform to JavaScript naming conventions).
<b>Type</b>	Metadata object type of this variable.
<b>Methods</b>	Checked if the Variable has associated methods.
<b>Visible</b>	Visible permission of the variable.
<b>Optional</b>	Optional permission of the variable.
<b>Editable</b>	Input field linked to this Variable can not be modified when this field is <i>False</i> .
<b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.	

When you select a variable in the list, all properties display on the **General** tab, while corresponding method information (if any) displays in the **Method** tab. The toolbar provides functionality to add, delete, sort, sort alphabetically, highlight, and copy and paste variables.

For an included (non top-level) User Interface object, when a metadata object is created, by default, variables designed to connect with that object type are created. For example, when a new Document object is added to the metadata, system variables associated to this document are added to the **Variables** tab. The User Interface then exposes the Document to its Forms using the **Model** variable (of type document).

Variables have different length of persistence depending on the type of User Interface:

- At runtime in the User Interface, a variable instance is persisted while a Page is being displayed.
- At runtime, in the Application page, a variable instance is persisted throughout the user session of the application.

As the variable instance is within the User Interface (MVC-Controller) at runtime, the variable persists as long as the Page's lifetime. In an Application Page, for example, the Variable persists throughout the user session of the application.

**Note:** Enumeration type variables cannot be added under the user interface variable list. To display the enumeration as part of a form, the variable must exist under a document variable before that document can be added as a variable of the User Interface. After adding the document as a user interface variable, the enumeration variable is visible through the document variable (for example, doc.enumVar) when attempting to add a variable through the Add Element wizard or through the variable drop-down list of the **Select Field** property.

## Initialize property

Initializing variables often takes this form:

```
this.uil = new ns.uil(null,this);
```

For variables that require initialization when the UI appears, you can create the variable with the proper UI type and set the **Initialize** property, which creates an instance of the variable type.

General Variables Methods

Name	Type	Methods	Vis	Opt	Edit
context	com.conceptwave.system.UserInterface(User Interface)	<input type="checkbox"/>			
content	com.conceptwave.system.UserInterface(User Interface)	<input type="checkbox"/>			
contentForm	com.conceptwave.system.String(String)	<input type="checkbox"/>			
contextForm	com.conceptwave.system.String(String)	<input type="checkbox"/>			
dialog	com.conceptwave.system.Dialog(User Interface)	<input type="checkbox"/>			
title	com.conceptwave.system.String(String)	<input type="checkbox"/>			
cwWindowTitle	com.conceptwave.system.String(String)	<input type="checkbox"/>			
HelpVariable	com.conceptwave.system.String(String)	<input type="checkbox"/>			
worklistImage	com.conceptwave.system.String(String)	<input type="checkbox"/>			
cwDownload	com.conceptwave.system.Download(User Interface)	<input type="checkbox"/>			
banner	com.conceptwave.system.String(String)	<input type="checkbox"/>			
workListFinder	cwf_pm.BaseWorklistFinder.UserInterface(User Interface)	<input type="checkbox"/>			
pollTimeSec	com.conceptwave.system.Decimal(Decimal)	<input type="checkbox"/>			
statusMessage	com.conceptwave.system.String(String)	<input type="checkbox"/>			
app1	InitFlagTest.app1(User Interface)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
app2	InitFlagTest.app2(User Interface)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

General Methods

Name:	app1	<input type="checkbox"/> Array	<input type="checkbox"/> Constant	<input type="checkbox"/> Audit	<input type="checkbox"/> Validate				
Description:									
Data type:	 InitFlagTest.app1				<input checked="" type="checkbox"/> Initialize				
Type error code:			Mandatory error code: <input type="text"/>						
Element properties:	<p>&lt;Inherited&gt;</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr><td colspan="2"> </td></tr> </tbody> </table>					Name	Value		
Name	Value								

## System-Defined Variables

Variable	Description
confirmObject	Name of the variable (used in scripts; must conform to JavaScript naming conventions).
model	The object representing the data for this user interface (that is, Document, Finder, Order).
content	Type user interface. The current object which is being displayed in the content Form Frame.
contentForm	Type String. The name of the Form that belongs to the content object being used to display the content.
context	Used for backward compatibility. Used to display two different objects on the screen.
contextForm	Type String. The name of the Form which belongs to the context object used to display the context.
dialog	Type Dialog. The current dialog user interface being displayed. Used by the Form Frame.
title	The title to display in the Web Browser.
HelpVariables	Type String. The text that displays in the Help dialog.
banner	The image location of the banner that is display on the top of the application menu.
cwDownload	Type Download. Populated when the user performs a download action.
dialogOwner	The user interface that invokes the dialog.
statusMessage	The message that is displayed at the bottom of the page.
workListFinder	Type User Interface. Displays the Work List finder if the user has correct permissions.
workListImage	Type String. The image that displays (available/unavailable) for the user who has permissions for the work list.
Editable	Optional permission of the variable.



## User Interface Methods

The User Interface **Methods** tab is used to define methods that facilitate business logic, such as JavaScript functions, to forms.

This screenshot shows the 'Methods' tab of a configuration interface for a 'UserInterface' object. The left pane displays a tree structure with 'UserInterface' selected, containing 'addItems', 'submitOrder', and 'isNewOrder'. The right pane contains fields for defining a new method:

- Name:** onInit
- Description:** (empty)
- Action Category:** (empty)
- Parameters:** A table with columns 'Name', 'Type', and 'Description'. It currently has one row with empty fields.
- Script:** A code editor containing the following JavaScript function:

```
function onInit() {
    this.cw$super_onInit();
    this.navbread = new Breadcrumbs.navbread();
}
```
- Return:** com.conceptwave.system.Void
- Return Description:** (empty)

On the far right of the interface are several small icons for navigating and managing the list of methods.

When you select the **Type** radio button, it means that the method can be invoked directly through the script and does not require an instance object.

To show all base methods, click the **Show base methods** (  ) button. Click the button again to hide all base methods. Base methods appear in blue. Any method highlighted in blue font indicates that you can override it. Methods in black font cannot be overridden.

If you have a long list of methods in the left pane, you can press the first letter of the method you are looking for to jump to the list of methods beginning with that letter. In this example, pressing the **I** key highlights the first instance of a method that begins with the letter **I**. The method displays in the right pane.

This screenshot shows the 'Methods' tab of a configuration interface for a 'UserInterface' object. The left pane displays a tree structure with 'UserInterface' selected, containing 'addItems', 'submitOrder', and 'isNewOrder'. The right pane contains fields for defining a new method:

- Name:** isNewOrder
- Description:** (empty)
- Reference** (checkbox)

On the far right of the interface are several small icons for navigating and managing the list of methods.

There are three types of methods that can be created and existing system-level methods that can be overridden:

Method Type	Description
 Script	Generic user-defined JavaScript function that can be defined in User Interface and explicitly invoked by other Methods or Form Elements.

 Permission	Permission Method that can be associated to properties of Form Elements to determine participant's access of the Form Element such as <i>Visible</i> , <i>Optional</i> , and <i>Editable</i> . See the <a href="#">Permissions</a> section for details.
 User Action	User Action Method that implements the response of user actions, such as clicking a Button or Image, by returning an Object presented in a specified Form to the user. See the <a href="#">User Action Method</a> section for details.

You can create multiple methods of the same method type by using unique method names (for example, Permission1, Permission2). However, for methods that are to be overridden, you can only define one Method per base Method. This is applicable to both system-predefined scripts from `com.conceptwave.system.UserInterface`) and any additional Methods that are defined in non-system-based, base User Interface.

For scripts that are already overridden, they appear in the **Methods List** pane and no longer available using the right-click menu.

If this User Interface extends from another (non-system based) User Interface, and the base User Interface has additional methods defined in it, these additional methods are not displayed in the Methods tab of this User Interface. However, the additional methods are available to be overridden by right-clicking the root node in the Method List pane.

The method created in **Method List** pane can be right-clicked, with various commands available. For more information see, [Scripting Interface](#).

Depending upon which User Interface metadata you are configuring, there are many different methods that can be overridden. A list specific to the User Interface displays when you right-click the method name that appears in the **Navigation List** pane. The following are examples of system-defined method that can be overridden.

Script Name	Action Type	Parameters	Return Type	Description
afterUserAction	After a user action occurs	None.	<code>com.conceptwave.system.Void</code>	On click script to set the function that is returned when a user action occurs. It is triggered when you perform an action specified in the script.
cwchildDialogClose	Close current dialog	None.	<code>com.conceptwave.system.Void</code>	Sets the dialog variable to null causing the current dialog to close.
cwDialogClose	Call to close current dialog	None.	<code>com.conceptwave.system.Void</code>	This method calls the cwChildDialogClose.
dynamicTimerDuration	Timer refresh duration	None.	<code>com.conceptwave.system.Void</code>	You can configure the <a href="#">user interface timer refresh script duration</a> by overriding this script in your metadata, and returning either an integer or a script that returns an integer to configure the time duration.
generateHelp	Generate help	None.	<code>com.conceptwave.system.Void</code>	This method displays the available help of the current object in the dialog.
hasWorklist	Worklist	None.	<code>com.conceptwave.system.Void</code>	This method checks if the worklist is created for a user based on permissions.
onAutoSave	Automatically save	True or False.	<code>com.conceptwave.system.Boolean</code>	The method <code>onAutoSave</code> is called on auto save notification of the user action.
onInit	Initialization	None.	<code>com.conceptwave.system.Void</code>	User Interface Initialization script to set initial values for Variables in the User Interface. It is triggered when the User Interface is created.
onTimer	Worklist timers	None.	<code>com.conceptwave.system.Void</code>	The default Worklist ( <code>cwf_pm.BaseWorklistFinder</code> ) has no time interval specified, but has an implemented <code>onTimer</code> method. A worklist's on Timer method simply refreshes Worklist Finder (in case New Worklist Tasks arrived to the Worklist) and updates application icon, where number of worklist tasks is displayed. Worklist Timer polls Worklist Timer interval from Config, which contains a setting for Worklist Timer Interval. If your metadata overrides the default <code>BaseWorklistFinder</code> , we can also specify worklist interval on the overriding finder's user interface. In this case, system uses the timer interval from the metadata rather the one polled out from Config.

				Any User Interface element can have a timer bound to it. If the timer interval is specified, the timer calls the <i>onTimer</i> method each time interval elapses.
onScreenReaderMode	Screen reader mode	True or False.	<i>com.conceptwave.system.Boolean</i>	In the past, screen reader mode could only be set at the configuration level. You can allow your application to set this mode on-demand. At runtime, you can use the <i>onScreenReaderMode()</i> API under the application user interface. If this method returns null, the value is taken from the system configuration using the <i>screenreader</i> configuration variable.
onValidate	Validate	None.	<i>com.conceptwave.system.Void</i>	The method is called on validate notifications and validates all fields and the UserInterface model, if it exists. It returns ValidationErrorsList object.
openWorklist	Open worklist	None.	<i>com.conceptwave.system.Void</i>	This method opens the worklist when user permissions match permissions for the Worklist.
permWorklist	Worklist permissions	Yes.	<i>com.conceptwave.system.Boolean</i>	This method is returns the worklist when the permissions set are satisfied.
statusMessagePerm	Permissions message	Yes.	<i>com.conceptwave.system.Boolean</i>	This method is a permission method used to indicate visibility of the status message (generally displayed at the bottom of the page).

**Note:** To access to the current application controller, call the *getCurrentApplication()* method.

## User Action Method

---

The User Action Method wizard enables you to configure settings or a script to invoke a user action in a Form. Although you can also create a User Action Method using a *generic method* script, the User Action Method is a convenient and advanced way of configuring a user action. For example, by creating a User Action Method script, you can create a dialog box and display a specified object.

The User Action Method, consists of an object, and depending upon the object type, a form. You need to specify an object as either a user interface or a model. You can define the object as either a top-level or embedded User Interface, neither of which requires you to define a Form. In the case of the embedded user interface, the model (Finder, Order, User Interface, Navigation Tree or Document) is created by the default constructor. When the object is defined as a Model, you need to specify a Form that is part of that User Interface.

Although any script can set a trigger, User Action Methods is an advanced way of creating a trigger. The User Action Method can display an object as a dialog or display a message (error or confirmation) before performing the Method Action.

### Configure a User Action Method

There are three ways to configure the User Action Method:

- Use the Object and Form list menus
- Use a User Action Method script
- Use a script

The **Object** and **Form** list menus make the configuration of the User Action Method a quick task. Select an object (a User Interface or a Model) and where applicable, select a Form (additional properties are available from the Methods tab).

Alternatively, you can configure the User Action Method using more advanced options. The script functions independently from the list menu options. When you initialize the User Action Method with a script, the script calls a constructor that creates an object. The User Action Method script can return a value (a Document, Finder, Order, User Interface, Navigation Tree) and that returned object is set as the content object at the Form level. In some cases you may not want to display the returned object as the content object of a Form (or a Form element such as a Form Frame).

For example, if the user action is set as a button click, and there is no content to return, you would not setup a Form. In other cases you may want the returned object set as content to display in a Form Frame. The Form Frame binds to a Form variable. The User Action Method instantiates corresponding variables and forwards them to the next action within the model.

**Note:** User Action Methods can be called from another script.

To add a User Action Method, complete these steps:

1. In the Navigation pane, click the User Interface object that you want to open.
2. Click the Methods tab.
3. Right-click the User Interface node and select **New User Action**.
4. In the New User Action dialog, configure the settings and then click **Finish**.
5. The User Action Method is added to the Methods list and the properties dialog displays.

### Configure the User Action Method Properties

After you have configured the User Action Method using the wizard, you can configure more properties using the Methods tab (see the descriptions that follow). When you are using a script, and you have multiple parameter in the script, you must ensure that the object is the last parameter in the script.

**Name:** toggleExpand

**Description:**

**Action Category:** HighPerformanceLoad

**Parameters:**

Name	Type

**Script:**   
The code above is a JavaScript function named 'toggleExpand'. It contains a single conditional statement that toggles the 'expandedView' property of the 'act.model' object. The condition checks if the current state is 'VOICE' (determined by the 'para, en, hacia, contra, según, 'VOICE')' part of the script). If it is, it sets the 'expandedView' to the negation of its current value (using '!' and 'this.product.model.expandedView'). The entire function is enclosed in parentheses and a closing brace.

**Return:**

**Object:** <NONE>

**Form:** <NONE>

**Confirmation:** <NONE>

Allow invalid object  Autosave  Show in popup  Validate

The Methods tab contains the following fields:

Field	Description
<b>Name</b>	Name of the Method. Must be unique within the User Interface.
<b>Description</b>	Descriptive text for documentation purposes.
<b>Action Category</b>	String value that enables User Action Methods to be queued to avoid overloading the system while simultaneously executing server intensive processes. <a href="#">Action Categories</a> are defined in the Configuration application.
<b>Parameters</b>	Adds parameters in your script. The last parameter must be defined as the object. For example, if the User Action Method is invoked by the User Interface (object), the only parameter that is used is this object.
<b>Script</b>	Any method, including Permission Methods, is considered to be a script Method when it includes a custom JavaScript that is invoked as an action. The User Action script can return a value (a Document, Finder, Order, User Interface, Navigation Tree) and that returned object is set as the content object at the page or pop-up dialog (returning a value overrides the object property).
<b>Return</b>	The object type to be returned by the script.
<b>Object</b>	The object of this type is instantiated when this Method is invoked.
<b>Form</b>	The Form to present to user when the object is instantiated. Choices are available and limited only to Forms of the selected object.
<b>Confirmation</b>	Configure a confirmation message that displays before the performing the user action.
<b>*Allow invalid object</b>	This parameter works with the Validate parameter. The Validate parameter determines whether user interface validation will be performed on the whole page and content or dialog box. If validation fails and this parameter is checked, then the user action script defined here will continue. Otherwise, if validation fails and this parameter is unchecked, the user action method will be interrupted quietly.
<b>Autosave</b>	Prompt the user to save the Document or Order if it has unsaved changes.
<b>Show in popup</b>	If checked, opens in a pop-up window.
<b>*Validate</b>	If checked, the onValidate method is called on the parent controller window (for example, the whole page or the dialog) when called as a menu Click action or a script method called from another script method. The validation on the UserInterface object will

be performed *before* the user action is invoked. If the onValidate method returns a False (not valid), then the user action will be quietly be interrupted without an exception being thrown. However, if the **Allow invalid object** is checked, then the user action script will continue with a UI that is not valid.

The following information pertains to the **Validate** and **Allow invalid object** checkboxes:

- If checked, the **Validate** checkbox calls the onValidate method for the "content" controller starting with the parent UI, and then to the children. The onValidate method checks standard validation rules. If the onValidate method returns a False (that is, the UI is not valid) and the **Allow invalid objects** parameter is checked, the User Action method is invoked. However, if the onValidate method returns a False (that is, the UI is not valid) and the **Allow invalid objects** parameter is *not* checked, the User Action is quietly interrupted without throwing an exception. For the system to display an exception or a custom message, the **Validation** parameter should not be checked and the onValidate() method should be called in the User Action script:

```
var parentWindow = this.getParentWindow();
if(!parentWindow.onValidate()){
    Global.throwException("Error", this);
}
```

- In the client UI, to present documents that have been previously generated and stored in the database, you can create a user action. Then, in the script panel, write the following:

```
var download = new com.conceptwave.system.DownloadBinary(null, this);
download.fileName = "/cwf/order.pdf";           //any name
download.mimeType = "application/pdf";          // mime type of the binary file
download.binaryFile = this.onLoadPdfFromDB(); // script which will read the binary file from the database
Global.doDownload(download);
```

## User Action Method: Dialog and Script Parameters

The user action script can have different parameters, depending on the action properties and the context that this action is invoked with:

- If the user action has the **Object** property specified, this object is created just before the action script invoked and being passed to the script as the *last* argument.
- If the user action under User Interface has the **Show in popup** flag selected, it means that the object specified as the **Object** property or returned from script needs to be displayed in the popup dialog.  
The Dialog object is also created before the action using the specified dialog properties and passed to the user action as the *last* argument. Then, the object is passed as the **second last** argument.
- There are also some special user actions, such as onRefClick and cwOnDetailView methods, that have more than two arguments passed to the script.

### Create a Dialog Object

When a user action has the **Show in popup** property selected, the Dialog object is created and passed to the action script. The cwCreateDialog method of the system com.conceptwave.system.UserInterface contains this logic and has the following script parameters:

Script Parameter	Description
actionName	This parameter denotes the name of the popup action ID.
dialogMDName	This parameter contains the full metadata name of the Dialog object specified in the user action. By default, it is set to com.conceptwave.system.Dialog.
width	This parameter indicates the dialog width, which is specified in the user action properties.
height	This parameter represents the dialog height, which is specified in the user action properties.
x	This parameter denotes the dialog's X position, which is specified in the user action properties.
y	This parameter contains the dialog's Y position, which is specified in the user action properties.

By default, the cwCreateDialog script calls the cwCreateDialogDefault method of the current application:

```
if (theApp && theApp.hasMethod("cwCreateDialogDefault")) {  
    return theApp.cwCreateDialogDefault(this, actionName, dialogMDName,  
    width, height, x, y);  
}
```

This call allows customizing the Dialog object for each user action, user interface, or application. If the script returns either null or an object that is not instance of Dialog, then the user action creates a dialog using the old hard-coded logic.

As an example, the cwCreateDialogDefault method's default implementation of the com.conceptwave.system.Application has some customization for both the onRefClick and cwOnDetailView methods. These actions are defined in com.conceptwave.system.UserInterface. However, depending on the context, it creates different Dialog objects. If the action owner extends either ui\_common.baseForm or ui\_common.baseComponent, it creates ui\_common.dialogExtend instead of the old com.conceptwave.system.Dialog.

### User Action Parameters

When initializing user action arguments, the following rules are applied:

1. If the user action has the **Object** property specified and is *not a popup*:
  - a. If the script does not have any parameters defined, the object is still passed to the script. To access it, use the **arguments** local variable:

```
function userAction () {
```

```
    var obj = arguments[0];
    ...
```

- b. If the script has one parameter defined, no matter what the parameter type is that is defined in the metadata, the first parameter is initialized with the object:

```
function userAction (param1) {
    var obj = param1; // or arguments[0]
    ...
```

- c. If more than 1 parameter is specified, it attempts to find the position of the object parameters by looking at the parameter type, which should be equal to the Object type:

```
function userAction (param1, param2) {
    var obj = param2; // or arguments[1] if param2 has type of Object
    ...
```

If the parameter is not found, the object is passed as the *last* parameter (param2).

## 2. For **Show in popup** actions:

- a. If the script does not have any parameters defined, the object and the dialog are passed to the script. To access them, use **arguments** local variable:

```
function userAction () {
    var obj = arguments[0];
    var dialog = arguments[1];
    ...
```

- b. If the script has 1 argument defined, it depends on what the parameter type is. If the parameter is the type of the Object, the argument is initialized *with* the object and the dialog can be accessed using the **arguments** variable:

```
function userAction (param1) {
    var obj = param1; // or arguments[0]
    var dialog = arguments[1];
    ...
```

If the parameter is the type of the dialog or or extends it, the argument is initialized with the dialog and the object can be accessed using the **arguments** variable:

```
function userAction (param1) {
    var dialog = param1 // or arguments[0]
    var obj = arguments[1];
    ...
```

- c. If the script has two or more arguments defined, it first attempts to define both the object and dialog parameter positions by looking at the parameters type. The object parameter should have same data type. The dialog parameter should have either com.conceptwave.system.Dialog or and extended type.

**Note:** There could be different situations where one or the other position is not found:

- If the object parameter is found, but the dialog parameter is not found, the object parameter is initialized with the object and the dialog is passed as the last parameter. If the object parameter is the last one, the dialog is passed as the *second last*.
- If the dialog parameter is found, but the object parameter is not found, the dialog parameter is initialized with the dialog and the object is passed as the parameter *before* the dialog parameter. Although, if the dialog parameter is the *first* one, the object would be the *second*.
- If none of the positions are found, the last parameter is initialized with the dialog and the second last parameter contains the object.

3. When a user action is used as either a reference or translation field action (for example, the system `onRefClick` or `cwOnDetailView` actions), the action arguments have a predefined order and the rules defined previously are not applied to them.

Parameter	Description
leafName	This parameter represents the reference variable's name.
isEditable	This parameter is true if you can select or change a referenced value.
targetObj	This parameter is an object that denotes the variable's owner.
refDataType	This parameter denotes the reference data type. If the variable has a non-reference type, this data type is used to find the reference finder.
dialog	This parameter represents the dialog object.

## User Interface Validation Example

This section shows an example of a User Interface Validation functionality for an Array and UserInterface Object. In this example, a Document is created with mandatory leaves that are validated by calling the onValidate script by using a UserInterface button.

### Create a Document

A Document is created with mandatory input fields that are mapped to a user interface and a button that calls the onValidate script.

#### Leaf Creation

Document variables have been defined as mandatory for user input by assigning False to the **Optional** field.

The screenshot shows the AVM Metadata interface. On the left, there is a tree view of the project structure:

- AVM Metadata
- validate
- Data Dictionaries
  - Data Types
    - Integer
  - Documents
    - TestDoc
      - UserInterface
      - Default
      - Menu
  - Presentation

On the right, there are two main panels:

  - Variables** panel:

Name	Type	Methods	Vis	Opt	Edit
name	String, 64			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
number	Integer, 3			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
textField	String, 64			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

The columns are: Name, Type, Methods, Vis, Opt, Edit. The 'Opt' column for all three rows has a red border around it, indicating they are mandatory.
  - Methods** panel:

Name	Type	Constant	Audit
name	cwf.String64	<input type="checkbox"/>	<input type="checkbox"/>

Below this, there are sections for 'Type error code', 'Mandatory error code', and 'Element properties'. Under 'Element properties', there is a table:

Name	Value
Display Format	
Optional	FALSE
Editable	
Visible	
Label	String64

The 'Optional' row in this table also has a red border around it, indicating its value is FALSE.

### Create Validation Methods

The leaf variables have been assigned validation methods of various severities.

AVM Metadata

- validate
  - Data Dictionaries
    - Data Types
    - Documents
      - TestDoc
        - UserInterface
          - Default
          - Menu
- Presentation

Name	Type	Methods	Vis	Opt	Edit
name	String, 64	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
number	Integer, 3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
textField	String, 64	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

General   Methods	
<ul style="list-style-type: none"> <li>TestDoc           <ul style="list-style-type: none"> <li>textFieldInfo</li> <li>textFieldError</li> <li>TextFieldWarn</li> </ul> </li> </ul>	<p><b>Name:</b> textFieldInfo <input type="checkbox"/> Is private</p> <p><b>Description:</b></p> <p><b>Script:</b></p> <pre>function cwOnDocValidRule(document) { // Validation rule. Used in generate     return document.textField == 'a'; }</pre> <p><b>Return:</b> com.conceptwave.system.Boolean</p> <p><b>Execute Permission:</b> &lt;NONE&gt;</p> <p><b>Severity:</b> information</p> <p><b>Message:</b> Info message</p> <p><b>Message Code:</b></p>

### Design User Interface

A user interface is designed for the document with a validation button.

Name	Value
Action Auditor	
Can Sort	<input checked="" type="checkbox"/>
Cell Alignment	
Cell Column Span	
Click method	ValidateButtonAction
Disabled	
Height	
Icon	
Icon Orientation	
Label	Validate
Name	Button
Show Label	<input checked="" type="checkbox"/>
Start Row	<input type="checkbox"/>
Style	
Tooltip	
Visible	
Width	

#### Define Validation Method for button element

In the UserInterface shown above, a **Validate** button displays, which has a Click Method property associated with the ValidateButtonAction method. The ValidateButtonAction method is a user-defined method that calls the onValidate script:

```

function ValidateButtonAction() { // Metadata type method. Can be called by scripts.
    this.onValidate();
    if (this.isValid)
        Global.showUserMessage("Document is valid. Validation status: " + this.validationStatus);
    else
        Global.showUserMessage("Document is invalid. Validation status: " + this.validationStatus);
}

```

#### Define the Presentation

At the Presentation layer, a UserInterface object is defined containing a variable (*testDocUI*) that points to the document object (*validate.TestDoc.UserInterface*) created in the

steps above and flagged for the **Validate** checkbox. The second variable, *testDocUIArray*, also points to the same document object, but this variable is an Array with the **Array** checkbox flagged.

Name	Type	Methods	Vis	Opt	Edit
confirmObject	Object				
model	Model				
testDocUI	User Interface				
testDocUIArray	User Interface				

**General** **Methods**

Name: *testDocUI*  Array  Constant  Audit  Validate

Data type: validate.TestDoc.UserInterface

Type error code:  Mandatory error code:

Element properties: <Inherited>

## Initialization

At runtime, the *testDocUI* user interface variable is initialized to the user interface (*validate.TestDoc.UserInterface*) by the *onInit* method. The *onInit* method of the *TestUI* object assigns the *TestDocUI* variable to the document user interface object (*validate.TestDoc.UserInterface*).

Name: onInit  Is private

Description:

Action Category:

Parameters:

Script:

```
function onInit() // Metadata type method. Can be called by scripts.
this.cw$super.onInit();
this.testDocUI = new validate.TestDoc.UserInterface(null, this);
this.testDocUIArray = new Array();
for (var i = 0; i < 3; i++) {
    this.testDocUIArray[i] = new validate.TestDoc.UserInterface(null, this);
}
```

Return: com.conceptwave.system.Object

## FormFrame Variable

For the FormFrame element of the *TestUI* Default form, the *testDocUI* variable is assigned to the **Variable** property:

The screenshot shows the AVM Metadata interface. On the left, the tree view displays the project structure under 'validate'. Under 'TestDoc', there is a 'UserInterface' folder containing 'Default' and 'TestUI' sub-folders, each with a 'Menu' item. The 'Default' folder is expanded, showing a 'VerticalLayout' containing a 'Label', a 'FormFrame' (selected), another 'Label', and a 'HorizontalLayout' containing an 'Iterator' and a 'FormFrame1'. The 'TestUI' folder also contains a 'Default' folder with a 'Menu' item.

The main panel shows the 'Single variable:' section for 'Validate TestUI'. The 'Default' form is selected. The properties grid on the right lists the following properties:

Name	Value
Cell Alignment	
External URL	
Form	testDocUI.Forms.Default
Height	
Name	FormFrame
On Enter	
Overflow	visible
Show Resize Bar	<input type="checkbox"/>
Style	
Tooltip	
Unmanaged	<input type="checkbox"/>
Variable	testDocUI
Visible	
Width	

The second FormFrame element of the *TestUI* Default form, has the *testDocUIArray* (array variable) assigned to the Form property and the associated Iterator variable is associated to the *testDocUIArray*. Below are the properties details for the FormFrame and Iterator elements:

The screenshot shows two side-by-side property grids. The left grid is for the 'FormFrame' element, and the right grid is for the 'Iterator' element.

**FormFrame Properties:**

Name	Value
Cell Alignment	
External URL	
Form	testDocUIArray.Forms.Default
Height	
Name	FormFrame1
On Enter	
Overflow	
Show Resize Bar	<input type="checkbox"/>
Style	
Tooltip	
Unmanaged	<input type="checkbox"/>
Variable	
Visible	
Width	

**Iterator Properties:**

Name	Value
Name	Iterator
Variable	testDocUIArray
Visible	

## Runtime

At runtime, the TestDoc document appears as a single instance and an array:

**Single variable:**

Name \*

Warning for number = 3 \*

Info message on textField = 'a'

**Array of variables:**

Name \*

Name \*

Name \*

Warning for number = 3 \*

Warning for number = 3 \*

Warning for number = 3 \*

Info message on textField = 'a'

Info message on textField = 'a'

Info message on textField = 'a'

## User Interface Validation

The base com.conceptwave.system.UserInterface object offers an onValidate method that will check certain standard validations like field length, mandatory fields, minimum and maximum values, etc. on UserInterface Objects. This onValidate method is called automatically when a UserInterface variable has its **Validate** checkbox flagged. It can also be called when an onValidate method is called from another method (like a UserAction Method) or via a variable's Validate method. The onValidate script will return a True or False value for the isValid property: False if there is an error and True if there are no errors (warnings are still valid) and it also returns a validationStatus (error, warning, or info). An example of the implementation of the Validation feature is documented in the [User Interface Validation Example](#).

When the **Validate** checkbox is flagged on a [User Action method](#), the onValidate method checks the validation of the page or dialog box (parent controller) and then the subsequent children. If the onValidate method returns a False (not valid), then this isValid property is propagated to the children. To find the parent object, the onValidate method will access the this.parent value (up to the com.conceptwave.system.Window parent). If a "parent" is not set correctly resulting in the parent window not being found, then the system will call the application UI (com.conceptwave.system.Application). Below is a listing of the UserInterface Objects and the validations that are performed:

UI Object	Validation
Document Finder	Validation is performed on the "detail" variable controller and "model".
Document	Validation is performed on the "model" variable controller.
Navigation Tree	Validation is performed on the "model" variable controller and selected "detail".
Order	Validation is performed on the "model" variable of type "order".

There are several methods available to make use of and call the User Interface Validation functionality:

- [User Action Method](#): When the **Validate** checkbox is set for any user action. The validation is invoked *before* the action is invoked. Please refer to the [User Action Method](#) section for more details.
- [Variables](#): Validate Method assigned to a variable and a **Validate** checkbox on the variable.
- [OrderUserInterface, NavigationTreeWithTabs and NavigationTree](#): Validation is performed on the current node or tab, whenever there is a selection change on either the tree node or tab.
- [onValidate method call](#): An onValidate method can be called when needed to validate a particular UserInterface controller.

The onValidate method checks for [standard validation errors](#) and displays standard and user-definable validation icons displaying [tab errors](#) when the UserInterface on the TabFrames are not valid.

## Variables

Variables can be checked against standard validations by checking the **Validate** checkbox on the variable. User-specific validations can also be called by creating a Validation method on the variable.

### Validate Checkbox

Within the User Interface **Variables** tab, it is possible to set a Validate property for each variable (including other UserInterfaces, documents, arrays, data types, etc.). This **Validate** checkbox is set by default for some of the system Window UI variables like the "Model", "Content" and "Detail". When the onValidate method is called, it will iterate all variables with the Validate property selected to validate them.

Name	Type	Methods	Vis	Opt	Edit
confirmObject	Object	<input type="checkbox"/>			
model	Model	<input type="checkbox"/>			
content	User Interface	<input type="checkbox"/>			
contentForm	String	<input type="checkbox"/>			
context	User Interface	<input type="checkbox"/>			
contextForm	String	<input type="checkbox"/>			
dialog	User Interface	<input type="checkbox"/>			
owner	User Interface	<input type="checkbox"/>			
title	String	<input type="checkbox"/>			
HelpVariable	String	<input type="checkbox"/>			
banner	String	<input type="checkbox"/>			
cwDownload	User Interface	<input type="checkbox"/>			
dialogOwner	User Interface	<input type="checkbox"/>			
millTimeSec	Decimal	<input type="checkbox"/>			

Validate

↓

**General** **Methods**

**Name:**   Array  Constant  Audit  Validate

**Data type:**

**Type error code:**  **Mandatory error code:**

**Element properties:**

Name	Value

### Validation Scripts

It is possible to assign a validation script that is called at runtime at the document and data type levels. The Validation script returns an Error, Warning, or Information severity on the validation and they are only called when the document is invoked. The following is an example of a Validation Script:

The screenshot shows the AVM Metadata interface. On the left, there's a navigation tree titled 'AVM Metadata' with a 'validate' node expanded. Under 'validate', there are 'Data Dictionaries', 'Documents' (with 'TestDoc' selected), 'Userinterface', 'Default', 'Menu', and 'Presentation'. The main window has four tabs: 'General', 'Variables', 'Methods', and 'Mapping'. The 'General' tab is active, showing a table with columns 'Name' and 'Type'. The table contains three rows: 'name' (String, 64), 'number' (Integer, 3), and 'textField' (String, 64). To the right of the table is a grid for 'Methods' with columns 'Methods', 'Vis', 'Opt', and 'Edit'. Below the table is a large text area for 'Script' containing the following code:

```
function cwOnDocValidRule(document) { // Validation rule. Used in generate
    return document.textField == 'a';
}
```

The 'Methods' tab is also visible, showing a list of methods under 'TestDoc': 'textFieldInfo', 'textFieldError', and 'textFieldWarn'. Each method has fields for 'Name', 'Description', 'Script', 'Return', 'Execute Permission', 'Severity', 'Message', and 'Message Code'.

Field	Description
Name	The object of this type is instantiated when this method is invoked.
Description	This is a description of the method.
Script	Any method, including Permission methods, is considered to be a script method when it includes a custom JavaScript that is invoked as an action.
Return	The object type to be returned by the script.
Execute Permission	This is a list of permission methods that return a <i>True</i> or <i>False</i> . When a <i>True</i> is returned, the validation method is run and if a <i>False</i> is returned, then this validation method is not run.
Severity	This is the severity of the error resulting from the script (Information, Error or Warning). The validationStatus parameter is set to this value.
Message	This is the message (error, warning or information) to be displayed as a hover over the icon.
Message Code	This is the message code that will be used in place of the message. The <a href="#">message codes</a> exist in the appRC_xx.xml files and can be modified through the Resources folder.

## Navigation Tree, NavigationTreeWithTabs, and OrderUserInterface

### Navigation Tree and Navigation Tree Tabs

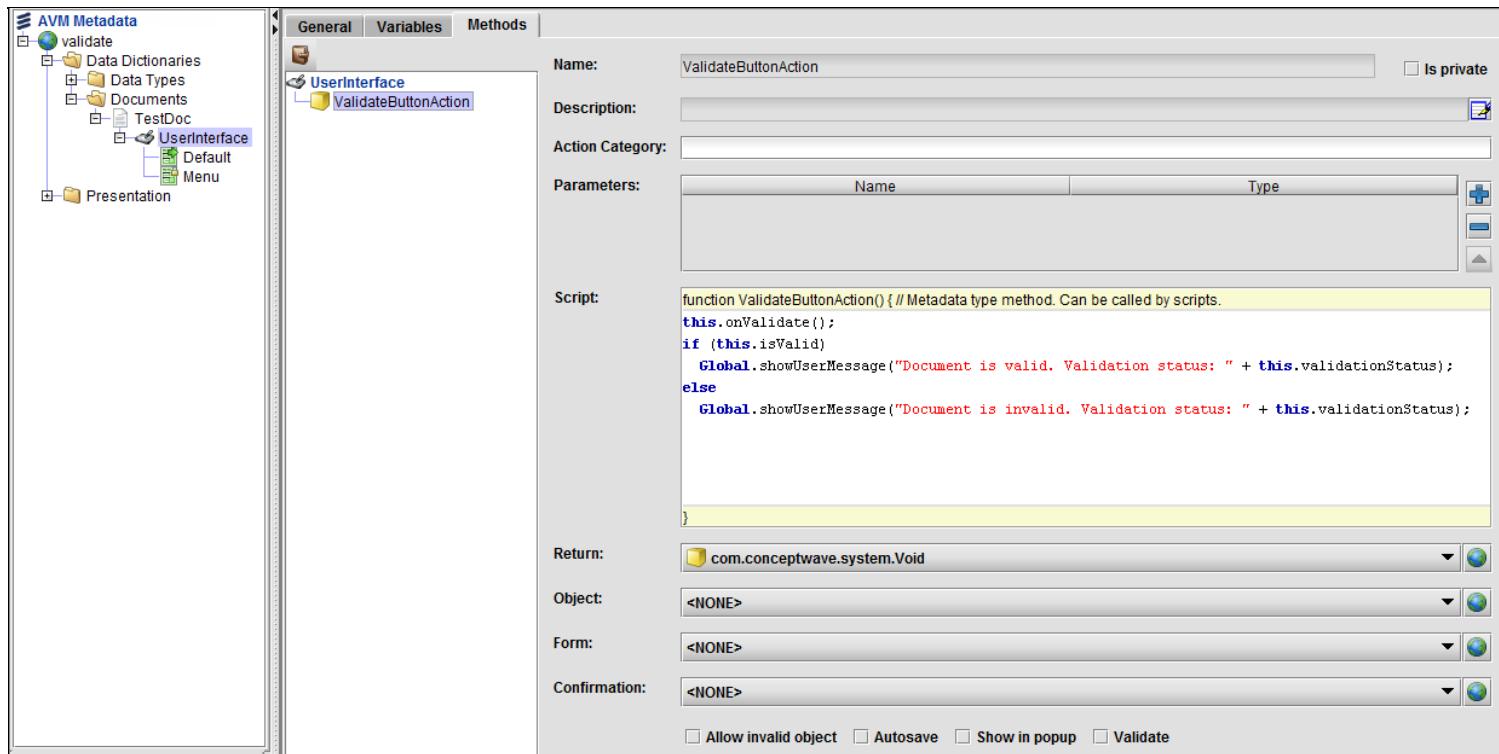
Validation is performed automatically when changing a selection in the Navigation Tree (either the tree node or tab). The `onValidate` method is called for the previously selected node (`com.conceptwave.systemTreeNode.onValidate`), which will validate the `modelUI` (the node UI detail) and the `model` of the tree node. If there are tab elements displayed, then the `onValidate` method will also validate all tabs. When navigating between tabs within a Navigation Tree, `onValidate` method will only validate the previously selected tab.

## OrderUserInterface

Similar to the Navigation Tree, the onValidate method will validate the tree nodes and tabs. OnValidate is called when there is a change in a tree node or a tab selection. When the onValidate returns a False (not valid), a confirmation dialog box appears with a warning message indicating that there are validation errors and asking users if they would like to continue. This functionality and warning messages can be modified by overwriting the metadata scripts.

## onValidate Call

OnValidate method can be called from another method. The onValidate method is called automatically when a variable is selected for Validation and the user navigates away from the form. However, the validation script can also be called with from a User Action Method without exiting the form by associating a user action method to a button. When the button is triggered by the user at runtime, the system would then trigger the onValidate method. The following is an example of a User Action Method script that calls the onValidate method.



## Standard Validation Errors

The onValidate method will check for certain validation errors including Nullable, Optional, length, max value, and minimum value.

These fields are available for validation in the Data Type and Document leaves:

### Data Type

It is possible to define a data type if the item is Nullable, Optional, a specific Length, Minimum Value, and Maximum Value. These are the fields that can be validated at runtime.

General    Methods    Enumeration

Name:	Integer	<input type="checkbox"/> Private	<input type="checkbox"/> Restricted	<input type="checkbox"/> Deprecated																		
Label:	Integer	...																				
Description:	<input type="button" value="Edit"/>																					
Help:	<input type="button" value="Edit"/>																					
Extends:	cwf.Integer3	<input type="button" value="Edit"/>	<input checked="" type="checkbox"/> Nullable	<input type="checkbox"/> Encrypt																		
Overrides:	<NONE>	<input type="button" value="Edit"/>																				
Length:	3 (<Inherited>)																					
Min:	(<Inherited>)																					
Max:	(<Inherited>)																					
Default value:																						
XML name:		Pattern:																				
Text format:		Text length:																				
Element properties:	<input type="button" value="Text Field"/> <table border="1"> <thead> <tr> <th></th> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Display Format</td> <td></td> <td></td> </tr> <tr> <td>Optional</td> <td>Optional</td> <td>FALSE</td> </tr> <tr> <td>Editable</td> <td></td> <td></td> </tr> <tr> <td>Visible</td> <td></td> <td></td> </tr> <tr> <td>Label</td> <td></td> <td></td> </tr> </tbody> </table>					Name	Value	Display Format			Optional	Optional	FALSE	Editable			Visible			Label		
	Name	Value																				
Display Format																						
Optional	Optional	FALSE																				
Editable																						
Visible																						
Label																						

It is also possible to define a [validation script](#) for each data type that is invoked when associated with the leaves of a document and that document is invoked.

## Documents

When defining the Document leaf variables, it is possible to specify whether the leaf is Optional. Setting the **Optional** field to False indicates that the leaf is mandatory for user input.

The screenshot shows the AVM Metadata interface with the 'name' field selected. The 'Edit' column for both 'name' and 'number' is highlighted with a red box.

Name	Type	Methods	Vis	Opt	Edit
name	String, 64				<input checked="" type="checkbox"/>
number	Integer, 3		<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>
textField	String, 64		<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>

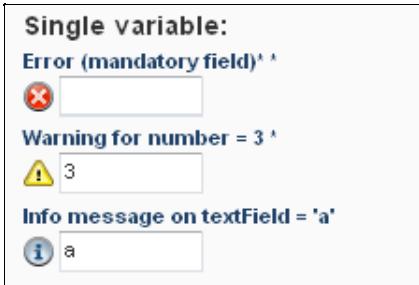
**General** **Methods**

**Element properties:**

Name	Value
Display Format	
Optional	FALSE
Editable	
Visible	
Label	String64

## Field Errors

The onValidate method provides a validateStatus that displays a validation error, warning, or information image next to the field that has been validated. If a field is in not valid and is in error, then an error image appears next to the field at runtime. Similarly, if a field validated with a warning or information status, then the field displays with the standard warning and information images:



## Tab Errors

When validating a TabFrame, it is possible to display an icon on the Tab that represents the Validation state of that tab. The icons that can be displayed can change dynamically based on the TabFrame's Validation Icon property that can be set to either Static, State Icon, or Overlay. The icons that are displayed represent the validation state of the tab as either Unknown (validation has not yet been performed), Error (validation errors exist), Warning (errors that constitute a warning only), and Info (information errors of low severity).

A **Static** Validation state indicates that there is a static icon (icon defined in the Icon property) and to the right of the icon is a validation warning that dynamically changes based on the validation.

A **State** icon are user-defined icons that change dynamically based on the state of the validation. The Icon property is set to the user-defined icon name (that is, IconName) and the associated validation icons are defined as IconName\_error, IconName\_warn, and IconName\_info. When there is a validation error (for example, mandatory field not filled), then the system displays the icon named IconName\_error. Similarly, for a warning validation, the icon that is displayed is the IconName\_warn and then for the information validation status, the IconName\_info is displayed. The Unknown validation state is the icon assigned to the Icon property. The user-defined icons should be put in the metadata Resources folder.

An **Overlay** icon overlaps the icon defined in the Icon property of the TabFrame element based on the validation status of the tab. The following table shows examples of the TabFrame with the various validation icons.

Validation Icon	State	Icon
Overlay	Unknown	Overlay
Overlay	Error	Overlay
Overlay	Warning	Overlay
Overlay	Information	Overlay
Static	Unknown	Static Icons
Static	Error	Static Icons
Static	Warning	Static Icons
Static	Infomation	Static Icons
State	Unknown	State Icon
State	Error	State Icon
State	Warning	State Icon
State	Infomation	State Icon

## Enable Back and Next Buttons in a Web Browser

You can enable your Web browser's standard **Back** and **Next** buttons for navigation between pages.

Web browsers consider the history of pages visited as a set of page URLs with page titles and data submitted on a form, provided that the previous request was completed by submitting a form. When you click the Web browser's **Back** button, the browser gets to the most current history record (a previous page URL) and resubmits the data, if applicable. When you click the **Next** button, the browser returns to the page URL that the **Back** button had left. Some Web browsers also allow you to click a down arrow located next to either the **Back** or **Next** button, which shows a short history of recent pages you have visited, and to navigate to any of them listed.



A new history record is automatically created when you switch between application URLs. For example, if you have arrived at the `http://localhost:8080/cwf/admin` application from the `http://localhost:8080/cwf/selectApp` page, clicking the **Back** button shows the Select Application page.

**Note:** If the application does not have its own URL map and the current application URL remains as `http://localhost:8080/cwf/selectApp`, clicking the **Back** button shows the login page instead (that is, `http://localhost:8080/cwf/`).

### Add a History Record

To create a new history record in browser, call the `setCurrentHistoryID` method of the global History object. The following are examples:

```
History.setCurrentHistoryID("startPage");

// with custom title:
History.setCurrentHistoryID("startPage", "Start Application Page");

// with custom title and user data
History.setCurrentHistoryID("startPage", "Start Application Page", userData);
```

where the first parameter is a unique identifier for the current page of the application.

After adding a history record in the browser, the application URL in your Web browser's address bar should change to the following:

```
.../cwf/app#startPage
```

**Note:** It is recommended that you assign a URL for the Application UI to ensure that the **Back** and **Next** buttons work properly.

To get the current history record, call this method:

```
String currentID = History.getCurrentHistoryID();
```

Each time the current history ID gets changed, the History object backs up the current ID, which can be retrieved by calling the following method:

```
String previousID = History.getPreviousHistoryID();
```

## Restore Page on a History Event

When you click either the **Back** or **Next** button in your browser to go to the previously visited or next page, respectively, the History object fires the callback notification, and then passes the history ID associated with that page and the user data object to the listener.

The callback method is fired in the following cases:

1. When you click either the **Back** or **Next** button in the Web browser. The method navigates to the previous or next history token that was previously added by the History.setCurrentHistoryID() method.
2. When you navigate back from one Application UI (URL) to the last history token of the previous Application UI.
3. When you navigate to the page by using a stored bookmark or a link with a history token (that is, .../cwf/app#historyID).

In cases 2 and 3, the history token may not have been added to the current application session. You can verify that the token has been added by calling the History.hasHistoryRecord(historyID) method.

The Application UI is automatically added as a listener for history notifications when it becomes current, and is removed on switching to another application or when logging out. To restore the page by its history ID, override the onHistoryCallback method:

```
function onHistoryCallback(historyID, userData) {  
    switch (historyID) {  
        case "startPage":  
            // restore start page content...  
            break;  
        case ...  
    }  
}
```

### Notes:

- If the callback function returns either true or nothing, the historyID parameter becomes current. If the callback function returns false, it stays on the current History record and History.getCurrentHistoryID() still returns the old one.
- When navigating away from the current application, all history data is cleared on the server, although created records remain in the browser history.

## Add a Custom Listener

You can also register your own history listeners. For example, the [navigation bar object](#) can be added as a listener. Sometimes, it may be more convenient to handle the **Back** and **Next** button functionality at the navigation bar level:

```
History.registerCallback(navBar, "onHistoryCallback");
```

where the first parameter can be any UI object (for example, top-level or Document UI, Navigation Bar, Navigation Tree, Order UI, and so on) and the second parameter is the name of the callback method.

### Notes:

- All custom listeners are being removed automatically on leaving the application, but it is recommended that you unregister the history listener manually when the object is not in use anymore by using this method:  

```
History.unregisterCallback(navBar);
```
- The order of callback method invocations is not guaranteed. For example, navigation bar's method can be called first and only then can the onHistoryCallback method of the application be called, or vice versa. It is recommended that you do not split the logic to handle the same history ID in different objects. Every listener should handle its own unique history IDs.

## History Class API

The following methods are available from the History API:

Method	Description
<code>setCurrentHistoryID(historyID, title, data)</code>	<ul style="list-style-type: none"><li>• If the <i>historyID</i> parameter is specified, it is different from the current one. Instead, a new</li></ul>

	<p>record is created in the browser history.</p> <ul style="list-style-type: none"> <li>The <i>title</i> parameter is the custom title for the history record. If the parameter is either null or unspecified, it takes the current window title.</li> <li>The <i>data</i> parameter is an optional user data object, which is stored for each current application session. When the callback function is invoked, it is passed as a second parameter.</li> </ul>
<code>getCurrentHistoryID()</code>	Returns the current history ID, which is either set by the <code>setCurrentHistoryMethod</code> or after the history callback.
<code>getPreviousHistoryID()</code>	Returns the previous history ID.
<code>getHistoryData(historyID)</code>	Returns the user data object stored by the history ID. The object is stored only while working with the current application.
<code>hasHistoryRecord(historyID)</code>	Returns true if the specified history ID has been added within the current application session.
<code>registerCallback(listener, method)</code>	Registers the custom listener for the history callback notification.
<code> unregisterCallback(listener)</code>	Removes the custom listener for the history callback notification.
<code> unregisterCallbackByMetadataName(metadataName)</code>	Removes all listener instances, except for the Application UI, of (or extends) the specified metadata.

# Scripting

As per MVC architecture pattern, the **User Interface** metadata object type is the *Controller* of the pattern that decouples and defines the interaction among data access, business logic, data presentation and user presentation. As such, scripting resides mostly within [User Interface](#) metadata objects.

This article explains various important ideas and tips when scripting, in addition to regular [JavaScripting guidelines](#) in Velocity Studio.

## User Interface Object Instantiation

As controller, User Interface metadata objects must be instantiated to serve requests from a browser. At runtime, it is automatically instantiated by AVM when that User Interface is requested by browser. Specifically, it is the browser's requested URL that identifies the User Interface as per [URL mapping](#) configuration.

However, when displaying a Form using Form Frame from a different User Interface, the User Interface of the Form Frame must include and explicitly instantiate, by script, the Form's User Interface. See section [cascading User Interface objects](#) below for details. With JavaScript, the syntax of User Interface instantiation is similar to other object instantiation. For example, if you are to instantiate a top-level User Interface metadata object *topUI* in namespace *nsTop*, you may write:

```
var myTopUI = new nsTop.topUI();
```

or, alternatively:

```
var myTopUI = new com.conceptwave.system.UserInterface("nsTop.topUI");
```

## Scripting with Included User Interface

*Included User Interface* -- User Interface metadata objects that are included by model metadata objects such as Document, Order, Finder and etc -- have a fixed pathname of:

```
<metadata object pathname>.UserInterface
```

For example, the included User Interface of Document *doc1* in Namespace *ns* has the pathname of *ns.doc1.UserInterface*.

Even though, in this pathname arrangement and in **Navigation Pane** of Velocity Studio, that the included User Interface appears "underneath" its model metadata object, the instantiation of model metadata object does not imply instantiation of the object's included User Interface.

In our example above, if Document *doc1* is instantiated via

```
var doc = new ns.doc1();
```

runtime only gives you a Document with no User Interface counterpart. Specifically, *ns.doc1.UserInterface* is null at this point.

To instantiate an included User Interface with its model counterpart created as well, you may write the following script:

```
//create a document instance
var doc = new ns.doc1();

//set some values on the doc here...
...

//create the user interface
var docUI = new ns.doc1.UserInterface(doc);
```

Or alternatively, simply:

```
var docUI = new ns.doc1.UserInterface(null);
```

In the second script, the User Interface instantiation automatically creates a new instance of `doc1` as the model. Whereas in the first script, the existing Document object `doc` is the model. In both scripts, the Document instance can be accessed via the User Interface Variable `ns.doc1.UserInterface.model`.

## Accessing User Interface Variables and Methods

Similar to JavaScripting with other metadata objects in Velocity Studio, Variables of User Interface can be accessed (get/set) by `UserInterface.VariableName` syntax. For example, Variable `confirmed` in User Interface object `docUI` that is created above may be set to true by:

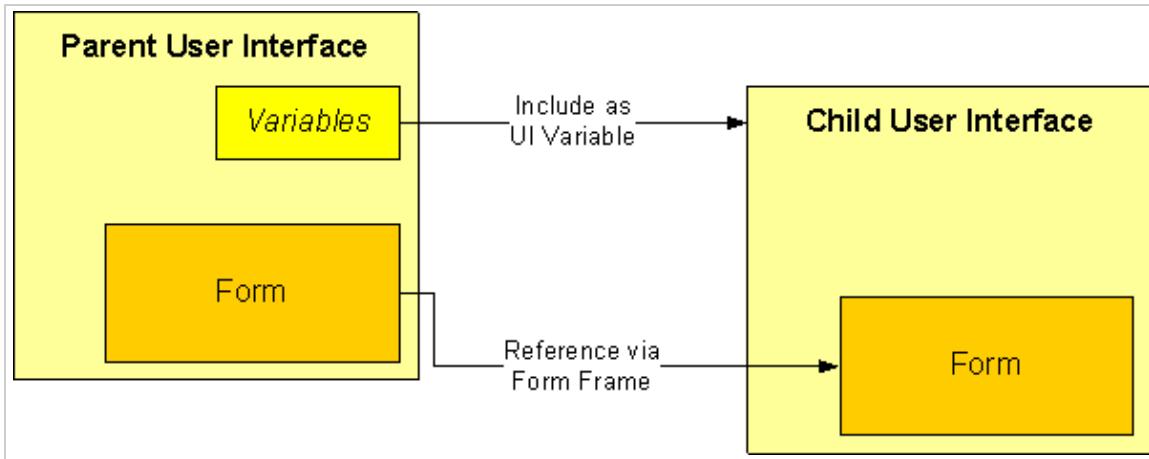
```
docUI.confirmed = true;
```

Similarly, Methods of User Interface can be invoked by `UserInterface.MethodName(param1, param2...)` syntax. For example, Method `submitOrder` without parameters may be invoked by:

```
docUI.submitOrder();
```

## Cascading User Interface objects

Often, a Form or a Page is required to display another Form from a different User Interface, via [Form Frames](#). For the sake of this discussion, let's denote the User Interface of the Form or Page that displays a foreign Form as the *parent*, and the User Interface of the foreign Form as the *child*. As explained in the [notes in Form Frame](#), the parent User Interface must include the child User Interface as one of its Variables.



At runtime, the parent User Interface delegates to the child User Interface to render its Form. Thus, whenever the foreign Form needs to be rendered, the child User Interface must already be instantiated. (Otherwise, the Form will not be rendered.) This is required to be done in script. For example, the child User Interface object instantiation can be done at parent User Interface's initialization script,

```
this.childUI = new ns.doc1.UserInterface(null, this);
```

where `childUI` is the UI Variable declared in parent User Interface metadata object, and the child User Interface to instantiate is an included User Interface in Document `ns.doc1`.

The second parameter of the constructor is responsible for specifying which User Interface, if any, that the instantiated User Interface object shall be parented. Notice that this parameter is necessary even though the parent User Interface has the child User Interface as one of its User Interface Variable. Indeed, there is no automatic parenting when a User Interface Variable is declared and set within another User Interface metadata object.

From the child User Interface object, the parent User Interface can be get/set using `.parent` variable. In our example, the parent User Interface can be retrieved by:

```
childUI.parent
```

### Design Note

You are not necessarily required to set the parental relationship. That is, you may instantiate a User Interface with no parent, even in the scenario of cascading User Interfaces. However, popups will not work, because the dialog variable,

which is set when an object is displayed in a popup, is found by going up the parenting chain.

More importantly, setting parental relationship enables sibling User Interface objects to share the parent's Variables, if there are multiple User Interfaces belong under the same parent.

User Interface objects may be cascaded at runtime by means of [User Action method](#) instead. For example, an application-level Page (for example, Page of a User Interface built from `com.conceptwave.system.Application`) may have a Menu Item, when clicked, invokes a User Action method that instantiates a model (a Document, Order, or Finder) and displays its Form in the Page. The model and its User Interface are not declared in application User Interface as variables. Yet, it is not necessary to instantiate these objects by script, because the model (that is, the **Object** in User Action) and its User Interface are automatically instantiated when the User Action method is invoked. In particular, the User Interface object is parented by the application User Interface at instantiation.

## Parameters in User Interface Object Instantiation

Please see [User Interface contructor syntax](#) in JavaScript reference for more details. The parameters of User Interface constructor are as follows:

- Parameter 1 is the model (for example, Document, Finder, Order, etc.). If it is a top-level User Interface, set to `null`.
- Parameter 2 is the parent User Interface. Set to `null` if there is no User Interface to parent from.

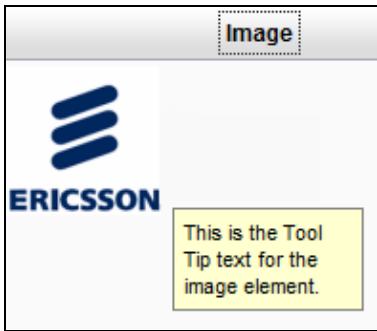
## Custom Help

---

The Velocity Studio provides two levels of custom help text creation that are available at the field element level and the user interface level through tooltips and popup windows. At the field element level a Tooltip property is available that appears at runtime when the user's mouse hovers over the field. The pop up help option is available at the User Interface level of Presentation, Documents, Finders and Orders and enables you to enter more extensive help information. The popup help is displayed at runtime when the user clicks on any input field in the form and the help image on the top right hand corner of the browser.

### Tooltip Help

Tooltip help is available at the form element level as a property of certain elements. You can enter help text or information about the element at the form element level via the tooltip property. At runtime, a small hover box is displayed when the user hovers a cursor over the field element (without clicking it).



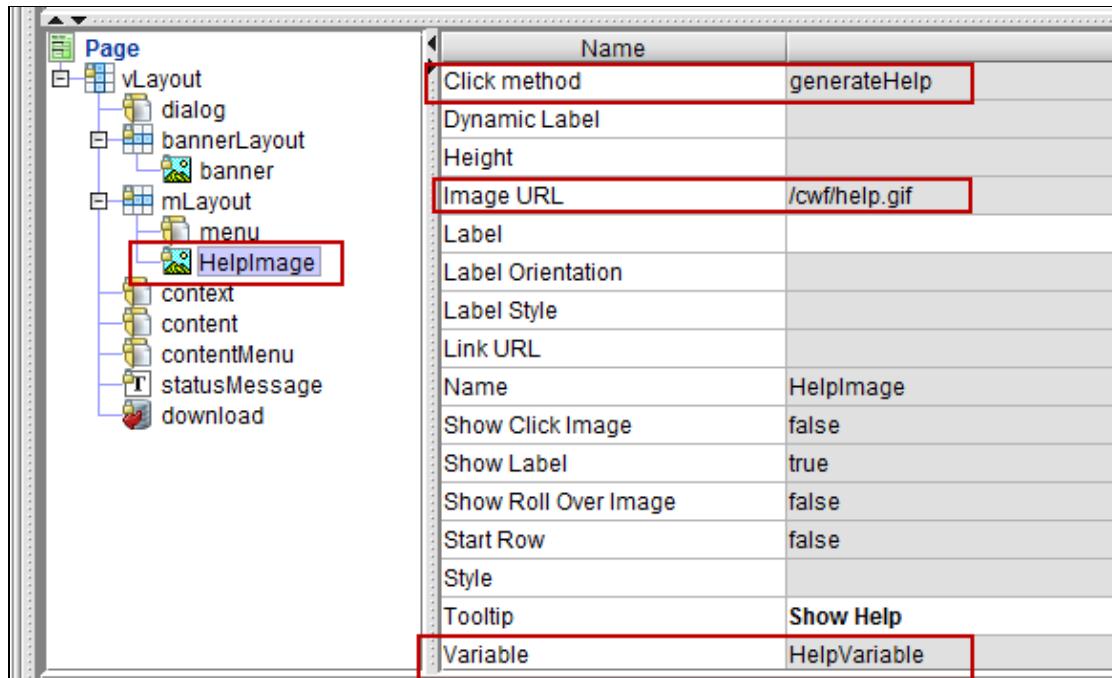
The following table shows which field elements supports the tooltip functionality.

Element Type	Tooltip Available
<a href="#">Button</a>	Yes
<a href="#">Cell Spacer</a>	Yes
<a href="#">Chart</a>	Yes
<a href="#">Checkbox</a>	Yes
<a href="#">Date Field</a>	Yes
<a href="#">Date Time</a>	Yes
<a href="#">Dialog</a>	Yes
<a href="#">Dynamic Document</a>	Yes
<a href="#">Dynamic Table</a>	Yes
<a href="#">Form Frame</a>	Yes
<a href="#">Grid Layout</a>	Yes
<a href="#">Header</a>	Yes
<a href="#">Horizontal Layout</a>	Yes
<a href="#">HTML Content</a>	Yes
<a href="#">Hyperlink</a>	Yes
<a href="#">Image</a>	Yes
<a href="#">Iterator</a>	No
<a href="#">Label</a>	Yes
<a href="#">Large Text</a>	Yes
<a href="#">Large Text Area</a>	Yes
<a href="#">Layout Spacer</a>	No

<a href="#">Menu</a>	Yes
<a href="#">Password Field</a>	Yes
<a href="#">Radio Button</a>	Yes
<a href="#">Reference Field</a>	Yes
<a href="#">Row Spacer</a>	Yes
<a href="#">Section</a>	Yes
<a href="#">Section Header</a>	No
<a href="#">Section Stack</a>	No
<a href="#">Separator</a>	No
<a href="#">Select Field</a>	Yes
<a href="#">Tab Frame</a>	Yes
<a href="#">Table</a>	Yes
<a href="#">Tabset</a>	No
<a href="#">Text Area</a>	Yes
<a href="#">Text Field</a>	Yes
<a href="#">Translation Field</a>	Yes
<a href="#">Tree</a>	Yes
<a href="#">Upload File</a>	Yes
<a href="#">Variable</a>	N/A
<a href="#">Vertical Layout</a>	Yes

## Popup Help

At the User Interface level, popup help is available for Presentation, Documents, Orders, and Finders. Velocity Studio contains the popup help functionality within the Page form of the com.conceptwave.system.Application. The scripts, variables and graphic images related to the popup help functionality defaults within the properties of the HelpImage of the Page form. The image graphic points to the help.gif file and this image appears on the top right hand corner of the application. The **Click method** property is pointing to the generateHelp script. The script uses the *string* variable HelpVariable to pass the correct help text at runtime when the user clicks on this image.



The generateHelp method calls the [Global.showHelp\(this.HelpVariable, width, height, this\)](#) object. The width and height properties control the size of the popup help.

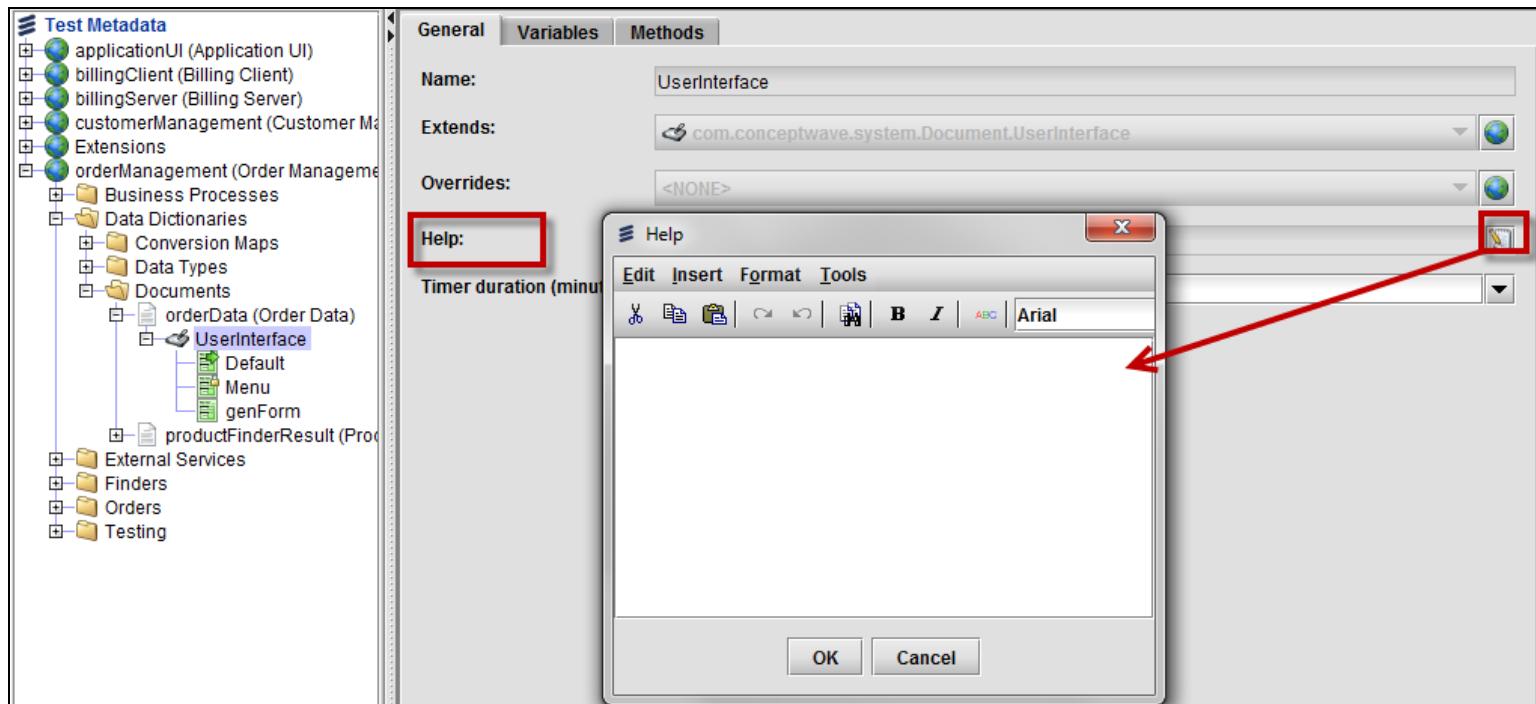
For elements like Finders, which display a dialog box, the help image will appear in the bottom left hand corner of the dialog box. The user would click the help image within the dialog box - such that the focus remains on this element. If help is not found within the dialog box then help for the parented form displays.



In the Global.showHelp object, "this" is the dialog controller and should point to the current user interface.

## Design Mode

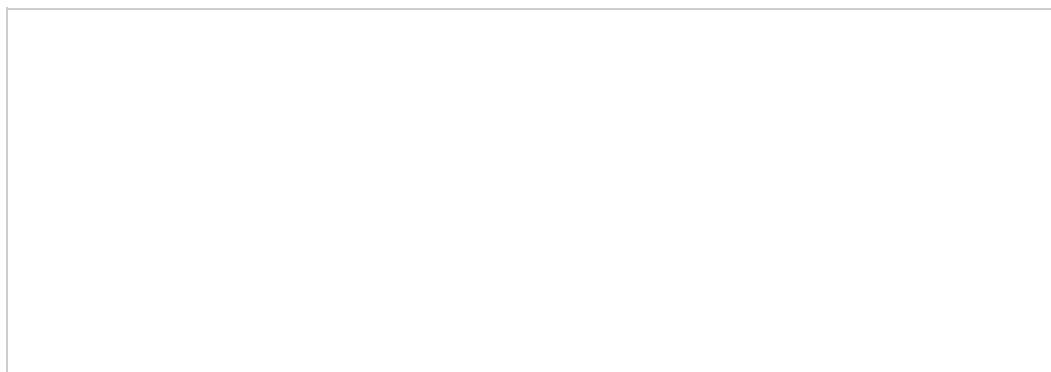
In design mode, the developer is able to create multiple lines of help text within the User Interface's General tab's **Help** field through a textbox editor.

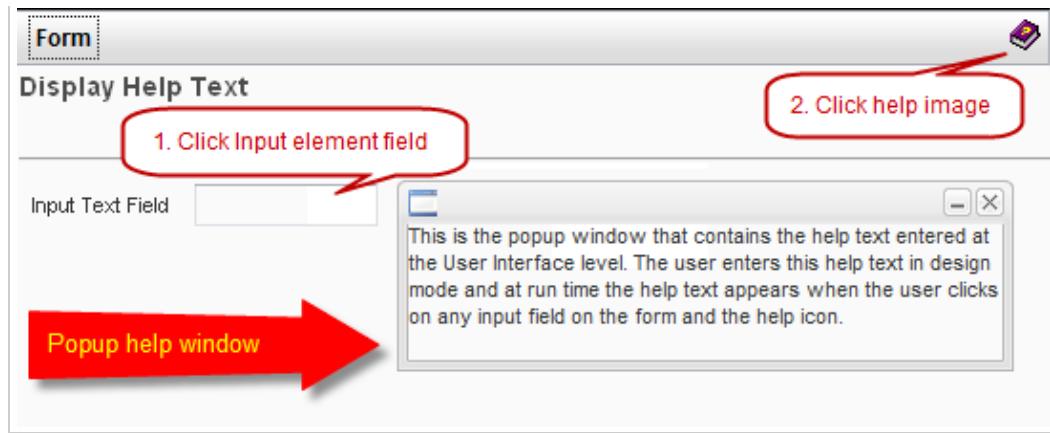


## Runtime

At runtime, the custom help will be displayed in a popup window by clicking any input field element within the form then clicking the help image (  ) on the top right hand corner of the page. The help image has a Visible permission. If help is not available for the current form, then the generateHelp method will execute a recursive search in a hierarchical process to find the next available help text at the parent level. If help text is not found within the hierarchical parent levels, then help is not displayed.

The following is an example of the runtime display of the popup help window.





## Dynamic Timer Duration

Any User Interface can have a timer bound to it. If a timer interval has been specified, the timer calls the `onTimer()` method each time an interval elapses.

The screenshot shows the SDE200 Training Metadata interface. On the left is a tree view of metadata objects, and on the right is a details panel with tabs for General, Variables, and Methods. The Variables tab is selected. In the tree view, under the applicationUI node, there is a UserInterface object named "UserInterface". This object has three sub-items: Default, Menu, and criteriaForm. The "UserInterface" item is currently selected. In the Variables tab, the "Name:" field contains "UserInterface". The "Extends:" field shows a relationship to "com.conceptwave.system.UserInterface". The "Overrides:" field shows "<NONE>". The "Help:" field is empty. The "Timer duration (minutes):" field contains "0 (<Inherited>)" and is highlighted with a red border.

The dynamic script duration is configurable. The `dynamicTimerDuration` script in `com.conceptwave.system.UserInterface` allows you to override this script in your metadata, and return either an integer or a script that returns an integer to configure the time duration.

To use the script, create a new `dynamicTimerDuration` script with a return type of integer in `com.conceptwave.system.UserInterface`.

General Variables Methods

UserInterface

- cwChildDialogClose
- dynamicTimerDuration
- editablePerm
- onAutoSave
- onClick
- onFullTextEdit
- OnInit
- onModelAttached
- onModelEvent
- onRefClick
- onTimer
- onValidate
- runTrigger
- save

By default, the implementation is empty. If a value is returned, it is the timerDuration. Otherwise, use the static value in your metadata.

## Form Overview

Forms define the presentation fragments of a Web page. Forms are always contained within a [User Interface](#) and typically define the presentation for metadata object included in the User Interface. For example, a Document's Form defines the fields of a Document and how these fields are shown and edited. In contrast, Forms of top-level User Interface, typically construct a Web page component or section to be included in a Page. Forms cannot be requested by a Web browser directly.

Forms can reference other Forms using a Form Frame (or similar Elements). For example, when configuring a search Form such as a Finder, this form can reference a Document Form to show search results.

## Form Editor

In Velocity Studio, the Form editor has three main components:

Component	Description
Element Tree Pane	Elements assembled in tree structure that determines how the Form is rendered. The root of the tree is the Form name.
Element Properties Pane	Properties of the element selected in the Element Tree can be edited in this Pane.
Preview Pane	Previews the Form as assembled in the Element Tree. Interaction (clicking and inputs) with the Form in the preview pane is available with the exception of variables and methods that are referenced by Form Elements.

The screenshot shows the Velocity Studio interface for editing a form. The left side features the 'Element Tree' pane, which displays a hierarchical tree structure of AVM Metadata nodes. The 'orderServicesPage' node is currently selected, showing its child nodes: 'centeringFrame', 'orderFrame', and 'footerFrame'. The right side is divided into three main sections: the 'Properties' pane, the 'Preview' pane, and a bottom 'Object' pane.

**Properties Pane:** This pane contains a table of properties for the selected 'orderServicesPage' node. The properties listed are:

Name	Value
Cell Alignment	center,top
Height (px, %)	100%
Name	orderServicesPage
Show Resize bar	

**Preview Pane:** This pane displays a promotional offer from 'all the best' for a bundle deal. The offer includes 'Cable, Road Runner and Digital Phone' for \$29.95 per month for 12 months. It also allows building a custom bundle by selecting packages. The selected packages are:

- CABLE:** Selected
- HIGH SPEED ONLINE:** Selected
- DIGITAL PHONE:** Selected

For each package, there are dropdown menus to select specific options. For example, under 'CABLE', options include 'Digital Cable', 'Standard Cable', and 'Basic Cable'. Under 'DIGITAL PHONE', options include 'Digital Phone Unlimited', 'Digital Phone Carolinas', and 'Digital Phone Local'.

**Bottom Object Pane:** This pane is currently empty, showing a table with columns for 'Description' and 'Object'.

## Form Icons

Form icons provide information about the state and functionality of a Form.

Form Icon	Description
	New Form
	Form is locked. The Form is inherited from a base User Interface and is read only.
	Form is extended from another Form.
	Form is overridden from a locked Form.

## Previewing with Style Sheets and Skins

In addition to using the toolbar, you can refresh the Preview Pane with the refresh icon , and also choose a **Style sheet** and a **Skin**. You can set the style sheet, by selecting the default system style sheet (cwf.css), or you can customize your own CSS style sheet as per your desired page design, and place the .css file under the **Resources** folder in your project (reloading project is required). The Skin option governs the overall appearance of the Page. Choose from a selection in the Skin list box; they are provided by our AJAX UI technology provider, *SmartClient*. The selected options in the Style sheet and Skin list box are not assigned to the Form itself. At runtime, style sheet and skin follows the Page that references the Form.

**Note:** You cannot create your own Skin.

## Changing Language Preferences

You can specify the language of the Label property for each Form element by clicking the ellipses. The Translation Editor opens where you can select the language preference for the label. If the Label property is a child to a parent Form element, the Inherit checkbox option is checked. The Inherit checkbox is checked by default, which means the element inherits the label of the Variable. If unchecked, then you can specify a local label for the element. The language preference is not available when the Inherit option is active.

To ensure that you have access to the language of your choice:

- You need to add the list of supported languages under the Metadata Header object in the *Languages* tab.
- You must have a label for the default language (en-xx) for translations of other languages to work. If you do not specify a label in English, then other languages will not appear in the list.
- At runtime, there is a default language for translations. The language is set based on the following section of your metadata:

```
<locale type="loc">
  <locale>en_CA</locale>
  <rightToLeft>false</rightToLeft>
</locale>
```

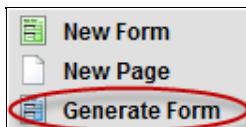
## Building New Forms

1. Right-click the top-level User Interface node and click **New Form**. This creates a blank new Form.
2. Continue by adding Form Elements to the Form and configuring the properties. For more information about Form Elements, see [Form Element Overview](#).
3. To expedite Form creation, use the [Variable](#) Form Element shortcut to create appropriate Form Elements for the User Interface variables.

**Note:** Upon creation of the metadata object, many metadata types (such as Finders, Orders, Documents), derived metadata objects and default Forms may already exist under the User Interface.

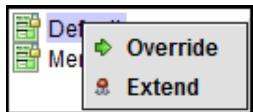
## Generated Forms

If you are working with a particular metadata object such as Documents or Orders, Forms can be generated under the Included User Interface using the *Generate Form* command. By right-clicking the User Interface node, a menu appears where you can select the **Generate Form** action. This action generates a User Interface Form for the Model with some default settings. Use the **Generate Form** command as the starting point for the Form and proceed by customizing it according to your needs.



## Extend and Override Forms

The default Form provides a default User Interface presentation that you can customize.



You can customize the default User Interface presentation by selecting one of the following right-click menu options:

- Use the [Extend](#) function on a Form to retain the Form's current features. You can add, replace and delete Form Elements from the original Form to implement the desired modifications.
- Use the [Override](#) function to completely override the Form's current features. You start building the overridden Form as a blank Form.

## Display Forms at Runtime

Forms represent fragments of a Web page. These fragments can represent items such as, menu items, layouts, and fields to name a few. There are a number of Form Elements that you can use to configure a Form (see [Form Elements Overview](#)). Each elements contains properties that control the look and feel, as well as the functionality, of the Form element. When you create a top-level or included User Interface object, by default, **Default** and **Menu** Forms are instantiated under that metadata object. See [Top-level User Interface Objects](#) for more information about configuring User Interface objects. In addition, you can view Forms at runtime in two different ways:

- By configuring a Form as part of an application
- By mapping the Form to a URL

### View Forms that are part of an Application

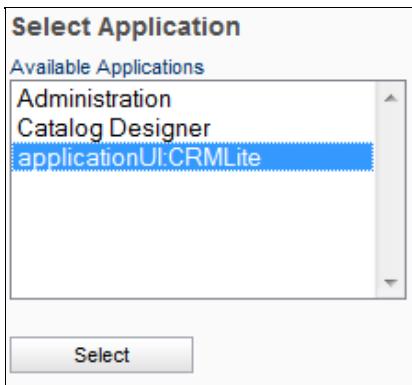
To view a Form that is part of an application, complete these steps:

1. In the Navigation Pane, right-click the namespace and add **New**.
2. In the New Metadata Object menu, select **Presentations > User Interface**.
3. In the New User Interface dialog, type the information for the User Interface.
4. In the Extends list menu, select *com.conceptwave.system.Application*.
5. Click the **Finish** button.
6. From the Navigation pane, select the User Interface.
7. Select the Menu Form and then right-click it and click **Extend**.
8. The Menu Form appears in the Element Tree pane.
9. Right-click the Menu Form and select **Add**.
10. The Add Element dialog box appears.
11. Select the elements and then click the **Finish** button.
12. The Form element appears in the Element Tree pane beneath the Form object.

### Test the Form

To test the Form, complete the following steps:

1. From the main menu, click **Runtime > Run**. The Framework starts.
2. Open a Web browser and in the address bar, type the URL of the application. For example, `http://<host>:<port>/<application name>`
3. The Select Application page appears.
4. Select the application that you want to view and click **Select**.



**Note:** By default, the Administration application appears in the Select Application dialog box.

### View a Form that is mapped to a URL

Before you begin, create a Page in the same User Interface of the Form. Add a Form Frame to the Page. In the Form Frame properties, add a reference to the Form.

1. Right-click the metadata root node.
2. Click the **URL Mapping** tab.
3. From the toolbar, click the Add button.
4. In the Add Url Map Item, type the name of the Form that you create preceded by a slash "/".

5. From the **Controller** field, select the User Interface of the Page that you want mapped.
6. Click **Save**.

## Test the URL Mapping

To test the URL mapping, complete the following steps:

1. From the main menu, click **Runtime > Run**. The Framework starts.
2. Open a Web browser and in the address bar, type the URL of the application. In the application name, type the Form URL. For example, `http://<host>:<port>/cwf/<Form_Name>`.
3. The login page appears.
4. Login to the application and click **Login**.
5. The content of the Form appears in the browser.

**New Order Application Tracking Application**

Last name	<input type="text"/>
First Name	<input type="text"/>
Current Subscriber	<input type="checkbox"/>

 **Save**

For more information, see [Metadata Root URL Mapping](#).

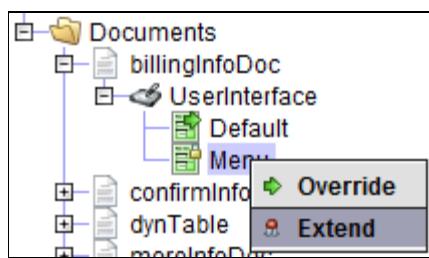
## Extend a Form

You can extend any existing Form, referred to as the *base Form*, to create a new Form under the same User Interface. The new Form contains all the Form Elements of the base Form; you can then add, replace or delete Form Elements to further customize from the base Form.

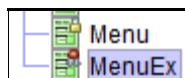
One of the common uses of the Extend function is to extend from a base Form that is inherited from a base metadata object (that is, a Document is extended from a base Document). This way, you only need to add on Form Elements of additional variables of the extended metadata object, rather than building the Form from scratch.

To extend a Form, do the following:

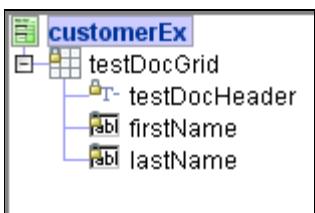
- In the **Metadata tab of Navigation** pane, right-click the base Form you wish to extend. The User Interface of the base Form can be a top-level User Interface or an included User Interface. Select **Extend**.



- The new extended Form has the Name of the base Form with appended "Ex"; the icon of the extended Form has a red flower at the top-right corner. The Form is added under the same User Interface of the base Form.



The extended Form has all the Form Elements of base Form, with "lock" at the top-left corner of their icons to show that they are from the base Form.

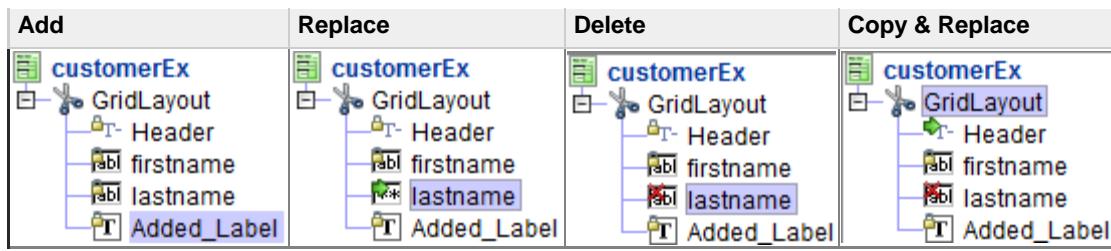


At the Element Tree Pane, you can:

- **Add** a Form Element to the Form, in the same way as in a regular Form. The added Form Element does not have the lock shown in its icon.
- **Delete** a base Form Element. The deleted Form element still appears in the Element Tree Pane, with a red cross at the top-left of the icon. However, it is not rendered at Preview pane or when at runtime. You can *undelete* it by right-click it and select **Delete**.
- **Replace** a base Form Element with another Form Element. The new Form Element appears in the Element Tree Pane with a green arrow at the top-left of the icon. You can revert the replacement by right-click it and select **Delete**.
- **Copy and Replace** a Form (which is the equivalent to editing a base Form element).
- **Copy a Form** allows you to copy and then edit an exiting Form.
- **Paste a Form** allows you to move an exiting Form to a new location.

In addition, there is drag and drop functionality which allows you to reorder Forms that appear in the Form Element Tree provided that they are not locked.

The following are Element Tree pane samples of each action performed to the base Form.



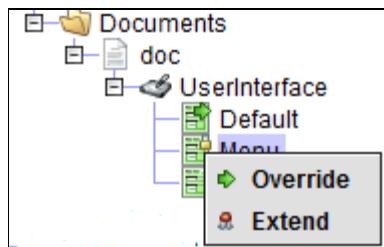
## Override a Form

You can override a base Form that is inherited in the User Interface when extending from a base metadata object. These Forms are shown with a lock mini-icon at the top-left of the Form icon . The base Form is removed from the User Interface, and is replaced by a blank Form in the same name.

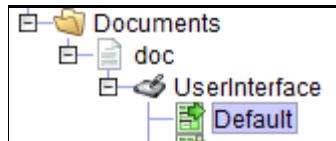
One of the common uses of the Override function is to implement **Default** Forms (that is, Forms named "Default") that are inherited from system-base metadata objects, some of them exist as a placeholder and intended to be overridden.

To override a Form, do the following:

- In the **Metadata tab** of **Navigation** pane, right-click the base Form that you want to extend. The User Interface of the base Form can be a top-level User Interface or an included User Interface. Select **Override**.



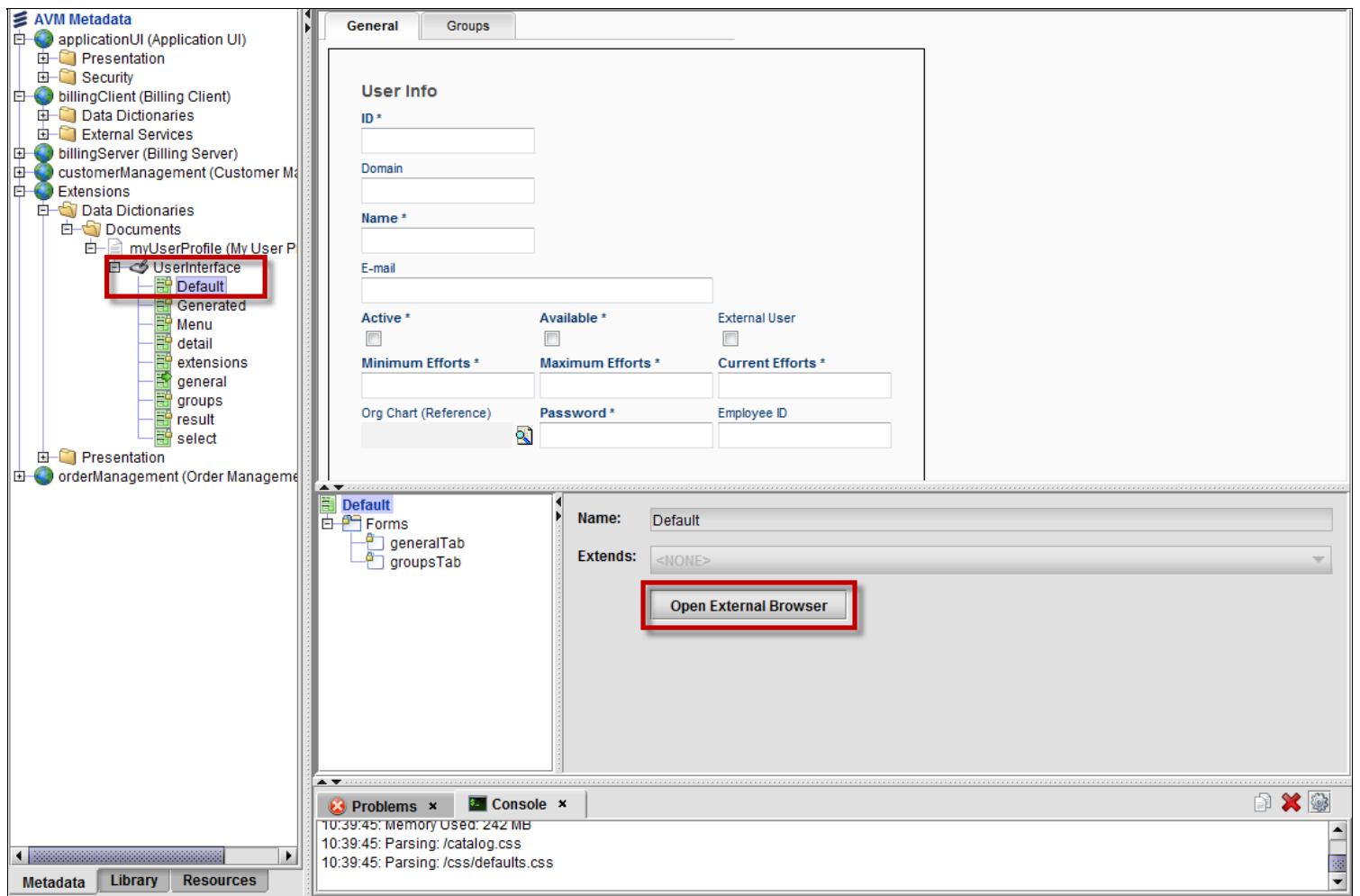
- The overriding Form has the same Name of the base Form; the icon of the overriding Form has a green arrow at the top-left corner. The Form is created under the same User Interface of the base Form.



The overriding Form is created as a blank Form with no Form Elements.

## Open a Form in a Web Browser

The **Open External Browser** button appears on a form, under the UserInterface, which allows you to open and view the form in a Web browser.



Clicking the **Open External Browser** button launches the form (in this example, it is the User Info form) in your default Web browser.

## Get Form Names from Metadata UI Objects

You can use the `getOverlays()` method, which is available in the `MetadataUserInterface` class, to get form names from metadata objects. This method returns an array containing all metadata overlays with unique names from both local and base controllers.

To get the name of each metadata overlay, the API's *name* can be used. The following sample code creates a document user interface and displays a message showing the names of all overlays in the interface:

```
var doc = new ns.MyDoc();
var docUI = new ns.MyDoc.UserInterface(doc);
var overlays = docUI.metadata.getOverlays();
var overlayNames = '';
for(var i=0; i<overlays.length; ++i) {
    overlayNames += overlays[i].name + ',';
}
Global.showUserMessage(overlayNames);
```

## Best Practices for Form Design

---

When modelling a form, follow these guidelines:

- It is not good practice to design a form that requires horizontal scrolling. Keep forms to a width that can be presented through a typical browser at a minimum resolution. This width varies by font size. This width is 60 characters for a 1024 x 768 resolution screen at the default font size.
- Vertical scrolling is more common, but can be minimized wherever possible. Vertical scrolling can be accomplished by using group forms with tabs to break up a long form into several smaller forms, each accessible through a tab. Note that this generates a more complicated user interface that may not be as straightforward as is required for self-care.
- Keep general form parameters consistent across the application (for example, number of columns, labels left or on top, column width, and so on).
- Keep radio button and checkbox use down to items where a small set of options exist (for example, less than five). These controls consume a lot of space on a form.
- Keep drop-down picklists to items with a medium result set (for example, less than 25). These controls, unless **editable key** is selected, do not permit filtering and may result in an inability to find or select the desired item, especially if the result set is larger than the permitted set. Popup finders are more appropriate for large results sets, especially where the size of the result set is not predictable.

## Design Dynamic Forms

A common scenario involves the presented form growing or shrinking, or changing upon entry of specific information by the end user. This scenario can be accomplished by designing different forms to represent each view, or by specifying access rules that enable or disable specific fields.

In general, the use of access rules is preferable over additional forms:

- Access rules define variable-level permissions that apply to wherever a variable is referenced, which precludes oversights where a form field may be presented to an unauthorized user.
- The use of access rules eliminates the need to define a form for every end user, and every state ( $N \times N$ ), in to control what is visible and read-only.
- Reduction in forms reduces translation efforts.

Additional forms, rather than access rules, are appropriate in the following scenarios:

- The form can also include different static text, computed fields, images, or controls, such as buttons. These items are not controlled through access rules and can be modelled using a different form.
- The use of access rules results in too much whitespace. Setting a field to non-visible through access rules does not remove the field from the form. As a result, a form with 80% of non-visible fields will be 80% blank. In this case, it is better to define a new, more streamlined form.
- The type of control or formatting for a field must change. For example, a read or write field may be displayed as radio buttons until selected, then change to read-only. Once the field is read-only, it may be preferable to display the selected option only. A new form is required to change the field from /rv to a standard format.

## Group Forms - Specify a Default Form

Group forms do not need to specify a default form (that is, a form with no permissions). However, it is good practice to define such a form in the unlikely event that the view conditions for any of the child forms in the group are not met. When the system is unable to identify a form to present, a message such as You are not authorized... displays. The specification of a default form eliminates the possibility of a user seeing such a form.

However, note that if the group form is defined with tabs, the default form is always displayed. In this case, it may be required to omit the default form and review the form conditions to ensure that at least one form is always displayed.

## Use of Form Labels

In general, the specification of form field labels can be minimized:

When form label is not specified, the visual label specified for the variable, or in its absence, the variable label is used in the user interface. Specification of labels at the form level, especially where multiple forms are defined that include the same variables, requires additional effort in the initial modelling.

- Changes to the labels assigned to the underlying variables are not automatically propagated to the forms.
- Internationalization is complicated as the translation effort must be performed for each form, rather than once at the variable level.

Nevertheless, the specification of form field labels is both valid, and required, in the following circumstance:

- The label displayed for the field varies by form. For example, the label presented on a self-care form is *customer number*, but it is presented internally as OCN, or the label may vary by context or state (for example, proposal number versus order number).

Even where form field labels are appropriate, consider using the variable's visual label for the most common label and specify the form field labels only for the exceptions.

## Use of Trigger Fields

Trigger fields are used to force a round trip to the server in response to a change in the value of a field. They can be used to:

- Perform an action on a change in a field (for example, clear the currently selected address if the customer selected changes, or retrieve the selected address and populate the form's address fields based on the selection).
- Force a re-evaluation of the contents of picklists (for example, present only those addresses applicable to the selected customer).
- Force a re-evaluation of the form to be presented.
- Force a re-evaluation of the access rules on each form field (that is, to enable or disable fields in response to the selection of an option).

Triggers can be used sparingly since:

- The round trip interrupts the order entry flow. If the form is scrollable, the form location is lost and the user must re-navigate to the appropriate position in the form.
- The round trip increases the load on the server. A solution with a significant number of triggers requires more hardware resources than one with no triggers.

## Use of Computed Fields

Computed fields provides a mechanism for presenting data on a form that needs a calculation or that is not located on the current document. When using computed fields, consider these guidelines:

- Computed fields are recomputed on each presentation. Keep these fields to a minimum, especially where the computation includes access to data outside the current document.
- When presenting a computed field based on data outside the current document, always check the context. For example, to present data from another part of the order, access the document from within the context of the order. Otherwise, the reference generates an exception.
- Consider using a calculated variable, rather than a computed field. These variables are computed on document load and save, rather than on each presentation of a form. Alternatively, consider using document load and store scripts to calculate the value and store it in a variable, which provides better performance when a series of variables are being calculated.
- Computed fields are not available for reporting as they are not database fields. Calculated variables may be persisted and be reported on.

## Use of Read-Only Attribute

Read-only formatting attributes exist at both the form and form field levels. The function performed by this attribute may be equally performed by the permission script associated with the document and variable access rule, respectively.

In general, avoid the read-only attributes at the form or form field level:

- The use of permissions at the document and variable levels ensures that the permissions are globally applied across all forms.
- Centralization of permissions in access rules provides a more transparent and easily maintained solution. The use of format codes /ro to drive read-only behaviour does not lend itself to a maintainable solution.

There are some scenarios where form-based rules are appropriate:

- Where the user is presented with two forms (for example, a summary form and a more detailed editable form) on request. In this case, the application of permissions results in the same treatment for each form and field, instead of the desired read-only treatment on the summary form, followed by read-write treatment on the detail form.

Some similar scenarios are not appropriate uses of form-based read-only attributes:

- Finder result set access: use of finder update permission to force result set to be read-only.
- Order retrieval outside of worklist: use of order or document permissions based upon the operation being performed (for example, if no operation is being performed, set to read only).

## **Use of Label Formatting**

Label formatting is specifying the font family, font size, and colour of a label. It is available for most labels and static text. In general, it is better to use the application stylesheet to control the attributes and to override only those items where special formatting is required (for example, static text).

## Page Overview

---

A Page is a top-level Form that can be requested and served to a Web browser. A Page is always contained within a User Interface object, and has a hard-coded name called **Page**. A User Interface can only have at most one Page.

A Page cannot be referenced by other Pages or Forms using a Form Frame or similar Form elements. Typically, a Page exists in a top-level User Interface that references Forms in the same User Interface and also outside of its User Interface.

**Note:** While Velocity Studio does not restrict you from creating a Page in the User Interface, this is not a common practice.

### URL Mapping to Pages

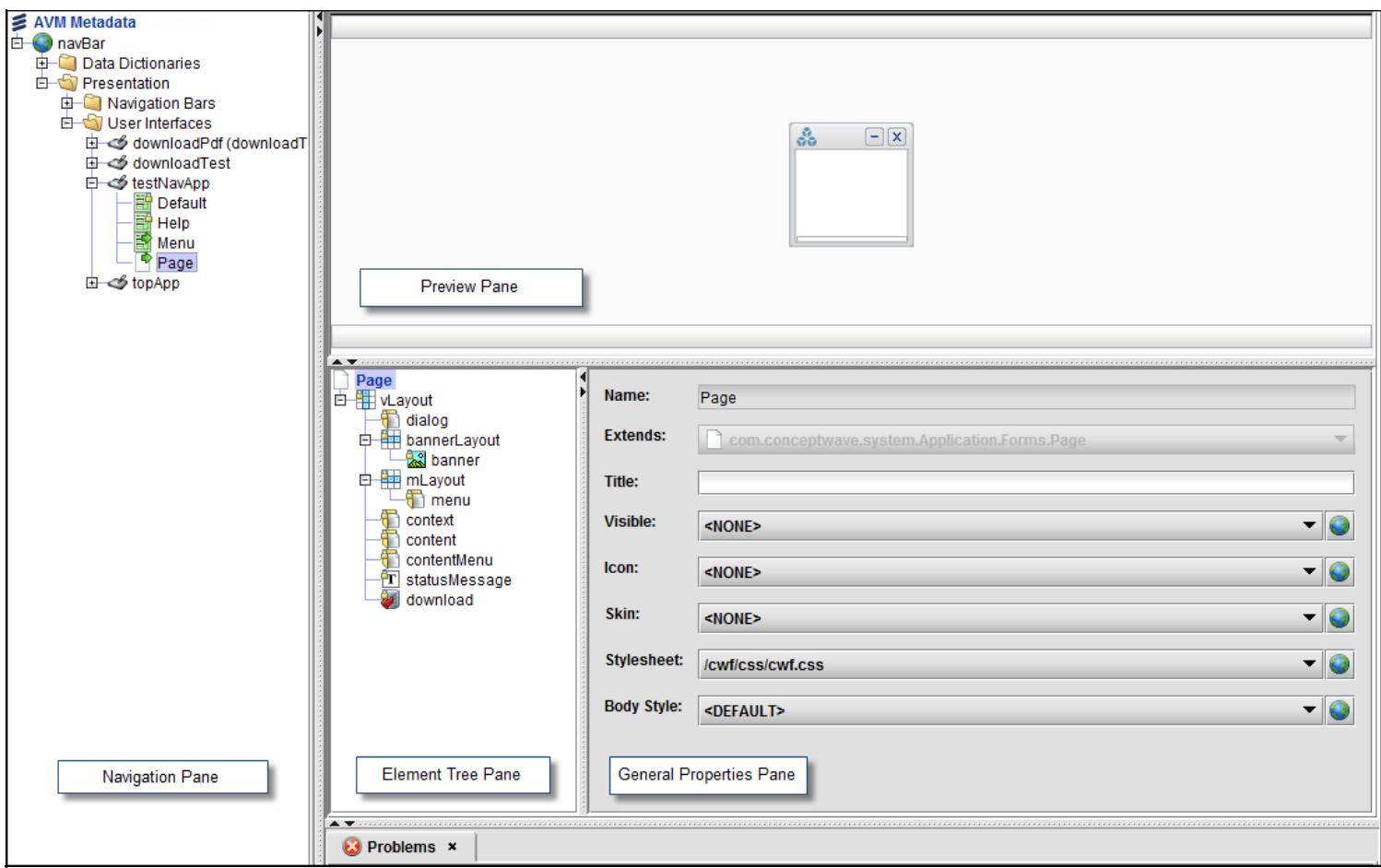
You can map a Page to a URL using the URL Mapping feature. The URL Mapping tab is found at the root node  **AVM Metadata** of your project. The URL Mapping enables you to expose (top-level) User Interface objects to Web clients, by assigning them to a relative URL path that is relative to `http://<hostname>:<port>/cwf`, where **cwf** refers to the name of the Page that you want displayed in the web client.

By configuring the URL Mapping of a Page, you allow for the configuration of the style sheet and Skin at the Page object level (providing the System.Application Object was selected when creating the User Interfaces Object). For more information

### Page Editor

In Velocity Studio, the Page editor has three main components:

Component	Description
Element Tree Pane	Elements assembled in tree structure that determines how the Page is rendered. The root of the tree is the User Interface object name.
General Properties Pane	Properties of the element selected in the Element Tree can be edited in this Pane.
Preview Pane	Previews the Page as assembled in the Element Tree.



The Page contains several Form elements. The Form Frame element named **dialog** displays a dialog box in the User Interface. This Form Frame enables objects to be created as dialog content and displayed it in the User Interface. Without the dialog Form Frame, objects continue to be created as dialog content, but they are not displayed in the User Interface.

**Note:** You can configure the Page's Form elements and properties by copying and replacing the Page.

- When the dialog element's *Unmanaged* property is set to *false*, content does not display in a dialog. If the Unmanaged property is set to *false*, and the Variable property is bound to a Form Frame, the dialog content displays in the main dialog window, top and left, positioned as 0.
- The **download** element is available under the Page form when the user interface parent is based on "com.conceptwave.system.Application". If an application default Page is overridden and the download element on the Page is deleted, the download element is automatically inserted as part of the Page.

## Change Skins

Skins controls the overall appearance of the Page. A Skin contains a fixed set of selections that cannot be changed or modified and is one of the elements contained in the style sheet. The style sheet is available from Velocity Studio's toolbar and from and the Page's general properties.

**Note:** At runtime, the system uses the style sheet and Skin configured on the Page (if any) to render the contents of any Form contained within the same User Interface.

## Assign Style Sheets and Skins

From a metadata perspective, Forms and a Page are equivalent, but a Page also has options for configuring CSS style sheet and Skin elements (see [Forms](#) for more information about their usage). The style sheet located on this property page is used to configure the Page at runtime. This is different from the style sheet located on the toolbar, which is only for rendering the preview pane and is not a choice that persists in metadata. Forms are not assigned CSS style sheet, but any Forms that are referenced in a Page use the style sheet of the Page. Therefore, it is important to ensure that you use the same style sheet when building your Forms and Pages.

Field	Description
<b>Name</b>	Name of the metadata object; hard coded to <i>Page</i> .
<b>Extends</b>	Page of the Base User Interface to derive from. Not changeable.

<b>Title</b>	Page title to be shown in browser at runtime.
<b>Visible</b>	Determines whether the Page is available at runtime or not.
<b>Icon</b>	Page icon to be shown in browser at runtime.
<b>Skin</b>	Determines the overall appearance of the page.
<b>Style Sheet</b>	Selects the CSS style sheet of the page. You can customize your own CSS and place the .css file in <b>Resources</b> folder of the project. Upon project reload, the Velocity Studio displays your .css file.
<b>Body Style</b>	Displays the background color for the Page.

## Change URL Mappings, Stylesheets, and Skins

There are several settings in Velocity Studio that enable you to modify the overall look and feel of the interface during design-time:

- URL Mapping
- Stylesheet
- Skins

### Configuring URL Mapping

The Application Page tab is found at the metadata root node of your project. Setting up this information enables you to display User Interface objects (for example, Form, Page) to Web clients by assigning them to a relative URL path that is relative to the following:

`http://<hostname>:<port>/cwf/<XX>`

where XX refers to the name of the Form or Page that you want displayed in the Web browser. For more information, see the [Metadata Root Application Page](#).

Show	Application	Security	URL	Login UI	Group	Source
Yes (<Default>)	cwf_pm.WorkflowAdministration		/WorkflowAdministration		Security	ConceptWave Metadata
Yes (<Default>)	cwt_sof.SUSMenu (Old App Menu)		/SUSMenu			Service Orchestration ...
Yes (<Default>)	applicationUI.CRMLite (CRM Lite)		/testApp		General	<Local>
Yes (<Default>)	cwa_worklist.worklistManagement (Worklis...		/worklistMgr		Security	CWA - Worklist Manag...
Yes (<Default>)	cwa_config.systemConfiguration (System ...		/configApp		System	CWA - Config
Yes (<Default>)	cwa_admin.systemAdministration (System ...		/systemAdministration...		System	CWA - System Admini...
Yes (<Default>)	cwt_pc.catalogMain (Catalog Management ...		/catalogMain5 (<Defau...		Other	ConceptWave Metadata
No (<Default>)	ui_common.ApplicationPage		/selectApp (<Default>)			UI - Common
No (<Default>)	ui_common.baseApplication		/baseApplication (<De...			UI - Common
No (<Default>)	ui_common.loginUI		/loginUI (<Default>)			UI - Common
No (<Default>)	ui_common.logoutUI		/logoutUI (<Default>)			UI - Common
Yes (<Default>)	ui_common.sessionErrorUI		/session_error (<Defa...			UI - Common
Yes (<Default>)	cwt_pcim.LifeCycleManagement		/LifeCycleManagement...			ConceptWave Metadata
Yes (<Default>)	cwt_pcim.plmConfiguration		/plmConfiguration (<D...			ConceptWave Metadata
Yes (<Default>)	cwt_sof.softMenu (Service Orchestration)		/softMenu (<Default>)			Service Orchestration ...
Yes (<Default>)	cwa_security.securityManagement (User Pr...		/securityManagement...			CWA - Security
Yes (<Default>)	cwt_pc.catalogMainApp (Catalog Managem...		/catalogMain (<Default...			ConceptWave Metadata

For applications that have a URL mapping, the stylesheet and skins properties are available at the Page object level of that application.

To set up the URL mapping, do the following:

1. In the top-level application controller, specify the relative URL path in the [URL Mapping](#) field.
2. In the left pane, click the [root node](#).
3. Click the [Application Page](#) tab.
4. Configure a mapping for the menu of your application by clicking the [URL](#) drop-down field of your application and ensuring that the relative path that you created in the [URL Mapping](#) field appears in the [URL](#) field.
5. In your application metadata, navigate to the [User Interface](#) node of the [Presentation](#) folder.
6. Navigate to the [Page](#) form.
7. Right-click and override the [Page](#) form. The Page properties appear.
8. Set the [Stylesheet](#) property to the stylesheet that you want to display.
9. From the main menu, click [File > Save](#).

### Changing the Stylesheet

The Velocity Studio system stylesheet controls the appearance of the Page design of the runtime metadata. Upon opening Velocity Studio, the [Stylesheet](#) list box displays the default stylesheet (`cwf.css [/cwf/css/cwf.css]`), which cannot be edited.

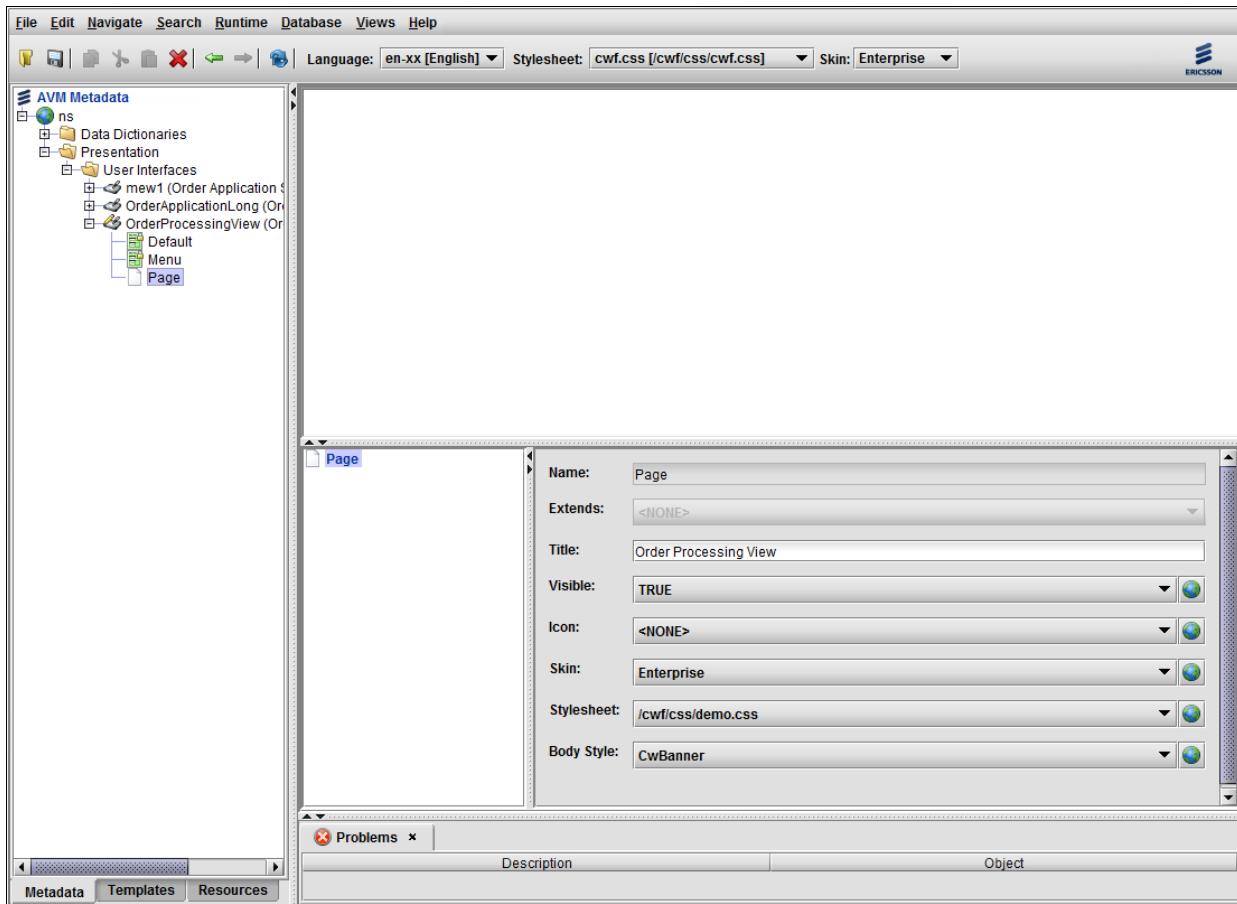
To customize the Page design, do the following:

1. In Windows Explorer, add a new CSS into the [Resources](#) folder of your project directory.
2. In Velocity Studio, open the project that contains the new CSS. Opening the project automatically loads all stylesheets.
3. From the toolbar, click the [Stylesheet](#) list box and select your new CSS.

The stylesheet is used for rendering the preview pane. Changing a stylesheet allows for look-and-feel changes at the form level. To change the overall application's look and feel, you

would change the Skin.

If you have set the User Interface object using the **Extends** parameter `com.conceptwave.system.UserInterface`, the stylesheet is available from the properties page of that Page. If you have set the User Interface object using the **Extends** parameter `com.conceptwave.system.application`, the stylesheet is available from the properties page of that Page providing that you have extended the Page. For more information about stylesheets, see [Navigation Pane - Resource](#).



## Parse Stylesheets

You can use the `CwfCssRuleSet.getTermsByPropertyName(String propertyName)` method to parse your stylesheets. The method allows you to get the following values from your stylesheets:

- Any font properties, such as font weight, font style, font family, and font size
- Colour
- Width
- Height

## Skin

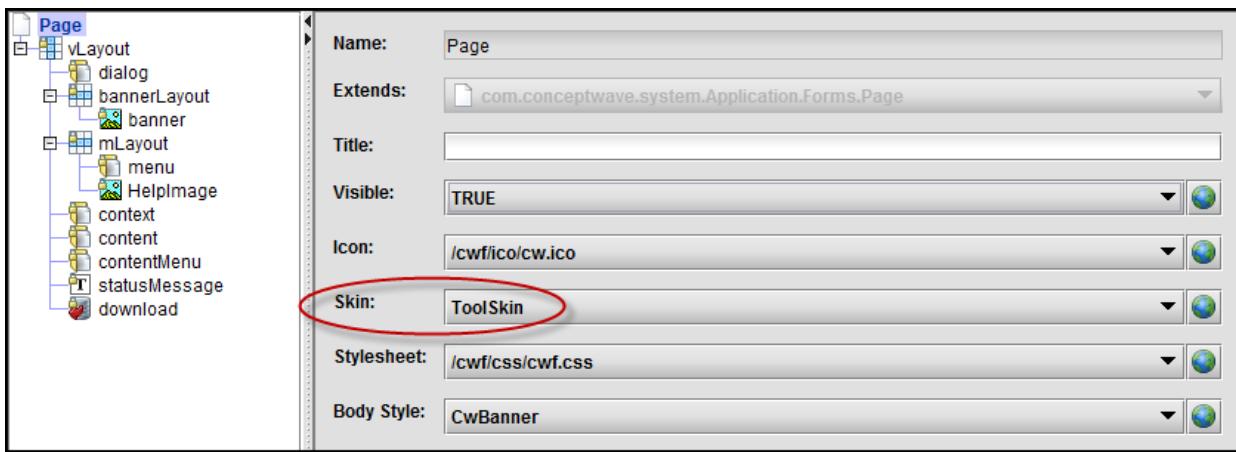
The default Skins directory is built into Velocity Studio. The skins directory contains in general a stylesheet and image files that controls the overall look and feel of the application. The Menu Bar provides a drop-down list of Skins that are available while configuring the application. Customizing the files in the Skins directory allow you to customize areas such as backgrounds, title bars, buttons, and other graphical interface elements. Skin customization allows for the overall control of the look and feel of the application including the ability to change ALL element styling including the images that are used to render the elements (for example, table or button). To customize the look and feel of a specific form, it is recommended that a customized stylesheet be used.

To customize a skin, the user would select the [Customize Skin](#) option from the File Menu. This would create a new Skins directory in the metadata and copy all the skins files and folders, including the CSS which can be modified as necessary.

There are three different ways to access the Skins:

Skin Type (accessed from)	Description	Example
Page Form	From the Details pane, set the Skin for the Page to be viewed at runtime.  <b>Note:</b> Since an application	

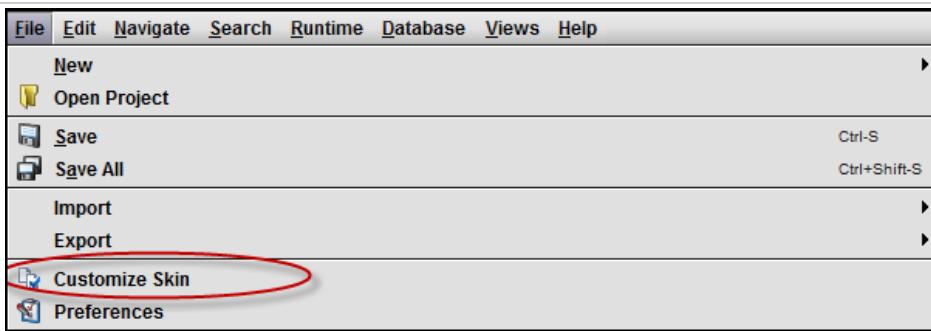
must have a URL mapping, the application skin that is defined at the Page Form is applied. This may result in some template applications to look different.



#### File menu

The **Customize Skin** menu item allows you to copy an existing Skin and rename it to a new name in the application metadata under the `skins/` folder (that is, `C:\<metadata home>\skins\`). The new Skin automatically saves and is available from the Page, toolbar or File menu. You can change the js/image files and then use this Skin for preview or runtime.

**Note:** If your customized skin is set on the page, this skin's stylesheet takes precedence, then the specified stylesheet on the page, followed by the system stylesheet.



#### From the toolbar

From the Toolbar, it is possible to change the Stylesheet and the Skin. However, this feature only provides a "preview" of the stylesheet and skin.



change. It does not actually set the Skin or Stylesheet for the form. To "permanently" set the Stylesheet and Skin for the application, the Skin and Stylesheet option is available at the Page level.

### Example: Change the Look and Position of Action in Progress Bar Using a Stylesheet

In normal applications, an Action in Progress bar appears in the top right corner when an action is taking longer than a few seconds to complete. If you want to change the look and positioning of this message, you can override them in your stylesheets as follows:

```
/* Operation In Progress message */
.CwActionInProgress {
color:black;
background-color: white;
background-image: url("../images/spinner.gif") ;
background-repeat: no-repeat;
background-position: 4px;
padding: 15px 60px 15px 60px;
font-family:Arial,Helvetica,sans-serif;
font-size:16px;
font-style:normal;
font-weight:bold;
overflow:visible;
position:absolute;
left:50%;
margin-left: -140px;
width: 220px;
top:100px;
white-space: nowrap;
border: 1px solid gray;
opacity: 1.0; /* Safari, Opera */
-moz-opacity:1.0; /* FireFox */
filter: alpha(opacity=100); /* IE */
}

.CwActionInProgressBackground {
background-color: black;
opacity: 0.7; /* Safari, Opera */
-moz-opacity:0.7; /* FireFox */
filter: alpha(opacity=70); /* IE */
}
```

These styles allow the message to appear as a full window screen. The message displays in the middle of the window.

### Example: Customize Popup Dialogs Using a Stylesheet

You can customize runtime popup dialogs using a stylesheet. The skin\_styles.css file controls the style of a popup dialog.

For popup dialogs such as warning messages, you can update the following:

- dialog.styleName - dialogBackground
- dialog.bodyStyle - dialogBody
- dialog.headerStyle - dialogHeader
- dialog.hiliteHeaderStyle - dialogHeaderHilite
- dialog.headerLabelDefaults.styleName - dialogHeaderText

For popup windows, you can customize the following:

- window.styleName - windowBackground
- window.bodyStyle - windowBody
- window.headerStyle - windowHeader
- window.hiliteHeaderStyle - windowHeaderHilite
- window.headerLabelDefaults.styleName - windowHeaderText
- window.statusBarDefaults.styleName - windowStatusBar

The image in a warning dialog can be changed by substituting the sign\_warning.png file in Velocity Studio's Resources tab.



**Notes:**

- Performing this change substitutes all occurrences of the original sign\_warning.png image. Changing the icon in a specific warning dialog is not supported.
- Changing styles for an individual warning message in a popup dialog is not supported. The global warning message style can be changed by changing the `normal` style in the `skin_styles.css` file. Note that the `normal` style is the default style for all text in the product. Changing it may cause significant appearance changes.
- Setting styles, such as `windowBody` or `dialogBody`, only take effects on a pure window object, such as the warning dialog. If you have some vertical layouts or a canvas on top of the dialog background, the dialog background ends up being covered and is not going to be shown as expected. If you want to change the background colour in this case, you must change the style for the containing layout.

## **Stylesheet Inheritance**

You can specify stylesheets on your metadata project as a whole. All applications within your metadata automatically extend this stylesheet.

You can have applications that require their own stylesheet. The metadata stylesheet is still inherited, but has a lower priority.

### **Specify the Stylesheet in Velocity Studio**

The **Style Sheet** property appears on your [metadata header's General tab](#).

### **Template Metadata**

When you create a template from your metadata, and you want to retain the styling of objects in your metadata even when viewed in a different metadata application, you must select the **Enforce** property. This property appears on your [metadata header's General tab](#).

**Note:** [URL mappings](#) continue to have a major impact in whether the stylesheet specified on both the metadata and application page levels takes effect at runtime.

## Advanced Styling

You can create element styles in your stylesheet by defining the following properties:

- Icon
  - Style property: `icon`
  - Must be in single quotation marks
- Width
  - Style property: `width`
  - Must include either px or %
- Height
  - Style property: `height`
  - Must include either px or %
- Column width
  - Style property: `colWidths`
  - Must include either px or % comma-separated values (for example, `colWidths: 70px, 70px, 30px;`)

After you have created these style properties in your metadata stylesheet, save your changes, and then open your metadata project in Velocity Studio. You can proceed to set these styles to an element.

### Icon Property

As an example, suppose that you have defined the following style in your stylesheet:

```
.cwTakeTasksMenu, .cwTakeTasksMenuOver, .cwTakeTasksMenuDown, .cwTakeTasksMenuDisabled {  
    icon: '/cwt/images/16/user_back.png';  
}
```

Ensure that the **Icon** property is applied to your element (for example, button, menu, input fields, tab, section header, and so on).

**Note:** The **Icon** property specified in Velocity Studio is the icon shown, regardless of whether the stylesheet contains an icon property.

### Width and Height Properties

An element's width and height can be taken from its **Style** property in both px and %. Its width and height can also be defined using the **Dynamic Style** property. When the dynamic style returns a different style than the original, both the width and height properties change accordingly.

When you have specified either static width or height, or both properties for an element in Velocity Studio as well as the style, the stylesheet properties take precedence.

Form frame have **Dynamic Width** and **Dynamic Height** properties. If these properties are defined, they take precedence over the stylesheet's width and height.

### Column Width Property

When you specify the column width in your stylesheet of a grid layout, the property takes into effect when the grid layout is visible in runtime.

When an element's **Column Width** property is specified in Velocity Studio, it takes precedence over the column width defined in your stylesheet.

#### Notes:

- URL mappings are needed for these styles to take effect.
- These advanced styles only apply to the stylesheet specified on the metadata header, and not the application page.
- It is recommended to that you use dynamic styles with the aforementioned properties only when absolutely needed. Overuse of this functionality can degrade UI rendering performance. Static styles are recommended over dynamic styles.

## Apply Stylesheet Changes for Tables

Whenever a table's row style is set by [cwOnVelocityRowStyle](#) (for example, UserDefinedStyle), add the corresponding style to the stylesheet (for example, tallUserDefinedStyle when applying the tall style). You also need to add the corresponding style for all appended styles from the base style (for example, tallUserDefinedStyleOver, tallUserDefinedStyleDark when applying the tall style).

**Note:** The tall style group is automatically applied when the table rows do not use the default height.

For a finder table, if the [Wrap Cells](#) property is checked, the name of the style that begins with tall is used. Otherwise, the style name without tall is used. The [Wrap Cells](#) property indicates to give the table cell flexible height.

The following is an example of styles for worklist finders with flexible heights:

```
.tallCwWLRow,
.tallCwWLRowDark,
.tallCwWLRowOver,
.tallCwWLRowOverDark,
.tallCwWLRowSelected,
.tallCwWLRowSelectedDark,
.tallCwWLRowSelectedOver,
.tallCwWLRowSelectedOverDark,
.tallCwWLRowDisabled,
.tallCwWLRowDisabledDark,

.tallCwWLRowErr,
.tallCwWLRowErrDark,
.tallCwWLRowErrOver,
.tallCwWLRowErrOverDark,
.tallCwWLRowErrSelected,
.tallCwWLRowErrSelectedDark,
.tallCwWLRowErrSelectedOver,
.tallCwWLRowErrSelectedOverDark,
.tallCwWLRowErrDisabled,
.tallCwWLRowErrDisabledDark,

.tallCwWLRowNewErr,
.tallCwWLRowNewErrDark,
.tallCwWLRowNewErrOver,
.tallCwWLRowNewErrOverDark,
.tallCwWLRowNewErrSelected,
.tallCwWLRowNewErrSelectedDark,
.tallCwWLRowNewErrSelectedOver,
.tallCwWLRowNewErrSelectedOverDark,
.tallCwWLRowNewErrDisabled,
.tallCwWLRowNewErrDisabledDark,

.tallCwWLRowNew,
.tallCwWLRowNewDark,
.tallCwWLRowNewOver,
.tallCwWLRowNewOverDark,
.tallCwWLRowNewSelected,
.tallCwWLRowNewSelectedDark,
.tallCwWLRowNewSelectedOver,
.tallCwWLRowNewSelectedOverDark,
.tallCwWLRowNewDisabled,
.tallCwWLRowNewDisabledDark {
    font-family:Verdana,sans-serif; font-size:11px; text-overflow:ellipsis;
    color:#333333;
    border-right:1px solid #E1E1E1;
}

.tallCwWLRowDark,
.tallCwWLRowErrDark {
    background-color:#F1F1F1;
}

.tallCwRowNewErrSelectedDark,
.tallCwRowNewErrSelected,
.tallCwRowNewErrSelectedOver,
.tallCwRowNewErrSelectedOverDark,
.tallCwWLRowNew,
```

```
.tallCwWLRowNewDark,  
.tallCwWLRowNewOver,  
.tallCwWLRowNewOverDark,  
.tallCwWLRowNewSelected,  
.tallCwWLRowNewSelectedDark,  
.tallCwWLRowNewSelectedOver,  
.tallCwWLRowNewSelectedOverDark,  
.tallCwWLRowNewDisabled,  
.tallCwWLRowNewDisabledDark,  
  
.tallCwWLRowNewErr,  
.tallCwWLRowNewErrDark,  
.tallCwWLRowNewErrOver,  
.tallCwWLRowNewErrOverDark,  
.tallCwWLRowNewErrSelected,  
.tallCwWLRowNewErrSelectedDark,  
.tallCwWLRowNewErrSelectedOver,  
.tallCwWLRowNewErrSelectedOverDark,  
.tallCwWLRowNewErrDisabled,  
.tallCwWLRowNewErrDisabledDark {  
    font-weight:bold;  
}  
  
.tallCwWLRowErr,  
.tallCwWLRowErrDark,  
.tallCwWLRowErrOver,  
.tallCwWLRowErrOverDark,  
.tallCwWLRowErrSelected,  
.tallCwWLRowErrSelectedOver,  
.tallCwWLRowErrSelectedOverDark,  
.tallCwWLRowErrDisabled,  
.tallCwWLRowErrDisabledDark,  
  
.tallCwWLRowNewErr,  
.tallCwWLRowNewErrDark,  
.tallCwWLRowNewErrOver,  
.tallCwWLRowNewErrOverDark,  
.tallCwWLRowNewErrSelected,  
.tallCwWLRowNewErrSelectedDark,  
.tallCwWLRowNewErrSelectedOver,  
.tallCwWLRowNewErrSelectedOverDark,  
.tallCwWLRowNewErrDisabled,  
.tallCwWLRowNewErrDisabledDark {  
    color:#b70000;  
}  
  
.tallCwWLRowOver,  
.tallCwWLRowOverDark,  
.tallCwWLRowErrOver,  
.tallCwWLRowErrOverDark,  
.tallCwWLRowNewErrOver,  
.tallCwWLRowNewErrOverDark,  
.tallCwWLRowNewOver,  
.tallCwWLRowNewOverDark {  
    background-color:#daecff;  
}  
  
.tallCwWLRowSelected,  
.tallCwWLRowSelectedDark,  
.tallCwWLRowErrSelected,  
.tallCwWLRowErrSelectedDark,  
.tallCwWLRowNewErrSelected,  
.tallCwWLRowNewErrSelectedDark,  
.tallCwWLRowNewSelected,  
.tallCwWLRowNewSelectedDark {  
    background:#d9e4f6;  
}  
  
.tallCwWLRowSelectedOver,  
.tallCwWLRowSelectedOverDark,  
.tallCwWLRowErrSelectedOver,  
.tallCwWLRowErrSelectedOverDark,  
.tallCwWLRowNewErrSelectedOver,  
.tallCwWLRowNewErrSelectedOverDark,  
.tallCwWLRowNewSelectedOver,  
.tallCwWLRowNewSelectedOverDark {  
    background-color:#bfdfc9;  
}
```

```
.tallCwWLRowDisabled,  
.tallCwWLRowErrDisabled,  
.tallCwWLRowNewErrDisabled,  
.tallCwWLRowNewDisabled {  
    color:#AAAAAA;  
    background-color:#FFFFFF;  
}
```

## Default Element Styles for Stylesheets

---

This section serves as a reference for default styles used for elements. You can use this reference to help customize your stylesheets, which contains the following areas:

- [Layout and structure](#)
- [Input elements](#)
- [Miscellaneous controls](#)
- [Other](#)

## Default Element Styles for Stylesheets - Layout and Structure

---

This section serves as a reference for default styles used for [layout](#) and [structure](#) elements.

### Layout

The following table denotes layout elements and their default styles:

Element	Default Styles
<a href="#">Section Header</a>	<ul style="list-style-type: none"><li>.sectionHeaderopened</li></ul>
<a href="#">Section Stack</a>	<ul style="list-style-type: none"><li>.sectionHeaderclosed</li></ul>

### Structure

The following table denotes the table structure element and its default styles:

Element	Default Styles
<a href="#">Table</a>	<ul style="list-style-type: none"><li>.loadingDataMessage</li><li>.offlineMessage</li><li>.emptyMessage</li></ul>

## Default Element Styles for Stylesheets - Input Elements

---

This section serves as a reference for default styles used for input elements.

Element	Default Styles
Common to all input elements	<ul style="list-style-type: none"><li>• .formTitle</li><li>• .formTitleFocused</li><li>• .formTitleDisabled</li><li>• .formCellError</li><li>• .formTitleError</li><li>• .formHint</li><li>• .headerItem</li><li>• .headerItemDisabled</li><li>• .headerItemError</li><li>• .staticTextItem</li><li>• .staticTextItemDisabled</li><li>• .staticTextItemError</li><li>• .textItem</li><li>• .textItemFocused</li><li>• .textItemDisabled</li><li>• .textItemError</li><li>• .textItemHint</li><li>• .selectItemControl</li><li>• .selectItemControlError</li><li>• .selectItemControlFocused</li><li>• .selectItemText</li><li>• .selectItemTextError</li><li>• .selectItemTextDisabled</li><li>• .selectItemTextFocused</li><li>• .comboBoxItemPendingText</li><li>• .pickListCell</li><li>• .pickListCellSelected</li><li>• .pickListCellDisabled</li><li>• .pickListCellDark</li><li>• .pickListCellSelectedDark</li><li>• .tallPickListCell</li><li>• .tallPickListCellSelected</li><li>• .tallPickListCellDisabled</li><li>• .tallPickListCellDark</li><li>• .tallPickListCellSelectedDark</li><li>• .labelAnchor</li><li>• .labelAnchorDisabled</li><li>• .labelAnchorError</li><li>• .labelAnchorFocused</li></ul>
<ul style="list-style-type: none"><li>• <a href="#">Checkbox</a></li><li>• <a href="#">Radio Button Group</a></li></ul>	<ul style="list-style-type: none"><li>• .labelAnchor</li><li>• .labelAnchorDisabled</li><li>• .labelAnchorError</li><li>• .labelAnchorFocused</li></ul>
<ul style="list-style-type: none"><li>• <a href="#">Date Field</a></li><li>• <a href="#">Date Time</a></li><li>• <a href="#">Hyperlink</a></li><li>• <a href="#">Password</a></li><li>• <a href="#">Reference Field</a></li><li>• <a href="#">Text Area</a></li><li>• <a href="#">Text Field</a></li></ul>	<ul style="list-style-type: none"><li>• .textItem</li><li>• .textItemFocused</li><li>• .textItemDisabled</li><li>• .textItemDisabledHint</li><li>• .textItemError</li><li>• .textItemHint</li><li>• .selectItemText</li></ul>

	<ul style="list-style-type: none"> <li>• .selectItemTextError</li> <li>• .selectItemTextFocused</li> <li>• .selectItemTextDisabled</li> <li>• .selectItemTextHint</li> </ul>
<a href="#">Select Field</a>	<ul style="list-style-type: none"> <li>• .selectItemText</li> <li>• .selectItemTextError</li> <li>• .selectItemTextFocused</li> <li>• .selectItemTextDisabled</li> <li>• .selectItemTextHint</li> <li>• .comboBoxItemPendingText</li> <li>• .pickListCell</li> <li>• .pickListCellSelected</li> <li>• .pickListCellDisabled</li> <li>• .pickListCellDark</li> <li>• .pickListCellSelectedDark</li> <li>• .pickListCellDisabledDark</li> <li>• .tallPickListCell</li> <li>• .tallPickListCellSelected</li> <li>• .tallPickListCellDisabled</li> <li>• .tallPickListCellDark</li> <li>• .tallPickListCellSelectedDark</li> <li>• .tallPickListCellDisabledDark</li> </ul>
<a href="#">Slider</a>	<ul style="list-style-type: none"> <li>• .sliderTitle</li> <li>• .sliderRange</li> <li>• .sliderValue</li> <li>• .rangebarTitle</li> <li>• .rangebarRange</li> <li>• .rangebarValue</li> </ul>
<a href="#">Spinner</a>	<ul style="list-style-type: none"> <li>• .selectItemText</li> <li>• .selectItemTextError</li> <li>• .selectItemTextFocused</li> <li>• .selectItemTextDisabled</li> </ul>
<a href="#">Tabset</a>	<ul style="list-style-type: none"> <li>• .tabButtonTopSelectedOver</li> <li>• .tabButtonTopOver</li> <li>• .tabButtonTopSelected</li> <li>• .tabButtonTopFocused</li> <li>• .tabButtonTopSelectedFocused</li> <li>• .tabButtonTopSelectedFocusedOver</li> <li>• .tabButtonTopSelectedOver</li> </ul>

## Default Element Styles for Stylesheets - Miscellaneous Controls

---

This section serves as a reference for default styles used for miscellaneous controls.

Element	Default Styles
<a href="#">Button</a>	<ul style="list-style-type: none"><li>.button</li><li>.buttonOver</li><li>.buttonFocused</li><li>.buttonFocusedOver</li><li>.buttonDown</li><li>.buttonFocusedDown</li><li>.buttonSelected</li><li>.buttonSelectedFocused</li><li>.buttonSelectedDown</li><li>.buttonSelectedFocusedDown</li><li>.buttonSelectedOver</li><li>.buttonSelectedFocusedOver</li><li>.buttonDisabled</li><li>.buttonSelectedDisabled</li></ul>
<a href="#">Header</a>	<ul style="list-style-type: none"><li>.headerItem</li><li>.headerItemDisabled</li><li>.headerItemError</li></ul>
<a href="#">Image</a>	<ul style="list-style-type: none"><li>.imgButton</li><li>.imgButtonOver</li><li>.imgButtonSelected</li><li>.imgButtonSelectedDisabled</li><li>.imgButtonSelectedOver</li><li>.imgButtonDown</li><li>.imgButtonSelectedDown</li><li>.imgButtonDisabled</li></ul>
<a href="#">Label</a>	CwMEIDefItem
<a href="#">Menu</a>	<ul style="list-style-type: none"><li>CwMEIDefItem</li><li>.selectedMenu</li><li>.selectedMenuFocused</li></ul>
<a href="#">Separator</a>	<ul style="list-style-type: none"><li>.formCell</li><li>.formCellFocused</li><li>.formCellDisabled</li><li>.nestedFormContainer</li><li>.nestedFormContainerFocused</li><li>.nestedFormContainerDisabled</li></ul>

## Default Element Styles for Stylesheets - Other

---

This section serves as a reference for default styles used for other elements.

Element	Default Styles
Calendar	<ul style="list-style-type: none"><li>.defaultCalendar,</li><li>.talldefaultCalendar</li></ul>
<u>Dialog</u>	<ul style="list-style-type: none"><li>dialogBackground</li><li>dialogBody</li><li>dialogHeader</li><li>dialogHeaderHilite</li><li>dialogHeaderText</li></ul>
Disabled fields	<ul style="list-style-type: none"><li>.fieldDisabled</li><li>.fieldDisabledHint</li><li>.fieldTitleDisabled</li></ul>
Error messages	<ul style="list-style-type: none"><li>.CwValidationErrorsHover</li><li>.CwHoverStyle</li><li>.cellDark</li></ul>
Menu bar	<ul style="list-style-type: none"><li>.menuBar</li><li>.menuBarOver</li><li>.menuBarFocused</li><li>.menuBarFocusedOver</li><li>.menuBarDown</li><li>.menuBarFocusedDown</li><li>.menuBarSelected</li><li>.menuBarSelectedFocused</li><li>.menuBarSelectedDown</li><li>.menuBarSelectedFocusedDown</li><li>.menuBarSelectedOver</li><li>.menuBarSelectedFocusedOver</li><li>.menuBarDisabled</li><li>.menuBarSelectedDisabled</li><li>.menuBarDisabled</li><li>.menuBarSelectedDisabled</li><li>.menuBarFocused</li><li>.menuBarFocusedOver</li><li>.menuBarSelectedFocused</li><li>.menuBarSelectedFocusedOver</li><li>.menuBarSelectedFocusedOver</li><li>.menuBarSelectedFocusedDown</li><li>.menuBarOver</li><li>.menuBarFocusedOver,</li><li>.menuBarSelectedFocused</li><li>.menuBarSelectedFocusedOver</li><li>.menuBarDown</li><li>.menuBarFocusedDown</li><li>.menuBarSelectedDown</li><li>.menuBarSelectedFocusedDown</li><li>.menuBarSelected</li><li>.menuBarSelectedFocused</li><li>.menuBarSelectedOver</li><li>.menuBarSelectedFocusedOver</li><li>.menuBarSelectedDisabled</li><li>.cwTabUnderlineImg</li></ul>

Operation in Progress message	<ul style="list-style-type: none"> <li>.CwActionInProgress</li> <li>• .CwActionInProgressBackground</li> </ul>
Window	<ul style="list-style-type: none"> <li>• windowBackground</li> <li>• windowBody</li> <li>• windowHeader</li> <li>• windowHeaderHilite</li> <li>• windowHeaderText</li> <li>• windowStatusBar</li> </ul>
Worklist finder	<ul style="list-style-type: none"> <li>• .CwWLRow</li> <li>• .CwWLRowDark</li> <li>• .CwWLRowOver</li> <li>• .CwWLRowOverDark</li> <li>• .CwWLRowSelected</li> <li>• .CwWLRowSelectedDark</li> <li>• .CwWLRowSelectedOver</li> <li>• .CwWLRowSelectedOverDark</li> <li>• .CwWLRowDisabled</li> <li>• .CwWLRowDisabledDark</li> <li>• .CwWLRowErr</li> <li>• .CwWLRowErrDark</li> <li>• .CwWLRowErrOver</li> <li>• .CwWLRowErrOverDark</li> <li>• .CwWLRowSelected</li> <li>• .CwWLRowSelectedDark</li> <li>• .CwWLRowSelectedOver</li> <li>• .CwWLRowSelectedOverDark</li> <li>• .CwWLRowDisabled</li> <li>• .CwWLRowDisabledDark</li> <li>• .CwWLRowNewErr</li> <li>• .CwWLRowNewErrDark</li> <li>• .CwWLRowNewErrOver</li> <li>• .CwWLRowNewErrOverDark</li> <li>• .CwWLRowNewErrSelected</li> <li>• .CwWLRowNewErrSelectedDark</li> <li>• .CwWLRowNewErrSelectedOver</li> <li>• .CwWLRowNewErrSelectedOverDark</li> <li>• .CwWLRowNewErrDisabled</li> <li>• .CwWLRowNewErrDisabledDark</li> <li>• .CwWLRowNew</li> <li>• .CwWLRowNewDark</li> <li>• .CwWLRowNewOver</li> <li>• .CwWLRowNewOverDark</li> <li>• .CwWLRowNewSelected</li> <li>• .CwWLRowNewSelectedDark</li> <li>• .CwWLRowNewSelectedOver</li> <li>• .CwWLRowNewSelectedOverDark</li> <li>• .CwWLRowNewDisabled</li> <li>• .CwWLRowNewDisabledDark</li> <li>• .CwWLRowDark</li> <li>• .CwWLRowErrDark</li> <li>• .CwRowNewErrSelectedDark</li> <li>• .CwRowNewErrSelected</li> <li>• .CwRowNewErrSelectedOver</li> <li>• .CwRowNewErrSelectedOverDark</li> <li>• .CwWLRowNew</li> <li>• .CwWLRowNewDark</li> <li>• .CwWLRowNewOver</li> <li>• .CwWLRowNewOverDark</li> <li>• .CwWLRowNewSelected</li> <li>• .CwWLRowNewSelectedDark</li> </ul>

- .CwWLRowNewSelectedOver
- .CwWLRowNewSelectedOverDark
- .CwWLRowNewDisabled
- .CwWLRowNewDisabledDark
- .CwWLRowNewErr
- .CwWLRowNewErrDark
- .CwWLRowNewErrOver
- .CwWLRowNewErrOverDark
- .CwWLRowNewErrSelected
- .CwWLRowNewErrSelectedDark
- .CwWLRowNewErrSelectedOver
- .CwWLRowNewErrSelectedOverDark
- .CwWLRowNewErrDisabled
- .CwWLRowNewErrDisabledDark
- .CwWLRowErr
- .CwWLRowErrDark
- .CwWLRowErrOver
- .CwWLRowErrOverDark
- .CwWLRowErrSelected
- .CwWLRowErrSelectedOver
- .CwWLRowErrSelectedOverDark
- .CwWLRowErrDisabled
- .CwWLRowErrDisabledDark
- .CwWLRowNewErr
- .CwWLRowNewErrDark
- .CwWLRowNewErrOver
- .CwWLRowNewErrOverDark
- .CwWLRowNewErrSelected
- .CwWLRowNewErrSelectedDark
- .CwWLRowNewErrSelectedOver
- .CwWLRowNewErrSelectedOverDark
- .CwWLRowNewErrDisabled
- .CwWLRowNewErrDisabledDark
- .CwWLRowOver
- .CwWLRowOverDark
- .CwWLRowErrOver
- .CwWLRowErrOverDark
- .CwWLRowNewErrOver
- .CwWLRowNewErrOverDark
- .CwWLRowNewOver
- .CwWLRowNewOverDark
- .CwWLRowSelected
- .CwWLRowSelectedDark
- .CwWLRowErrSelected
- .CwWLRowErrSelectedDark
- .CwWLRowNewErrSelected
- .CwWLRowNewErrSelectedDark
- .CwWLRowNewSelected
- .CwWLRowNewSelectedDark
- .CwWLRowSelectedOver
- .CwWLRowSelectedOverDark
- .CwWLRowErrSelectedOver
- .CwWLRowErrSelectedOverDark
- .CwWLRowNewErrSelectedOver
- .CwWLRowNewErrSelectedOverDark
- .CwWLRowNewSelectedOver
- .CwWLRowNewSelectedOverDark
- .CwWLRowDisabled
- .CwWLRowErrDisabled
- .CwWLRowNewErrDisabled

	<ul style="list-style-type: none"> <li>.CwWLRowNewDisabled</li> <li>•</li> <li>.emptyNavStyle</li> </ul>
<a href="#"><u>Worklist finders with flexible heights</u></a>	<ul style="list-style-type: none"> <li>• .tallCwWLRow,</li> <li>• .tallCwWLRowDark,</li> <li>• .tallCwWLRowOver,</li> <li>• .tallCwWLRowOverDark,</li> <li>• .tallCwWLRowSelected,</li> <li>• .tallCwWLRowSelectedDark,</li> <li>• .tallCwWLRowSelectedOver,</li> <li>• .tallCwWLRowSelectedOverDark,</li> <li>• .tallCwWLRowDisabled,</li> <li>• .tallCwWLRowDisabledDark,</li> <li>•</li> <li>• .tallCwWLRowErr,</li> <li>• .tallCwWLRowErrDark,</li> <li>• .tallCwWLRowErrOver,</li> <li>• .tallCwWLRowErrOverDark,</li> <li>• .tallCwWLRowErrSelected,</li> <li>• .tallCwWLRowErrSelectedDark,</li> <li>• .tallCwWLRowErrSelectedOver,</li> <li>• .tallCwWLRowErrSelectedOverDark,</li> <li>• .tallCwWLRowErrDisabled,</li> <li>• .tallCwWLRowErrDisabledDark,</li> <li>• .tallCwWLRowNewErr,</li> <li>• .tallCwWLRowNewErrDark,</li> <li>• .tallCwWLRowNewErrOver,</li> <li>• .tallCwWLRowNewErrOverDark,</li> <li>• .tallCwWLRowNewErrSelected,</li> <li>• .tallCwWLRowNewErrSelectedDark,</li> <li>• .tallCwWLRowNewErrSelectedOver,</li> <li>• .tallCwWLRowNewErrSelectedOverDark,</li> <li>• .tallCwWLRowNewErrDisabled,</li> <li>• .tallCwWLRowNewErrDisabledDark,</li> <li>• .tallCwWLRowNew,</li> <li>• .tallCwWLRowNewDark,</li> <li>• .tallCwWLRowNewOver,</li> <li>• .tallCwWLRowNewOverDark,</li> <li>• .tallCwWLRowNewSelected,</li> <li>• .tallCwWLRowNewSelectedDark,</li> <li>• .tallCwWLRowNewSelectedOver,</li> <li>• .tallCwWLRowNewSelectedOverDark,</li> <li>• .tallCwWLRowNewDisabled,</li> <li>• .tallCwWLRowNewDisabledDark</li> <li>• .tallCwWLRowDark,</li> <li>• .tallCwWLRowErrDark</li> <li>• .tallCwRowNewErrSelectedDark,</li> <li>• .tallCwRowNewErrSelected,</li> <li>• .tallCwRowNewErrSelectedOver,</li> <li>• .tallCwRowNewErrSelectedOverDark,</li> <li>• .tallCwWLRowNew,</li> <li>• .tallCwWLRowNewDark,</li> <li>• .tallCwWLRowNewOver,</li> <li>• .tallCwWLRowNewOverDark,</li> <li>• .tallCwWLRowNewSelected,</li> <li>• .tallCwWLRowNewSelectedDark,</li> <li>• .tallCwWLRowNewSelectedOver,</li> <li>• .tallCwWLRowNewSelectedOverDark,</li> <li>• .tallCwWLRowNewDisabled,</li> <li>• .tallCwWLRowNewDisabledDark,</li> </ul>

- .tallCwWLRowNewErr,
- .tallCwWLRowNewErrDark,
- .tallCwWLRowNewErrOver,
- .tallCwWLRowNewErrOverDark,
- .tallCwWLRowNewErrSelected,
- .tallCwWLRowNewErrSelectedDark,
- .tallCwWLRowNewErrSelectedOver,
- .tallCwWLRowNewErrSelectedOverDark,
- .tallCwWLRowNewErrDisabled,
- .tallCwWLRowNewErrDisabledDark {
- .tallCwWLRowErr,
- .tallCwWLRowErrDark,
- .tallCwWLRowErrOver,
- .tallCwWLRowErrOverDark,
- .tallCwWLRowErrSelected,
- .tallCwWLRowErrSelectedOver,
- .tallCwWLRowErrSelectedOverDark,
- .tallCwWLRowErrDisabled,
- .tallCwWLRowErrDisabledDark,
- .tallCwWLRowNewErr,
- .tallCwWLRowNewErrDark,
- .tallCwWLRowNewErrOver,
- .tallCwWLRowNewErrOverDark,
- .tallCwWLRowNewErrSelected,
- .tallCwWLRowNewErrSelectedDark,
- .tallCwWLRowNewErrSelectedOver,
- .tallCwWLRowNewErrSelectedOverDark,
- .tallCwWLRowNewErrDisabled,
- .tallCwWLRowNewErrDisabledDark
- .tallCwWLRowOver,
- .tallCwWLRowOverDark,
- .tallCwWLRowErrOver,
- .tallCwWLRowErrOverDark,
- .tallCwWLRowNewErrOver,
- .tallCwWLRowNewErrOverDark,
- .tallCwWLRowNewOver,
- .tallCwWLRowNewOverDark {
- .tallCwWLRowSelected,
- .tallCwWLRowSelectedDark,
- .tallCwWLRowErrSelected,
- .tallCwWLRowErrSelectedDark,
- .tallCwWLRowNewSelected,
- .tallCwWLRowNewSelectedDark
- .tallCwWLRowSelectedOver,
- .tallCwWLRowSelectedOverDark,
- .tallCwWLRowNewErrSelectedOver,
- .tallCwWLRowNewErrSelectedOverDark,
- .tallCwWLRowNewSelectedOver,
- .tallCwWLRowNewSelectedOverDark
- .tallCwWLRowDisabled,
- .tallCwWLRowErrDisabled,
- .tallCwWLRowNewErrDisabled
- .tallCwWLRowNewDisabled

## Page and Form Comparison

---

Although there are similarities between Page and Form objects, there are fundamental differences in functionality; specifically which Form elements that they display and how they them. From a metadata perspective, a Page is similar to a Form, but a Form has more functionality than a Page. Understanding how to configure and use these objects is a key component to configuring metadata objects in a User Interface.

### User Interface Page

A Page represents the formatting of a Web Page and displays the presentation (physical layout) for an application at runtime. Under the User Interface object, a Page represents a top-level Form. It can appear in metadata objects that have included User Interfaces such as Documents, Finders, Orders, and Data Structures. Only one Page exists for each User Interface object. A Page contains settings for Style sheets, Skins, Icons and the Body Style. A Page can be configured with additional elements that format the Page at runtime.

### User Interface Form

By default, Forms are created when a User Interface object is created. While a Page represents the formatting of a Web page, a Form defines the presentation of a fragment of a Web page (field, sections). It can appear in metadata objects that have included User Interface objects, such as Documents, Finders, Orders and Data Structures.

For more information see, [Page](#) and [Form](#).

The following information details the characteristics of a Page and a Form.

Page Characteristics	Form Characteristics
Contained within the User Interface Object with a fixed name of <b>Page</b> . The Page name cannot be renamed.	Contained within the User Interface Object. By default, when the User Interface object is added, two Forms appear with the names, <b>Default</b> , <b>Menu</b> . The names of the new Forms can be user-defined.
Cannot be referenced by other Page or Form using a Form Frame.	Can be referenced by other Forms and Page.
Can be overridden.	A default Form can be extended or overridden.
Can be requested and served to Web browser directly.	Must be rendered to Web browser by being referenced in a Page via Form Frame.
Maximum one Page for each User Interface	One User Interface may have many Forms
Defines its own Stylesheet and Skin	Must follow the Stylesheet and the Skin of the Page that references it.

## Form Elements Overview

---

A comprehensive set of form elements is available to construct rich, user-intuitive Web pages. Elements are categorized into four sections:

- **Layouts**

Layout elements determine the layout directives of the user interface such as Vertical, Horizontal, Grid, Tabs, or Sections.

- **Input Elements**

Input elements consist of field-level elements that at runtime require user input.

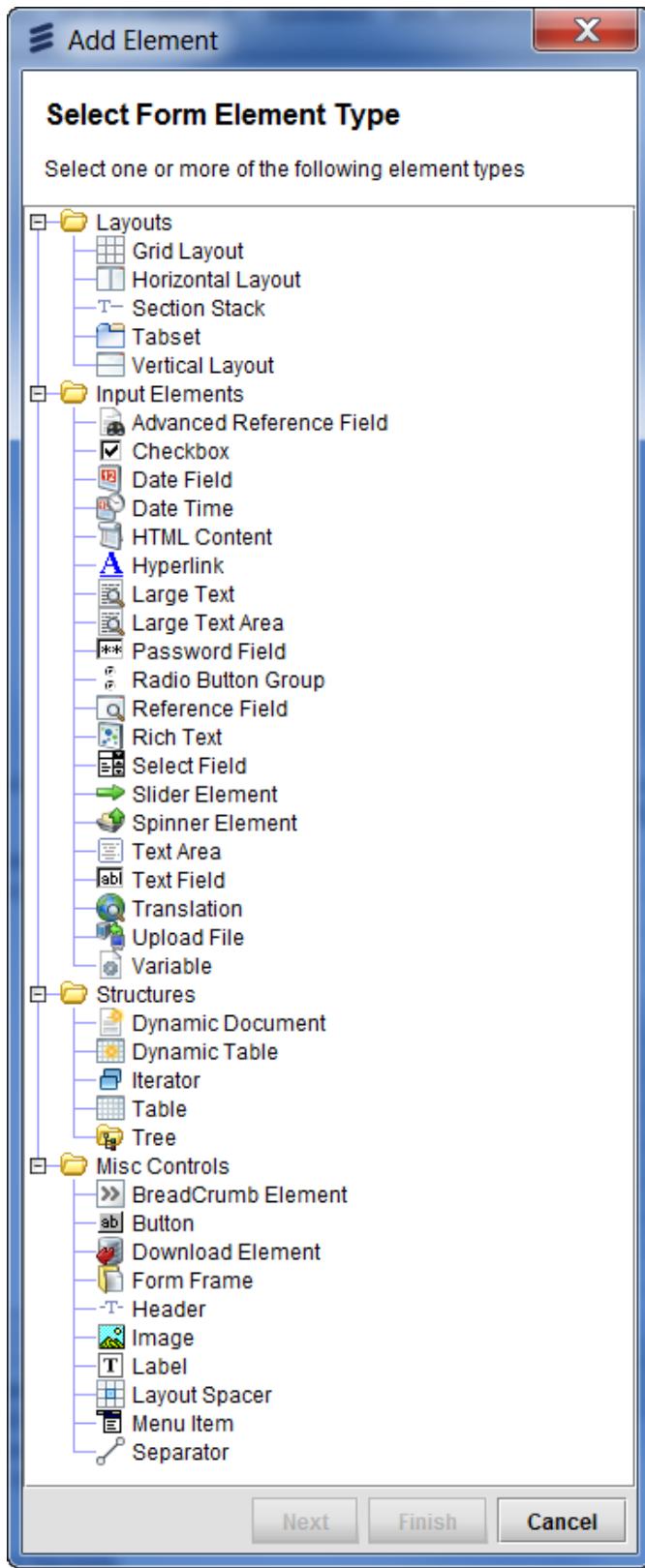
- **Structures**

Structure elements consist of Tables, Charts, Trees, and more.

- **Misc Controls**

Miscellaneous controls include other elements such as buttons, labels, images, etc.

Form element types can be found in the Element tree pane of a product application metadata (for example, within the navigation pane, navigate to <namespace>\Presentation\User Interfaces). The following is a summary of some of the available form element types:



Each element contains a set of properties that you can configure. Properties that are mandatory have their property names highlighted in blue font as shown in the following example for the Date Field element:

Name	Value
Name	DateField
Label	
Can Sort	<input checked="" type="checkbox"/>
Cell Alignment	
Cell Column Span	
Date Icon	
Display Format	
Dynamic Label	
Dynamic Label Style	
Dynamic Tooltip	

Element properties that refer to a method have an **Independent** checkbox. Selecting this checkbox indicates that multiple elements referencing the same method will receive individual values.

To share a method between multiple elements, the method must contain at least one string parameter. This string parameter represents the name of the element. When it comes to permissions, the first parameter is always \$psCondition, which is the boolean result of the privilege or state condition.

When you set a property as a shared method, the **Independent** checkbox appears next to the property in Velocity Studio appears. By default, the checkbox is set to false. To allow Velocity Studio to know when to send the name of the element currently being processed, select this checkbox.

#### Notes:

- By default, properties are set to dependent so that the product does not make unnecessary method calls. Once a method is called and a value is returned, this value remains the same until the next server request. This behaviour can only be overwritten if the property is set to **Independent**.
- The **Independent** checkbox only works at the Form level.

Name	Value
Cell Column Span	
Default Enumeration	true/false
Dynamic Label	cwPageSize
Dynamic Label Style	
Dynamic Tooltip	
Dynamic Tooltip Style	cwOnShowFields
Editable	canEditInTable
Error Icon Gap	<input type="checkbox"/>
Error Icon Orientation	

Some examples of properties that have the **Independent** checkbox include the following:

- **Dynamic Label**
- **Dynamic Label Style**
- **Dynamic Tooltip**
- **Dynamic Tooltip Style**
- **Editable**

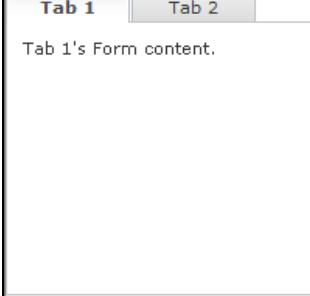
The following is an example of using the **Independent** checkbox in a permission script to return the UI element name in FinderUserInterface:

1. Create a permission script with a string parameter called *elementName*.
2. Assign the permission to the **Visible** property of a column in a finder result table.
3. After assigning the permission, select the **Independent** checkbox.
4. Run the debugger and check the permission script. The *elementName* parameter returns the UI element name.

**Note:** If you assign this permission script to the **Optional** property of a variable in a document, the variable may not have any elements bound to it. As a result, the element name is not returned. The element name is only returned when the method is used as an element property.

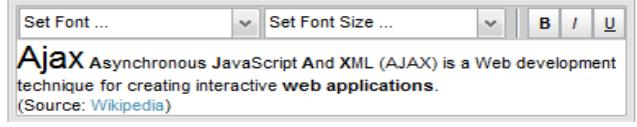
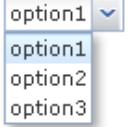
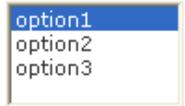
## Layouts

Element Type	Sample	Description	contain child	has Variable?	has Form?

			<b>Element?</b>		
<u>Grid Layout</u>	A <input type="text"/> B <input type="text"/> C <input type="text"/> D <input type="text"/> E <input type="text"/> F <input type="text"/> G <input type="text"/> H <input type="text"/> I <input type="text"/>	The form elements are displayed in a grid layout. The user specifies the number of columns and the system displays the elements sequentially as shown in the diagram containing 6 columns.	Yes	No	No
<u>Horizontal Layout</u>	A <input type="text"/> B <input type="text"/> C <input type="text"/>	The graphical user interface displays the form elements in a horizontal fashion. Each sequential element is placed adjacent to each other.	Yes	No	No
<u>Section Stack</u>		A Section Stack is a grouping of Sections that share the same "real estate" and can be collapsed or expanded at runtime. The Section Stack element parents the Section and Section Header elements.	Yes	No	No
<u>Tabset</u>		A Tabset consists of a set of Tab Frames or tabs.	Yes	No	No
<u>Vertical Layout</u>	A <input type="text"/> B <input type="text"/> C <input type="text"/>	The graphical user interface is presented in a vertical layout. All form elements are aligned vertically with respect to each other.	Yes	No	No

## Input Elements

Element Type	Sample	Description	contain child Element?	has Variable?	has Form?
<u>Checkbox</u>	<input checked="" type="checkbox"/> Bundle Pack	Creates a checkbox whose value is assigned at runtime.	No	Yes	No
<u>Date Field</u>	Due Date <input type="text"/> 7/1/2009 	Creates a Date and Time Field.	No	Yes	No
<u>Date Time</u>	Expiry Date/Time <input type="text"/> 6/18/2010  7	Creates a Date Field with a seconds text box.	No	Yes	No
<u>HTML Content</u>	<pre>&lt;html&gt; &lt;body&gt;&lt;h1&gt;Concept&lt;/h1&gt; &lt;/body&gt; &lt;/html&gt;</pre>	Allows HTML content to be created within the Velocity Studio Framework	No	Yes	No
<u>Hyperlink</u>	<a href="http://www.concept.com">www.concept.com</a>	Creates a Hyperlink.	No	Yes	No
<u>Large Text</u>	Province <input type="text"/> 	A text field which allows for a single line of preview text.	No	Yes	No
<u>Large Text Area</u>		A text text field which allows for multiple	No	Yes	No

		lines of preview text.			
<a href="#">Password Field</a>	Password 	Creates a textfield that masks the entry text and is ideal for passwords.	No	Yes	No
<a href="#">Radio Button</a>	<input type="radio"/> option1 Options <input type="radio"/> option2 <input type="radio"/> option3	Creates a set of radio buttons.	No	Yes	No
<a href="#">Reference Field</a>		Creates Reference field that displays a finder.	No	Yes	No
<a href="#">Rich Text Editor</a>		Create an area to perform rich text editing, such as making changes to your text's font, colour, size, alignment, and more.	No	Yes	No
<a href="#">Select Field</a>	Options   OR  Options 	Creates a listbox or a drop-down list of enumerated values.	No	Yes	No
<a href="#">Slider</a>		Creates a slider widget for numeric value selection.	No	Yes	No
<a href="#">Spinner</a>		Creates a numeric selector within a text box.	No	Yes	No
<a href="#">Text Area</a>		Creates a large text field with a label and icon.	No	Yes	No
<a href="#">Text Field</a>	Text Field Label 	Creates a text field with a label and icon.	No	Yes	No
<a href="#">Translation Field</a>	translateDT 	Creates a Translation field that displays a finder.	No	Yes	No
<a href="#">Upload File</a>		Enables the ability to upload a document at runtime.	No	Yes	No

	<input type="text" value="File :"/> <input type="button" value="Browse..."/> <input type="button" value="Upload..."/>				
<u>Variable</u>	N/A	A shortcut for creating an appropriate Element based on a chosen Variable.	No	Yes	No

## Structures

Element Type	Sample	Description	contain child Element?	has Variable?	has Form?
<u>Chart</u>		Creates a Chart for Finder result.	Yes	Yes	No
<u>Gantt Chart</u>		Creates a Gantt Chart for Finder result.	No	Yes	No
<u>Nodal Chart</u>		Creates a Nodal Chart for Finder result.	No	Yes	No
<u>Dynamic Document</u>	N/A	Enables the dynamic creation of a document leafs at runtime.	Yes	Yes	No
<u>Dynamic Table</u>	N/A	Enables the creation of tables dynamically at runtime	Yes	Yes	No
<u>Iterator</u>		Enables repeated configuration of the User Interface using an array.	No	Yes	No
<u>Table</u>		Creates a Table for Finder result.	Yes	Yes	No
<u>Tree</u>		Creates a Tree to enable navigation of hierarchical dataset.	Yes	Yes	Yes

## Misc Controls

Element Type	Sample	Description	contain child Element?	has Variable?	has Form?
<u>Bread Crumb</u>		This element provides a navigation control that accumulates the history of navigation in a delimited set of links, which allows easy navigation to previously visited pages	No	Yes	No
<u>Button</u>		Creates a button.	No	Yes	No

<u>Cell Spacer</u>		Add empty empty space(s) in a Grid Layout that spans columns. The example shows a cell spacer that spans four (4) columns.	No	No	No
<u>Dialog</u>		Creates a dialog box that can be configured to display a warning message and user action button.	No	Yes	No
<u>Form Frame</u>	N/A	References a form within a web page that can dynamically change at runtime based on methods and variables.	No	Yes	Yes
<u>Group</u>		Group element behaves like a Section Stack element but is parented by the Grid Layout and contains only one header and one stack.	Yes	No	No
<u>Header</u>	<b>Header 1</b>	Creates a header within a layout on a Web page.	No	No	No
<u>Image</u>		Attaches an Image to a Web page.	No	Yes	No
<u>Label</u>	Label1	Creates a label.	No	No	No
<u>Layout Spacer</u>		Add empty space in the Vertical and Horizontal Layouts.	No	No	No
<u>Menu</u>		Creates a Menu item.	Yes	No	No
<u>Row Spacer</u>		Add empty row(s) space between rows in a Grid Layout. The example shows a row spacer between field elements A and C.	No	No	No
<u>Section</u>		A section contained within a Section Stack. Used to divide logical blocks of data that share the same "real estate"	Yes	Yes	Yes
<u>Section Header</u>		The header of a Section element that has limited functionality other than to be used to display a banner and used to expand or collapse a Section.	Yes	No	No

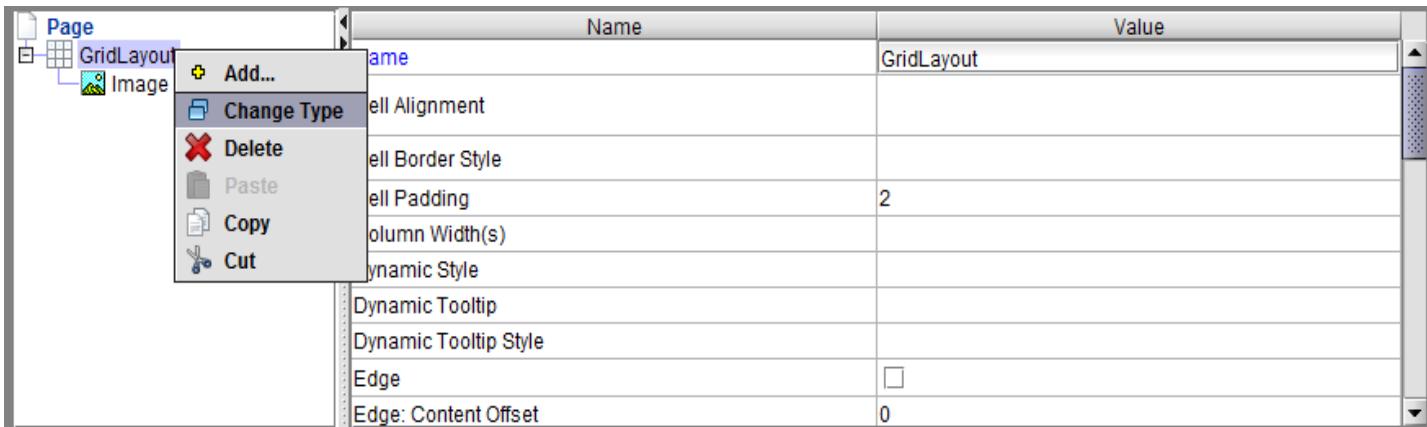
<u>Separator</u>		The Separator is a static element that is used to place a separator (space/line) between elements.	No	No
<u>Tab Frame</u>		Creates a tab within the Tabset. The contents of the tab is that of a referenced Form.	No	Yes

## Change Type

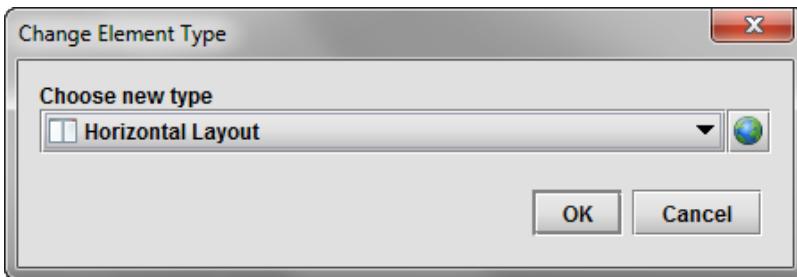
The change type function allows you to change between control types. This change includes replacing the control that is in place, and copying over any pertinent settings such as the label, onClick method, column span, and so on. This function copies all common properties from the old element (that is, the changing-from element) to the new changing-to element. You can manually remove unexpected properties after changing the type. Where the new control does not support a setting, the setting is ignored. Where a method or action is not supported, a warning message appears, allowing you to choose whether to proceed.

The following example shows how to use the change type feature to change the Grid Layout element:

1. Right-click the **GridLayout** element and select **Change Type** from the menu.

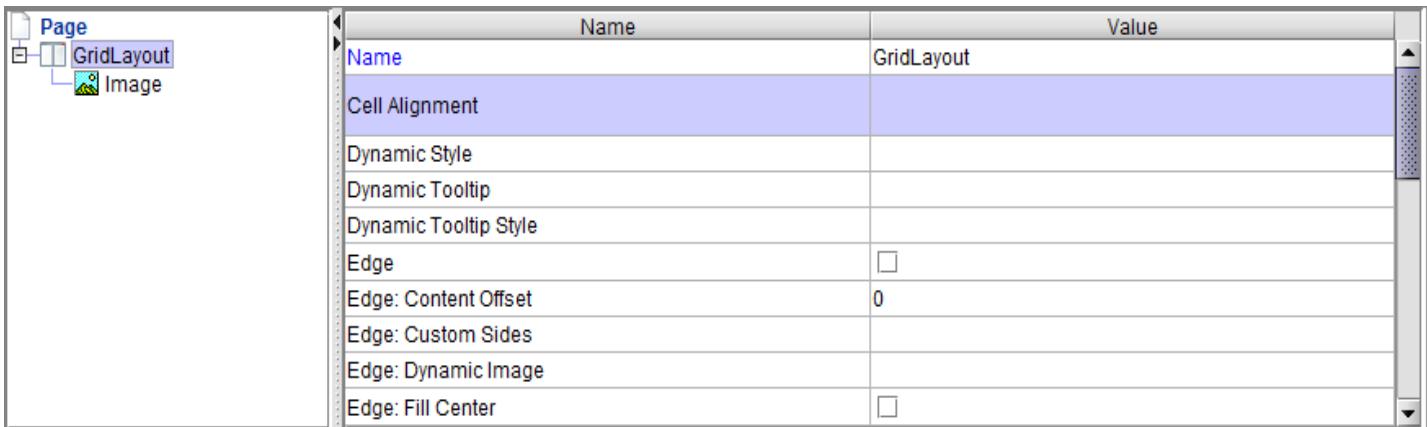


2. The Change Element Type dialog appears. Click the drop-down menu and select an element from the list. You can also click the icon and enter the element name in the field provided. In this example, the **Horizontal Layout** element will be used instead of the **Grid Layout** element.



Click the **OK** button to continue.

3. The Grid Layout element has changed to the Horizontal Layout element. The old properties of the Grid Layout element are copied to the Horizontal Layout element (for example, the **Name** field continues to be called **GridLayout**).



4. Save your metadata.

The list of applicable change types depends on the element's parent, overlay, and type:

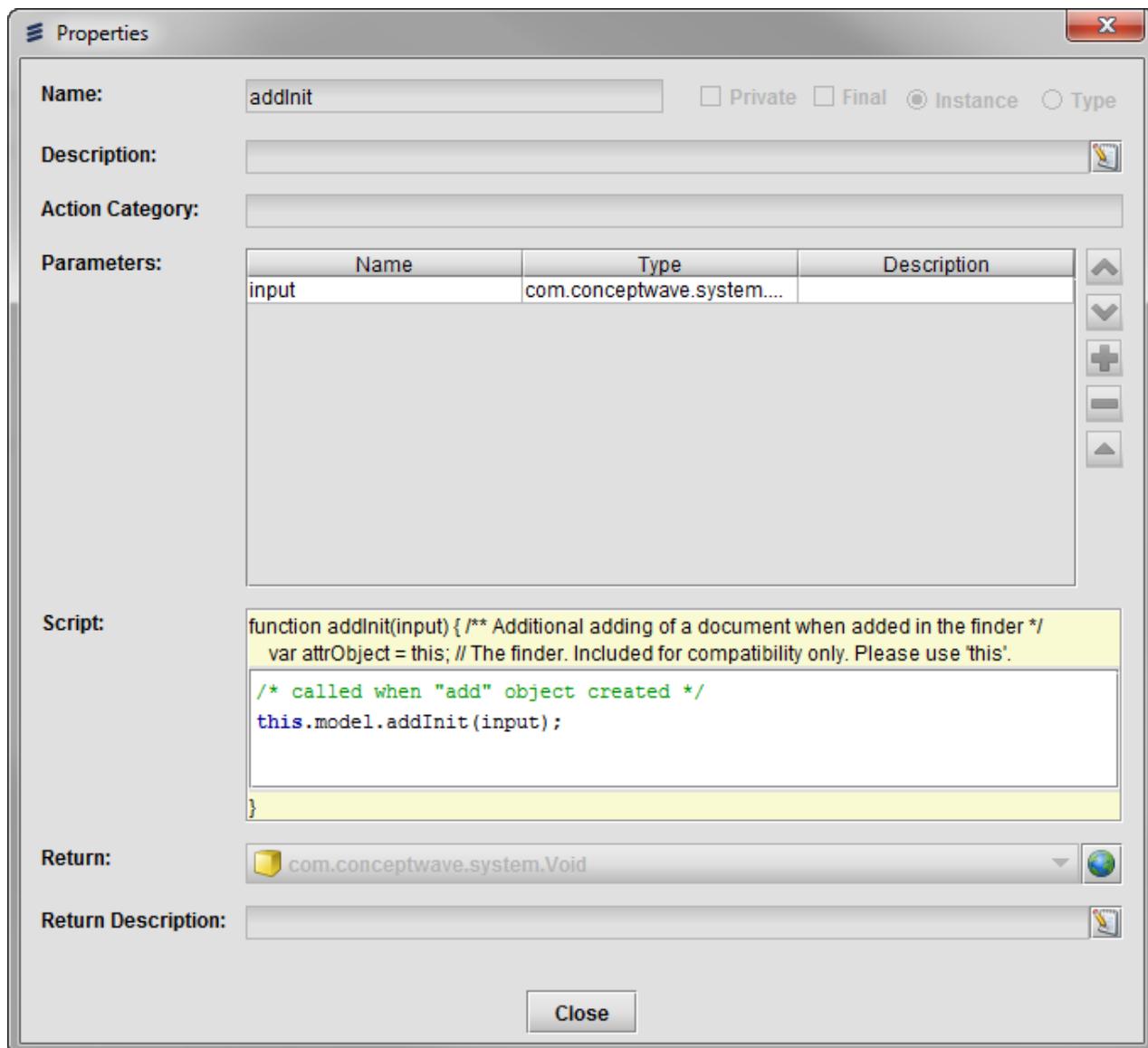
- If the element's variable is null, the **Variable** type is not shown.
- When copying the old element's properties to the new one, if the value is the variable path, a check is performed to determine whether this variable is applicable. If there is no supported value, a warning message appears, asking whether you want to proceed with the change.
- If the size of the applicable change type list is one (that is, itself), the Change Element Type dialog is not shown.
- A layout element can only be changed to another layout element. Note the following details pertaining to layout elements:
  - A Grid Layout element can be changed to other layouts. However, other layouts that have layout or data source container children cannot be changed to the Grid Layout element.
  - A Section Stack element that has children cannot be changed to other layouts. Other layouts that have children cannot be changed to a Section Stack element.
- A data source container element can only be changed to a data source container element. Note the following details:
  - A Dynamic Table cannot be changed to another container. However, other containers can be changed to a dynamic table.
  - An Iterator cannot be changed.
- A Menu Item element that has children cannot be changed to another control type.
- For all other control types that have children, these types cannot be changed to other types.
- For all other control types that do not have children, they can be changed to other types, depending the element's parent and overlay.

## Click Method and Variable Properties Lookup Button

When specifying an element's **Click Method** or **Variable** properties, a lookup button appears in the **Value** column.

Name	Value
Name	Button
Label	
Click Method	addInit
Action Auditor	
Autofit	<input type="checkbox"/>
Can Sort	<input checked="" type="checkbox"/>
Cell Alignment	
Cell Column Span	
Disabled	

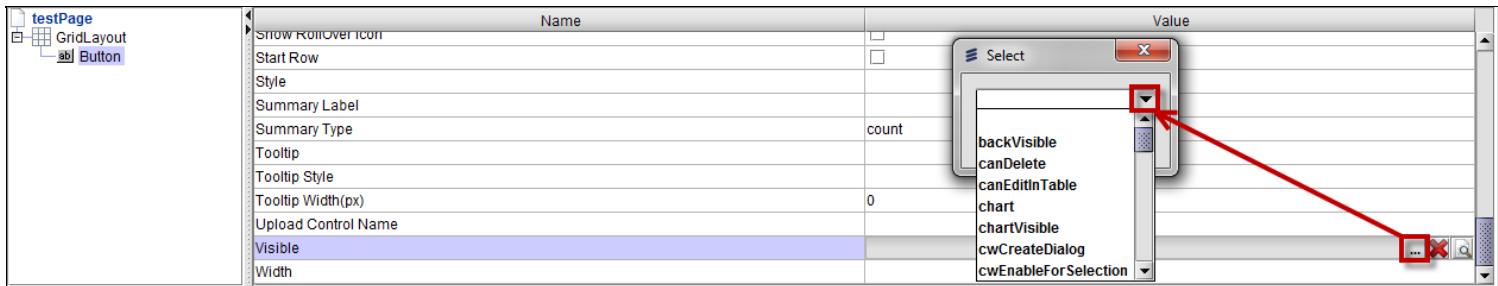
Clicking the lookup button launches the Properties dialog, allowing you to view definition details on either the **Click Method** (see the example that follows) or **Variable** property.



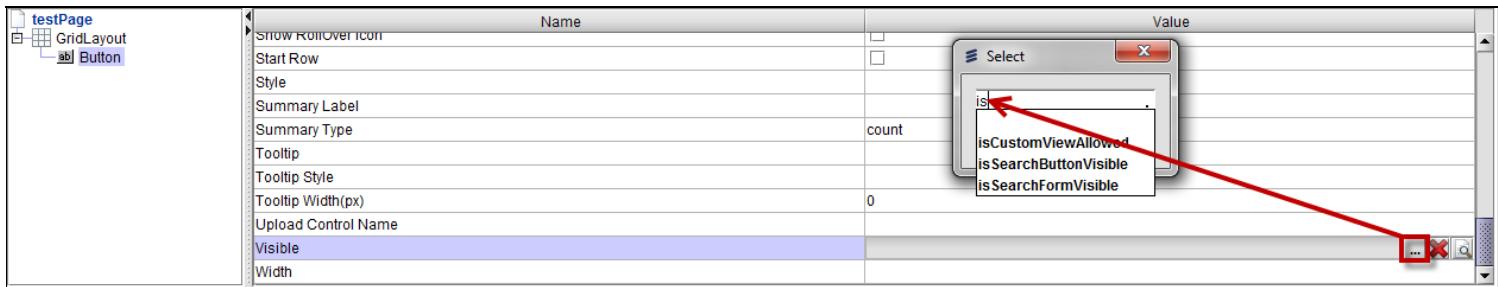


## Value Selection for Editable, Variable, and Visible Properties

When you click a form element property with an ellipsis button, such as the **Editable**, **Variable**, or **Visible** properties, a dialog appears, which allows you to specify the value that you want. A list is available for you select the method, variable, path, or value that you want.



This list of values appears in alphabetical order. You can also quickly narrow down the value that you want by entering its first few letters and then select the value from the shorter list that appears.



## Advanced Reference Field

The Advanced Reference Field element is similar to the [Reference](#) element, except that it has additional shortcut buttons next to the field.



At runtime, the Advanced Reference Field contains the following buttons:

Button Function	Button	Description
<b>Finder</b>		Clicking this button still performs the onRefClick action to show the finder to select or deselect values.
<b>Clear</b>		Clicking this button clears the value for the reference field on the browser side. This button is disabled when the field is read-only.
<b>Detail</b>		Clicking this button shows the detail user interface in a dialog.

### Customization

The Advanced Reference Field has its **Style** property set to **cwAdvancedReference**. This style contains the following:

- icon (set to finder.png)
- cwClearIcon (set to document\_delete.png)
- cwDetailIcon (set to document\_view.png)

You can customize the buttons for a single reference field by doing the following:

1. Create your own style, set the aforementioned properties, and then set the style on the Advanced Reference Field.
2. Clear the **Style** property.
3. Set both the **Clear Icon** and **Detail Icon** properties.

To change the buttons for all reference fields in the application, complete these steps:

1. In the application's stylesheet, add the cwAdvancedReference style and set the properties that you need.
2. Substitute other images for the default button images being used.

### Summary

references Variable	Yes
references Method	Yes
references Form	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
<b>Name</b>	Mandatory	Name of the Label.
<b>Label</b>	Mandatory	Visible label of the Label. If empty, there is no Label label. Default is empty.
<b>Variable</b>	Mandatory	Choose from the list of Variables available in the User Interface to bind to this element. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details. <b>Note:</b> The <b>Variable</b> property cannot bind to data types belonging to a user interface. The data type used must belong to a document.
		This property indicates whether the focus moves to the next input field when the permitted length of the data is reached. By default, this checkbox property is selected. The next field is considered to be any input field (for example, text, select field, etc.) that is visible and editable (that is, not disabled).

<b>Auto-focus to next field</b>	Optional	The behaviour applies to the following: <ul style="list-style-type: none"><li>• Masked text field, where the <b>Display Format</b> defines the permitted length</li><li>• Regular text field, where the <b>Data Length</b> indicates the permitted length</li></ul>
<b>Can Sort</b>	Optional	Specifies whether the Reference Field can be sorted (true) or unsorted (false). The default is set to true. This option is available when the Reference Field appears in a table.
<b>Cell Alignment</b>	Optional	Controls the horizontal and vertical positioning of the Label in the page or parent Element. Default is undefined, which behaves as if it is top-left
<b>Cell Column Span</b>	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Label shall occupy in the Grid Layout, to the right of current cell. Default is empty, which occupy one column.
<b>Clear Icon</b>	Optional	Denotes the image used for the <b>Clear</b> icon button.
<b>Data Length</b>	Optional	Specifies the maximum number of characters that can be entered into the text box. Default is 0, which means no character restrictions.
<b>Detail Icon</b>	Optional	Denotes the image used for the <b>Detail</b> icon button.
<b>Disable Paste</b>	Optional	By default, the element allows for pasting content in its field. To disable this functionality, select this property's checkbox.
<b>Dynamic Label</b>	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
<b>Dynamic Label Style</b>	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the <b>Variable</b> property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
<b>Dynamic Tooltip</b>	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.  When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Editable</b>	Optional	Specifies whether the Reference Field is editable (true) or read-only (false). Default is NONE, which is editable. Setting this property to <i>True</i> enables editing of this element when it appear within a Table.  <b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.
<b>Editable Visual Key</b>	Optional	Specify whether the text field accepts an editable visual key (set this property to TRUE) or not (property's value is set to FALSE).
<b>Enumeration Reference</b>	Optional	This property provides a list of all data types within the metadata that have either an enumeration or a reference finder defined.  At runtime, when the form is displayed, the value is still retrieved from the <b>Variable</b> property. The variable set on the fields can be primitive strictly for storing values. When you click the reference icon, the product uses the enumeration reference data type to find the finder user interface to display in a dialog.  See the <a href="#">example</a> on how to use this property for details.
<b>Error Icon Gap</b>	Optional	Select this property's checkbox to make a gap for the validation icon. By default, this property is false.  <b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Group by In Table</b>	Optional	Displays the Reference Field data in Finder results. The Reference data is displayed in a column with elements that are grouped with the same name.
<b>Height</b>	Optional	Height to be taken up by the Label; can be in integer which is in pixels, or in percentage (for example, "50%") of page height or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>Hint</b>	Optional	Indicate a hint for the field, if required. This field is translatable.
<b>Icon</b>	Optional	If an icon is selected, the icon appears in the Label, besides the label. Default is NONE.
<b>Icon Action</b>	Optional	Only applicable when an Icon is specified. Choose from the list of Methods available in the User Interface to bind to the Icon. The Method is invoked when a user clicks on the Icon.
<b>Include Icon in Tab Order</b>	Optional	The property allows you to optionally exclude the icon button from the tab order.

<b>Include in Tab Order</b>	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>• When set to &lt;Default&gt;, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>Label Align</b>	Optional	<p>Click the <b>Value</b> field and select from one of the following alignment options for your label:</p> <ul style="list-style-type: none"> <li>• <b>Left</b></li> <li>• <b>Centre</b></li> <li>• <b>Right</b></li> </ul>
<b>Label Orientation</b>	Optional	Choose the location of the Label. Options include top, left or right.
<b>Label Style</b>	Optional	Choose from styles in the selected style sheet on the toolbar, which defines the styling of the Label such as font color. Default is NONE.
<b>Multi Line</b>	Optional	Indicate whether the text field contains multiple lines. By default, this property is not selected.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Run Trigger</b>	Optional	<p>When data is changed in the Advanced Reference Field, and the dialog box becomes out-of-focus, the action of the selected Method is invoked immediately after the reference is selected in the popup finder. This property can be bound to any method of the UserInterface data structure.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>• You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>• You click a menu.</li> <li>• You change your selection in a navigation tree.</li> <li>• You click another tab of a tabsheet.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• Field triggers do not fire on save. However, document triggers do.</li> <li>• As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> <li>• The trigger does not fire for editable visual key types when entering a visual key value immediately in the reference text field.</li> </ul>
<b>Show Hint</b>	Optional	By default, this property is selected, meaning that the hint is automatically shown. To hide the hint, deselect this property.
<b>Show Label</b>	Optional	This property determines whether the label is displayed on the element. Choose from <i>True</i> or <i>False</i> . Default is True which means that the label is displayed.
<b>Start Row</b>	Optional	Applicable only when the parent element is a Grid Layout. If true, the Label starts in a new row of the grid. If false, it is placed in the current cell as-is. Default is /default, which does not start a new row.
<b>Style</b>	Optional	Defines the styling of the label such as color and font. Default is NONE.
<b>Text Align</b>	Optional	Specifies the alignment of the label; it can be <i>left</i> , <i>center</i> , or <i>right</i> . Default is empty, which is left-aligned.
<b>Textbox Style</b>	Optional	<p>Choose from styles in the selected style sheet on the toolbar, which defines the styling of the text box of the Textfield such as color and font. Default is NONE.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the MyStyle style to the <b>Textbox Style</b> property, the stylesheet must have MyStyleDisabled to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Label. Default is empty, which means no tooltip for the Label.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the label. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Visible</b>	Optional	Determines whether the Label is visible or not. Choose true (visible) or false (not visible) or from the list of Methods available in the User Interface to bind to the Field.
<b>Width</b>	Optional	Width to be taken up by the Label; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

**Note:** Displaying visual keys for multi-select reference fields does not work with the editable visual key format.

## Button

This element creates a button field within the User Interface that can display a label and icon. Clicking the button can invoke a user-defined method. This button can be enabled or disabled, or hidden or shown at runtime based on user permissions. The following is an example of a button with a save icon and label.



### Summary

references Variable?	No
references Method?	Yes, Click Method and On Enter.
references Form?	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Button.
Label	Mandatory	Visible label of the Button. If empty, the Button has no label. Default is empty.
Action Auditor	Optional	Specify the Action Validator object for this button's <b>Action Auditor</b> property. During runtime, when you click this button, the associated logging action is also invoked, provided that the <b>Enable action logging</b> checkbox from the System <b>Logging</b> tab has been selected.
Autofit	Optional	Select this property's checkbox to automatically fit the image's label. By default, the image width is set at 100 px. To have the button fit both the label and icon, set the width to a small value (for example, 10 px) and select the <b>Autofit</b> checkbox.
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Button in the page or parent Element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Button occupies in the Grid Layout, to the right of current cell. Default is empty, which occupy one column.
Cell Row Span	Optional	<p>This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
Click Method	Optional	Choose from the list of methods available in the User Interface to bind to this Button. The method is invoked when a user clicks the Button. Use the property's <a href="#">lookup button</a> to view property details.
Disabled	Optional	<p>This property controls the user's ability to click the button to invoke a click method. Choose from <i>True</i>, <i>False</i> or a permissions method. Setting this property to <i>True</i> means that the user is unable to click the button. The permissions method will determine if the button is enable or disabled at runtime.</p> <p>When this property is set to true and the <b>Icon</b> property is set, <b>icon_Disabled</b> must be provided. Otherwise, it appears as a broken image.</p> <p><b>Note:</b> This property is only available at runtime. It does not work in the preview pane.</p>
Dynamic Icon	Optional	Use this property to display a dynamic icon on the button, as opposed to a static "icon" property.
Dynamic Label	Optional	This property allows you to provide a dynamic label for a button. See the <a href="#">Notes</a> section for details.
Dynamic Style	Optional	Select a dynamic style from the drop-down menu to indicate that the style is applied to the entire button.

<b>Dynamic Tooltip</b>	Optional	<p>Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i>. The static Tooltip property is considered as default until the action returns a value.</p>
<b>Dynamic Tooltip Style</b>	Optional	<p>When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.</p>
<b>Error Icon Gap</b>	Optional	<p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwf/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
<b>Error Icon Orientation</b>	Optional	<p>This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.</p>
<b>Height</b>	Optional	<p>Height to be taken up by the Button; can be in integer which is in pixels, or in percentage (for example, "50%") of page height or parent element.</p>
<b>Icon</b>	Optional	<p>Choose an image file to appear as an icon on the button. The default is none.</p>
<b>Icon Orientation</b>	Optional	<p>This property determines the orientation of the icon compared to the label on the button. Choose from <i>Top</i>, <i>Left</i> or <i>Right</i>. The default is <i>Left</i>, which means that the icon appears to the left of the label.</p>
<b>Icon Spacing</b>	Optional	<p>This property accepts numerical values in pixels (px), allowing you to customize the gap between the icon and the button label.</p>
<b>Image Button</b>	Optional	<p>Select this checkbox to render the button using either the <b>Image: Static</b> or <b>Image: Dynamic</b> background image. The background image refers to the stretch image that is being used for the entire button, which would replace the current grey stretch image that is used to display a button. To display an actual image on top of the grey image, use the style for the button and specify an image button background.</p>
<b>Image: Dynamic</b>	Optional	<p>Use this property to specify the base URL for the image. By default, the image button consists of three image parts:</p> <ul style="list-style-type: none"> <li>• A start image (displayed at the top or left)</li> <li>• A scalable central image</li> <li>• An end image displayed at the bottom or right</li> </ul> <p>The images displayed in the image button are derived from this property in the following way:</p> <ol style="list-style-type: none"> <li>1. When the button is in its standard state, the <i>_start</i>, <i>_end</i>, and <i>_stretch</i> suffixes are applied to the src (that is, before the file extension). By default, the images displayed are <i>button_start.gif</i> (sized to be this.capSize by the specified width of the <i>stretchImgButton</i>), <i>button_stretch.gif</i> (stretched to the necessary width), and <i>button_end.gif</i> (sized the same as the start image).</li> <li>2. As the button's state changes, the images have suffixes appended before the <i>_start</i>, <i>_end</i>, and <i>_stretch</i> suffixes to represent the button state. Possible states are as follows: <ul style="list-style-type: none"> <li>◦ Down</li> <li>◦ Over</li> <li>◦ Focused</li> <li>◦ Disabled</li> </ul> </li> </ol> <p><b>Note:</b> Selected and Focused are compound states that may be applied in addition to Down, etc.</p> <p>As an example, the centre piece of a <i>stretchImgButton</i> with the mouse hovering over it might have the URL <i>button_Over_stretch.gif</i>.</p> <p><b>Note:</b> It is recommended that media be present for each possible state of the <i>_start</i>, <i>_end</i>, and <i>_stretch</i> images.</p>
<b>Image: Label Padding</b>	Optional	<p>Use this property to specify the right and left padding for the label inside the button.</p>

<b>Image: Static</b>	Optional	Specify a static image from the drop-down menu for this property.
<b>Include in Tab Order</b>	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>On Enter</b>	Optional	The system calls any script defined in this property. This method is invoked when the Button is focused on the Web page and the user presses the <b>Enter</b> key.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Show Down</b>	Optional	Select this property's checkbox to allow a button to have its style changed when you click the button. See the section following this table that provides an example on how to use this property.
<b>Show Focused Style</b>	Optional	Select this property's checkbox to show the focused style when a button receives focus. This property only takes effect when the <b>Style</b> property is set, and both the standard style and focused style must be available in the stylesheet (that is, <code>&lt;StandardStyleName&gt;</code> and <code>&lt;StandardStyleName&gt;Focused</code> ).
<b>Show Label</b>	Optional	This property determines whether the label is displayed on the button. Choose from <i>True</i> or <i>False</i> . Default is <i>True</i> which means that the label is displayed.
<b>Show Rollover Icon</b>	Optional	This setting allows you to change the icon of a button when the mouse pointer hovers over it. The Button element requires that you set the <b>Icon</b> property (for example, ensure that image.png and image_Over.png are present in your resource directory), and that you select the <b>Show Rollover Icon</b> property. At runtime, hovering your mouse pointer over the button icon shows image_Over.png.
<b>Show Upload Button</b>	Optional	This property determines whether the Upload button is displayed on the form. Choose from <i>True</i> or <i>False</i> . Default is <i>True</i> which means that the button is displayed. It is possible to set this property to <i>False</i> and add a Button element. Associate the Button element with the Upload file by using the Upload Control Name property on the Button element. These properties are only visible when the form exists under an UploadPopup User Interface type.
<b>Start Row</b>	Optional	Applicable only when the parent element is a Grid Layout. If true, the Button appears in a new row of the grid. If false, it is simply placed in the next available cell. Default is <i>False</i> , which means that the Button element does not appear on a new row.
<b>Style</b>	Optional	<p>Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the button such as positioning and color. Default is none.</p> <p><b>Note:</b> CSS styles only appear in the property dropdown if the following have been provided:</p> <ul style="list-style-type: none"> <li>styleName</li> <li>styleNameDown</li> <li>styleNameOver</li> <li>styleNameDisabled</li> </ul>
<b>Tooltip</b>	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers over the Button. Default is empty, which means that a tooltip has not been defined.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the Button element. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
<b>Width</b>	Optional	Width to be taken up by the Button; can be in integer which is in pixels, or in percentage (for example, "50%") of page width or parent element.

\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a

width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

#### Notes:

- Be sure to create and assign a Method to the Button so that the desired action is achieved when the Button is clicked.
- You can provide a dynamic label for a button by following these steps:
  1. Create a new variable of type String and set its initial value. Proceed to create a new method to change this variable's value.
  2. In the User Interface, create a new button, set a static label, and then set the **Click Method** property to the method you just created.
  3. Set the **Dynamic Label** property to the new String variable you created.
  4. When you click this button, it changes to what the method has set its value to be.

## Change the Button's Style When Clicked

To change a button's style when it is clicked, do the following:

1. Create a style for the button. Ensure you have at least the following styles:
  - Over
  - Down
  - Disabled
2. Set the Down style to be a different style, such as its background colour such as the following:

```
.myButton{  
background-color: blue;  
}  
.myButtonDown{  
background-color: red;  
}  
.myButtonOver{  
background-color: blue;  
}  
.myButtonDisabled{  
background-color: blue;  
}
```

3. Select the button element's **Show down** property.
4. At runtime, when you hold the mouse down on the button, it changes style.

## Create a Button and Text Fields on a Grid Layout

The following example shows how to create a button using its **Cell Row Span** property and text fields on a grid layout, where the button aligns with the second text field:

1. Create a grid layout with the column number as **2**, and set the values to **Column Width(s)** and **Width** properties.
2. Add a Button element under the grid layout and set the element's **Cell Row Span** property to **3**.
3. Add another three Text Field elements and select the **Start Row** field's checkbox for the second and the third Text Field elements.
4. Save your changes. In both runtime and when you preview your form, the button appears, along with three text fields to its right. The button's position aligns with the second text field.

## Element - BreadCrumb

---

This element provides a navigation control that accumulates the history of navigation in a delimited set of links, which allows easy navigation to previously visited pages. Breadcrumb page navigation is the page transition that occurs when you click a section of the breadcrumb or an item in a breadcrumb dropdown.

Property	Mandatory/Optional	Comment
Name	Mandatory	This property indicates the name of the breadcrumb element.
Click Method	Mandatory	This property allows you to choose a method from the list of methods available for the User Interface (UI). The method is invoked when you click the breadcrumb. This method receives two parameters: <ol style="list-style-type: none"><li>1. Index of the select breadcrumb.</li><li>2. If selected breadcrumb is an array, the parameter is an index of the selection within the array. Otherwise, it is the first parameter.</li></ol>
Variable	Mandatory	This property allows you to select a variable. The variable must be of type <code>com.conceptwave.Object</code> that is initialized to <code>CwfScriptBreadCrumb</code> at runtime.

### Notes:

- It is recommended that only one UI should contain the breadcrumb element and hold the breadcrumb instance object in a local variable. The breadcrumb then can be used to control the flow of pages in a single content area of the UI.
- The UI method that controls the content change must call the `pushBreadCrumb()` method. Use the UI object instance of the content area as the first parameter of the `pushBreadCrumb( )` API.
- When the breadcrumb page navigation occurs, the method selected in the *Click Method* property of the breadcrumb element is invoked. It is recommended to invoke the `popToBreadCrumb( )` API within this method. Set the content of the UI as the result of this method.

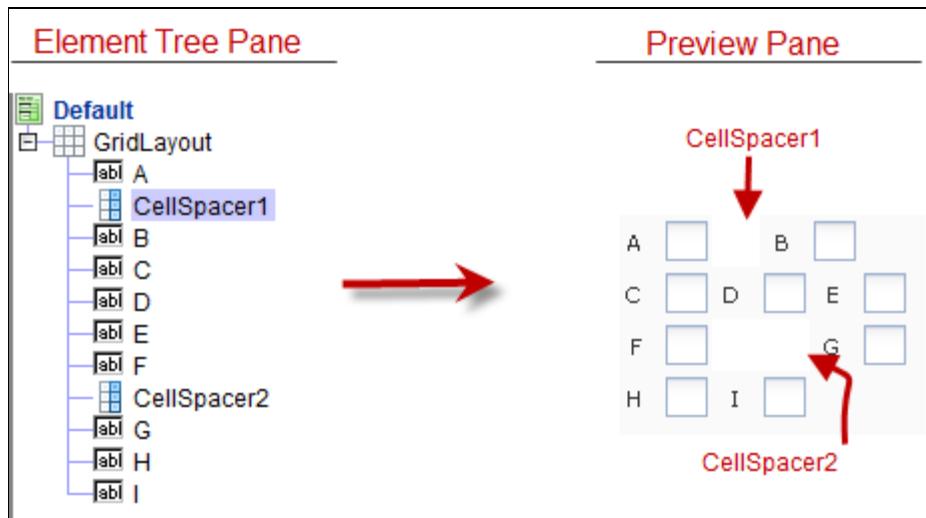
## BreadCrumb with Dropdown Options

BreadCrumb with dropdown options are best used for the pages with several options available for the flow to the next page. When the page navigation occurs, the `pushBreadCrumb()` method is called with the first parameter being an array of all possible options available, and the second parameter being the index of the actually selected option within the array. When the breadcrumb page navigation occurs on a dropdown, the method selected in the *Click Method* property of the breadcrumb element is invoked, that in turn invoke the `popToBreadCrumb()` and `setSelectionIndex()` methods initialized in this selected method. Using the parameter value given to the selected method, that indicates the index of the selected option, you can retrieve the UI object resulted from the `popToBreadCrumb()` API in the array.

## Cell Spacer

Cell Spacer inserts blank cells within a Grid Layout.

Place the cell spacer anywhere within the Grid Layout that you would like to insert a blank space or multiple spaces within a row. The figure below shows two cell spacers: 1) Cell Spacer1 spans one cell and 2) Cell Spacer2 spans two cells.



### Summary

references Variable?	No
references Method?	Yes
references Form?	No
Allowable Child Elements	None

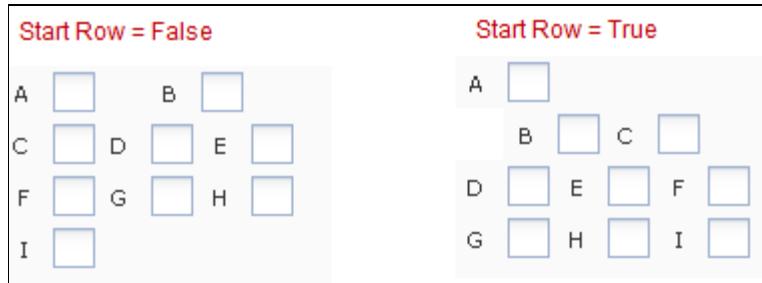
### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Column Spacer.
Cell Column Span	Optional	This defines the number of cells that the Cell Spacer will occupy within a row. This will insert blank cells across the number of columns defined in this property. The default is one (1); the system will insert one blank cell.
Include in Tab Order	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"><li>• When set to &lt;Default&gt;, the element is included in tab order according to global rules and its focusable state.</li><li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li><li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li></ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p> <p>The system will call any script defined in this property (including onInit,</p>

<b>On Enter</b>	Optional	onClick, and toString). This method is invoked when the form element is focused on the Web page and the user presses the <b>Enter</b> key.
<b>Tooltip</b>	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers over the Cell Spacer. Default is empty, which means that a tooltip has not been defined.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the cell spacer. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Start Row</b>	Optional	This property works with the Cell Column Spacer property to define whether the blank cells will be started on a new row. A <b>True</b> value will insert blank cells in a new row and a <b>False</b> value will insert blank lines within the row that the Cell Spacer was inserted.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Cell Spacer such as background color. Default is NONE.

## Notes

- Cell Spacer must be parented by a Grid Layout.
- The cell spacer will not break up a label and its field within a given row. The label and its field element will move to the next row if the cell spacer infringes on one of its cell space.
- The following example displays the a Cell Spacer between field elements A and B with a Cell Column Span property equal to 1 (one). The first example shows the Start Row property = False and the second examples shows a Start Row property = True. Note that when a Start Row property is True that the system inserts a blank cell starting in a new row.

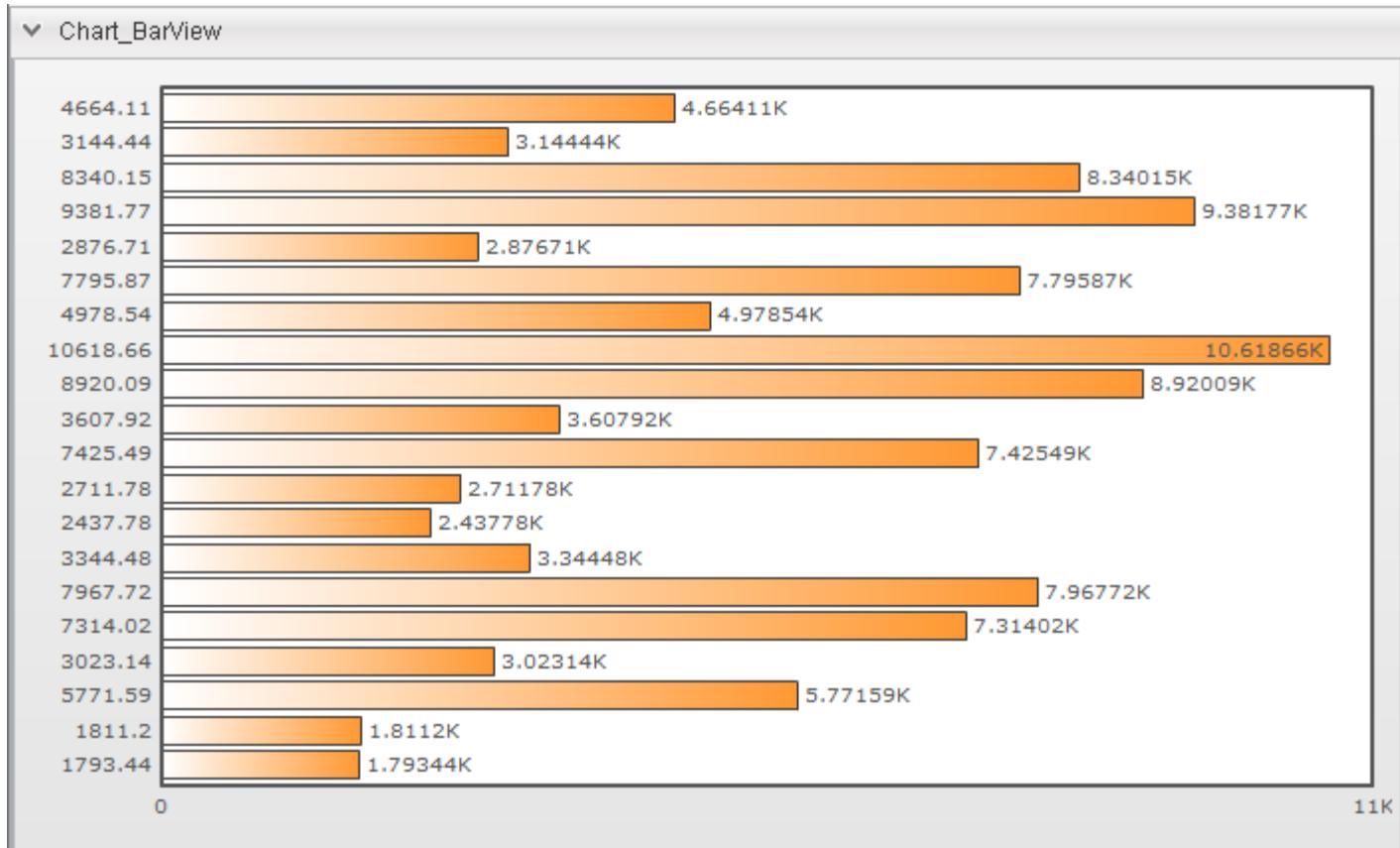


## Chart

Chart is a container that allows for the configuration of charts that represent data in the result Form of the User Interface (of type Finder User Interface). The Chart element enables you to configure the overall look and feel of the chart display at runtime. The values that display in the chart are controlled using the [ChartField](#) Form element.



The following is an example of a chart at runtime.



### Displaying Multiple Charts at Runtime

You can display multiple charts in a Form by creating multiple Form Frame Form elements and then referring the variable of each element to the Finder User Interface Chart Form that contains the chart that you want to display. Other Form elements can be added between the Form Frames to create a customized look.

1. In the Finder User Interface, right-click the node and select **New Form**. Alternatively, you can use an existing Form.
2. In the New Metadata Object dialog, type the name of the new form.
3. In the Form Element pane, right-click the node and select Add.
4. From the Add Element dialog, select **Misc Controls > Form Frame**.
5. Continue to add Form Frame elements.
6. Configure the Variable property of each Form Frame to refer to the Chart Form element.

### Summary

references Variable	Yes
---------------------	-----

references Method	No
references Form	No
Allowable Child Elements	ChartField

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	The chart name.
Type	Mandatory	<p>The type of chart. The default value is Bar. Select from one of the following chart types:</p> <ul style="list-style-type: none"> <li>• Area</li> <li>• Bar</li> <li>• Column</li> <li>• Line</li> <li>• Pie</li> <li>• Scroll area</li> <li>• Scroll column</li> <li>• Scroll line</li> <li>• Scroll stacked column</li> <li>• Stacked area</li> <li>• Stacked bar</li> <li>• Stacked column</li> </ul>
Variable	Mandatory	Finder result form for the chart. Use the property's <a href="#">lookup button</a> to view property details.
3D	Optional	Displays a three-dimensional (3-D) version of the chart. Note, some chart types do not render in 3D.
Caption	Optional	The chart label.
Click Method	Optional	<p>When you specify this property, chart elements are clickable. Clicking these elements invokes the method. The method receives two parameters:</p> <ul style="list-style-type: none"> <li>• The data object that the chart node represents</li> <li>• The variable name, if applicable, that the chart node represents</li> </ul>
Dynamic Height	Optional	<p>This property allows you to dynamically assign a variable or a script to the <b>Dynamic Height</b> property. For a variable, the data type must be either an integer or string. For the script, only the script action can be used and its return type must be defined as either integer or string.</p> <p>At runtime, by default, the regular width and height are used. However, if <b>Dynamic Width</b> and <b>Dynamic Height</b> are defined, they replace the regular width and height, and the values are calculated based on either the variable or script.</p>
Dynamic Width	Optional	<p>This property allows you to dynamically assign a variable or a script to the <b>Dynamic Width</b> property. For a variable, the data type must be either an integer or string. For the script, only the script action can be used and its return type must be defined as either integer or string.</p> <p>At runtime, by default, the regular width and height are used. However, if <b>Dynamic Width</b> and <b>Dynamic Height</b> are defined, they replace the regular width and height, and the values are calculated based on either the variable or script.</p>
Height	Optional	The average height of the chart.
Number of Div Lines	Optional	The number of horizontal lines in the graph.
Number Prefix	Optional	The prefix for the number shown in the chart.
Number Suffix	Optional	The suffix for the number shown in the chart.

<b>Show Validation</b>	Optional	This boolean property has values of either <i>True</i> or <i>False</i> , which determines whether a validation warning message displays at runtime. By default, this property is unchecked, meaning that validation errors do not appear under the chart at runtime and supports backward compatibility. Otherwise, selecting this property allows validation errors to appear under the chart at runtime.
<b>Show zeros</b>	Optional	If checked, show 0 value data.
<b>Sub Caption</b>	Optional	This field allows you to enter a sub-caption for your chart, which appears under the title.
<b>Variable</b>	Optional	Set this property to the variable that will hold an array of documents to look up values for the chart. The values to be used depend on the chart fields added under this chart element. The first field displays as the x-coordinate and second field displays as the y-coordinate. Any chart fields after the second field are only relevant to specific chart types. Use the property's <a href="#">lookup button</a> to view property details.
<b>Width</b>	Optional	The average width of the chart.
<b>X Axis</b>	Optional	X Axis label (horizontal).
<b>Y Axis</b>	Optional	Y Axis label (vertical).
<b>Y axis Max Value</b>	Optional	Specifies the maximum value to display on the y axis. If the value specified is less than the maximum value in the dataset, the chart will ignore the yAxisMaxValue. Proper use of this property is to set it equal to or greater than the maximum value that can be returned in the data set.
<b>Y axis Min Value</b>	Optional	Specifies the minimum value to display on the y axis. If the value specified is greater than the minimum value in the dataset, the chart will ignore the yAxisMinValue. Proper use of this property is to set it equal to or lesser than the minimum value that can be returned in the data set.

## Notes:

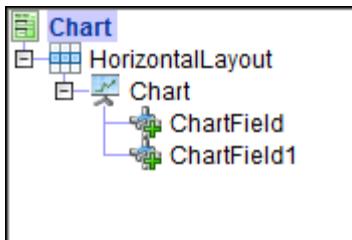
- The Chart Form element has a child element called ChartField.
- At runtime, right-click the chart for more viewing options.
- To view charts, Adobe Flash Player 8.0 or higher must be installed for the Firefox browser. Adobe Flash Player 8.0 or higher as an Active X object must be installed for Internet Explorer. Adobe Flash Player can be downloaded from Adobe's Web site at [www.adobe.com](http://www.adobe.com).
- To view JavaScript-based charts, FusionCharts automatically renders them on computers that do not have a Flash player installed. The only client-side requirement is a browser that supports JavaScript.
- To use multilingual characters in a chart, the XML must be UTF-8-encoded. More importantly, the XML file requires a Byte Order Mark (BOM) stamp to be present as the first three Bytes of the file.

## ChartField

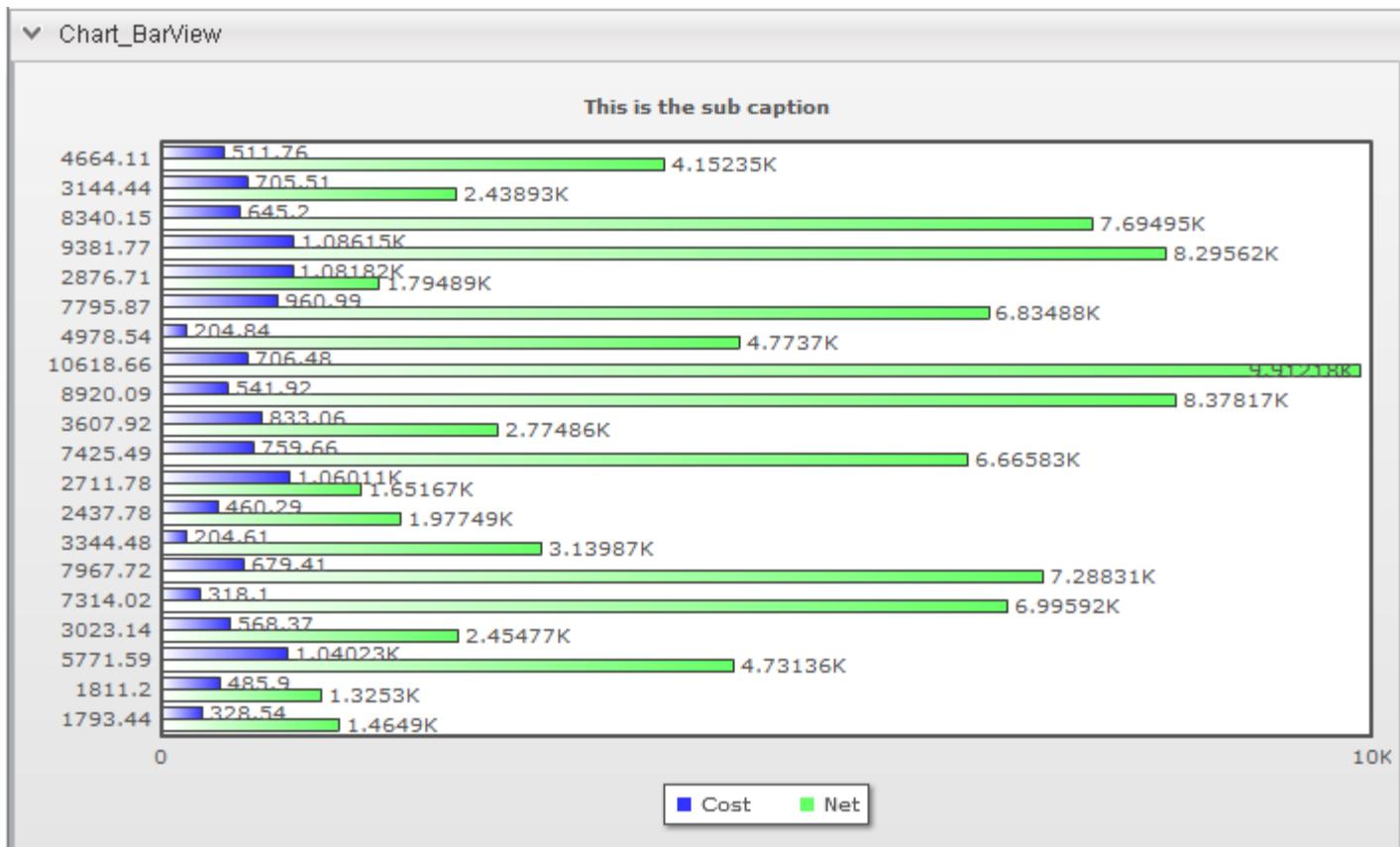
The ChartField provides additional properties for configuring the appearance of the Chart Form element at runtime. Using the properties you can configure how the chart renders in the Web browser at runtime.

The chart data is configured to display data from the Result Form. Each column of data in the Result Form is displayed as a part of the entire chart. The ChartField Form element properties enable you to configure the data that is displayed in the chart. You can have multiple ChartFields elements under the Chart Form element (see list in the note section) each displaying different data for that Finder.

**Note:** You can only add a chartfield to a User Interface of type Finder User Interface.



Each ChartField element displays a column of data in the Finder results.



## Summary

references Variable	Yes
references Method	No
references Form	No
Allowable Child Elements	No

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	The ChartField name.
Label	Mandatory	The ChartField label.
Color	Optional	The chart background color (HTML #rrggb format).
Height	Optional	The average height of the chart.
On Enter	Optional	The system will call any script defined in this property (including onInit, onClick, and toString). This method is invoked when the form element is focused on the web page and the user presses the ENTER key.
Style	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Grid Layout such as positioning and colour. Default is NONE.
Variable	Optional	Finder result form for the chart. Only Forms of the Output Document with more than one field are listed here. The result form fields are specified as follows: category name, series 1 data, series 2 data, etc. The category label is used as the default <b>Domain axis label</b> , range axis labels are used as the legend labels. Use the property's <a href="#">lookup button</a> to view property details.
Visible	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
Width	Optional	The average width of the chart.

## Notes

You cannot use multiple ChartField form elements with the following charts: Bubble, Scatter, Scroll Line, Scroll Area, Scroll Column, Scroll Stacked Column, Stacked Area, Stacked Bar, Stacked Column.

## Checkbox

The Checkbox element creates a checkbox. The value of the Checkbox can be assigned to a variable and modified dynamically at runtime by the user or by a user action. Similarly the label of the Checkbox can be modified dynamically. When it refers to a Finder table, the values can be sorted at runtime into True and False groupings via the Group By In Table property.



### Summary

references Variable	Yes
references Method	Yes, User Action Methods, Scripts and Permissions.
references Form	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Checkbox.
Label	Mandatory	Visible label of the Checkbox. Default is empty which results in no label appearing.
Allow Empty Value	Optional	Checking this property provides three options (NULL, TRUE, FALSE) when the checkbox is clicked in runtime. By default, a checkbox is unchecked. The NULL value is displayed as a normal checkbox but with a solid square in it and it is saved in the database as a NULL value.
Can Filter	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to filter information in a table at runtime.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the checkbox in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Checkbox shall occupy in the Grid Layout, to the right of current cell. Default is empty, which occupy one column.
Cell Row Span	Optional	<p>This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
Dynamic Label	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
		The value of this property is a method with a Return type <i>String</i> . In conjunction with the

<b>Dynamic Label Style</b>	Optional	<b>Variable</b> property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
<b>Dynamic Tooltip</b>	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
<b>Dynamic Tooltip Style</b>	Optional	When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.
<b>Editable</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Error Icon Gap</b>	Optional	This property determines whether the checkbox is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user this field will be enabled or disabled).  <b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.
<b>Error Icon Orientation</b>	Optional	Select this property's checkbox to make a gap for the validation icon. By default, this property is false.  <b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.
<b>Group By In Table</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Height</b>	Optional	Height to be taken up by the Checkbox area; can be an integer which is in pixels, or in percentage (for example, "50%") of page height or parent element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>Icon</b>	Optional	Choose an image file that appears as an icon to the right of the Checkbox. Default is NONE.
<b>Include in Tab Order</b>	Optional	This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values: <ul style="list-style-type: none"><li>• When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li><li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li><li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li></ul> <b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.
		This property's checkbox is selected by default, which indicates that the label is set as the checkbox field's title. For right label orientation, the order appears as follows: <checkbox> <icon> <label>

<b>Label as Title</b>	Optional	<p>When you uncheck this property, the order is as follows:  &lt;checkbox&gt; &lt;label&gt; &lt;icon&gt;</p> <p>By default, the checkbox with icon only takes the label column, as the label column may be too big.</p> <p>To take both columns (that is, label and field), set <b>Label Orientation</b> for the checkbox element to <b>right</b> and set <b>Cell Column Span</b> to <b>2</b>.</p> <p>If you require displaying the checkbox with icon only at the field column, set <b>Label Orientation</b> for the checkbox element) to <b>left</b>, which leaves a gap for the label column.</p> <p><b>Note:</b> By default, a checkbox element's label is rendered as a separate label element, which may affect the overall alignment of a grid layout where the label forces the column to take up more room than it requires. To fix the alignment for certain layouts that have checkboxes with label alignment as right/left, deselect the <b>Label As Title</b> property.</p>
<b>Label Orientation</b>	Optional	<p>Specifies where the label of the Checkbox appears in relation to the checkbox area; can be <i>left</i>, <i>right</i> or <i>top</i> of the checkbox area. The default orientation is <i>right</i>. However, in a Grid Layout , the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout.</p>
<b>Label Style</b>	Optional	<p>Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the Label such as font color. Default is NONE.</p>
<b>Remove Label Line</b>	Optional	<p>Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.</p>
<b>Run Trigger</b>	Optional	<p>The method assigned to this property is called whenever the value of the Checkbox changes (user clicks or uncicks the Checkbox). The method can be a User Action Method or a Script.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>• You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>• You click a menu.</li> <li>• You change your selection in a navigation tree.</li> <li>• You click another tab of a tabset.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• Field triggers do not fire on save. However, document triggers do.</li> <li>• As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
<b>Show Label</b>	Optional	<p>This property determines whether the label is displayed on the Checkbox. Choose from <i>True</i> or <i>False</i>. Default is True which means that the label is displayed.</p>
<b>Start Row</b>	Optional	<p>Applicable only when the parent element is a Grid Layout. If true, the Checkbox appears in a new row of the grid. If <i>False</i>, it is simply placed in the next available cell. Default is <i>False</i>, which means that the Checkbox element does not appear on a new row.</p>
<b>Style</b>	Optional	<p>Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the button such as positioning and color. Default is none.</p>
<b>Toggle Edit</b>	Optional	<p>This property works in conjunction with the Editable property of the Table element to allow the user to be able to modified the value of this element on first click. The Editable property of the Table element must be set to <i>True</i>. Setting this property to <i>False</i> will require that the user click the Checkbox field twice to change the value: 1) the first click enters in edit mode and 2) the second click changes the value. Setting this property to <i>True</i>, allows the user to click the checkbox once to change the value. For information about setting configuring checkbox in an editable table, see Edit in Table in the <a href="#">Finder Result Form</a>.</p>

<b>Tooltip</b>	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers over the Button. Default is empty, which means that a tooltip has not been defined.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the <b>Checkbox</b> field. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Variable</b>	Optional	This property can be a <i>String</i> , <i>Integer</i> or <i>Boolean</i> data type (for example, String values equal to <i>True</i> , <i>False</i> , 1 (one) or 0 (zero), Integer values equal to 1 (one) or 0 (zero), and Boolean values equal to <i>True</i> or <i>False</i> ). Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
<b>Width</b>	Optional	Width to be taken up by the Button; can be in integer which is in pixels, or in percentage (for example, "50%") of page width or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Notes

If Checkbox is parented by a Grid Layout, the label orientation of the Checkbox follows Grid Layout's **Label Orientation** property, unless the Checkbox's Label Orientation property is specified.

If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

Use the "Run Trigger" property to add a Method that will be triggered when the checkbox is toggled. For example, you can create validation rules for the Checkbox this way.

The Group By In Table property allows the values of an Element to be grouped by the value of the result of the Element in a Finder. If this value is set to True, then at runtime the user will have the ability to sort the results by the value of the field. The example below shows the Checkbox results grouped by True and False results when the user selects the "Group by" field menu option.

Date
12/14/2007
7/8/1979
5/5/2011
11/28/1982
9/24/2014
4/19/1986
11/27/2003
6/22/1975
4/19/2007
11/11/1978
9/8/2010

\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent

layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Date Field

The Date Field element is a text field with a calendar widget icon. The Date Field can be enable/disabled and displayed/hidden. The Date Field can be used in conjunction with the Finder object and Table elements to allow the search results data of the Date Field to be sorted and/or grouped. The format of the field is controlled by the Date Field property to allow various date formats. Below is an example of the Date Field sorted in ascending order and grouped by month.

The screenshot shows a 'Date Time - Date Field' list within an 'Edit In Table' window. The list is grouped by month, with 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', and 'Dec' each having a plus sign next to it. Under 'Jan', there are three entries: '2/14/2018', '2/14/1997', and '2/16/1976'. At the bottom of the list is a minus sign followed by '-none-'.

### Summary

references Variable	Yes
references Method	Yes
references Form	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Date Field.
Label	Mandatory	Visible label of the Date Field. Default is empty which results in no label appearing.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Date Field in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Date Field occupies in the Grid Layout, to the right of current cell. Default is empty, which occupies one column.

<b>Cell Row Span</b>	Optional	<p>This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
<b>Date Icon</b>	Optional	Specify a specific Date Icon to use. Default is empty, which renders a default calendar icon.
<b>Display Format</b>	Optional	Specify the format displayed for the Date Field element. A listing of supported formats is outlined in the Notes section below. Default is empty field but the system will use the protected static string "DF_DEFAULT_DATE = "MM/dd/yyyy"" as the default.
<b>Dynamic Label</b>	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
<b>Dynamic Label Style</b>	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
<b>Dynamic Tooltip</b>	Optional	<p>Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i>. The static Tooltip property is considered as default until the action returns a value.</p> <p>When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.</p>
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Editable</b>	Optional	This property determines whether the Date Field is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be enabled or disabled).
<b>Error Icon Gap</b>	Optional	<p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.</p> <p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Group By In Table</b>	Optional	This property allows the user to group the Finder results by the elements value. In the case of a Date Field, the results can be grouped by Day, Week, Month, Quarter, Year, Day of Month and Upcoming.
<b>Icon</b>	Optional	Choose an image file that appears as an icon to the right of the Date Field. Default is

		NONE.
<b>Include Icon in Tab Order</b>	Optional	The property allows you to optionally exclude the icon button from the tab order.
<b>Include in Tab Order</b>	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>Label Align</b>	Optional	Click the <b>Value</b> field and select from one of the following alignment options for your label: <ul style="list-style-type: none"> <li><b>Left</b></li> <li><b>Centre</b></li> <li><b>Right</b></li> </ul>
<b>Label Orientation</b>	Optional	Specifies where the label of the Date Field appears in relation to the Date Field area; can be <i>left</i> , <i>right</i> or <i>top</i> of the Date Field area. The default orientation is <i>right</i> . However, in a Grid Layout, the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout.
<b>Label Style</b>	Optional	Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the Label such as font color. Default is NONE.
<b>On Enter</b>	Optional	The system calls any script defined in this property. This method is invoked when the Date Field is focused on the Web page and the user presses the ENTER key.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Run Trigger</b>	Optional	<p>The method assigned to this property is invoked whenever the value of the Date Field changes. The method can be a User Action Method or a Script.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>You click a menu.</li> <li>You change your selection in a navigation tree.</li> <li>You click another tab of a tabset.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>Field triggers do not fire on save. However, document triggers do.</li> <li>As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
<b>Show Data in Hover</b>	Optional	<p>This property must be set for the column's data to appear as part of the detail viewer. The viewer shows a maximum of five data columns.</p> <p>If no columns are selected and the table element's <b>Show Hover Detail</b> property is selected, all data columns marked as <b>Visible = false</b> are shown in the detail viewer.</p>
<b>Show Data Summary</b>	Optional	The element must exist under a Table element. The Table element must have the <b>Show Data Summary</b> checkbox selected.

<b>Show Group Summary</b>	Optional	The element must exist under a Table element. At least one table field element must exist with the <b>Group by In Table</b> checkbox selected. Additionally, the table element must have the <b>Show Group Summary</b> checkbox selected.
<b>Show Label</b>	Optional	This property determines whether the label is displayed on the Date Field. Choose from <i>True</i> or <i>False</i> . Default is <i>True</i> which means that the label is displayed.
<b>Show Picker Icon</b>	Optional	This property determines whether the Date Icon is displayed on the Date Field. Choose from <i>True</i> or <i>False</i> . Default is <i>True</i> which means that the Date Icon is displayed.
<b>Start Row</b>	Optional	Applicable only when the parent element is a Grid Layout. If true, the Date Field appears in a new row of the grid. If <i>False</i> , it is simply placed in the next available cell. Default is <i>False</i> , which means that the Date Field element does not appear on a new row.
<b>Start Year</b>	Optional	This property sets the minimum year that the user can select from the calendar. If you click the drop-down Year on the calendar, you can see that the range of available years starting at the Start Year. The default value is 1995, and the minimum year specified has to be 1901 or greater.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the button such as positioning and color. Default is none.
<b>Summary Label</b>	Optional	The element must exist under a table element. The <b>Summary Type</b> property must be set to <b>count</b> .
<b>Summary Type</b>	Optional	The element must exist under a Table element. The <b>Show Data Summary</b> checkbox must be checked. See the <a href="#">Summary Types</a> section for a list of available summary types.
<b>Text Align</b>	Optional	Specifies the text alignment of the Date Field entry at runtime; it can be <i>left</i> , <i>center</i> , or <i>right</i> . Default is empty, which is right-aligned.
<b>Textbox Style</b>	Optional	<p>Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the textbox of the Date Field such as color and font. Default is <b>NONE</b>.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the <b>MyStyle</b> style to the <b>Textbox Style</b> property, the stylesheet must have <b>MyStyleDisabled</b> to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the <b>Date</b> field. Default is empty, which means no tooltip for the <b>Date</b> field.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the <b>Date</b> field. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Use TextField</b>	Optional	Boolean, when true, renders text field to present the date, or when false, renders three drop-down listbox to present month, day of month and year respectively. Default is true.
<b>Variable</b>	Optional	Choose from the list of elementary Variables of the types <i>Date</i> , <i>DateTime</i> , <i>DateMin</i> , and <i>DateSec</i> available in the User Interface to bind to this Date Field. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Determines whether the Date Field is visible or not. Choose true (visible) or false (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this field. Pre-defined system <a href="#">Permissions</a> such as <code>addPerm</code> , <code>deletePerm</code> , <code>editablePerm</code> , <code>attachPerm</code> , <code>executePerm</code> , and <code>visiblePerm</code> can be assigned as well as Object Permissions. Defaults to <b>NONE</b> or <b>True</b> , which means no permissions assigned and field is visible..
<b>Width</b>	Optional	Width to be taken up by the Date Field; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element. If this field is empty, the Width of the field should be taken up by that defined in the Column Width property of the Grid Layout. If this element is nested under a Table element, the width and height

properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

\* If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Supported Formats

Below is a listing of supported display formats for date and date time:

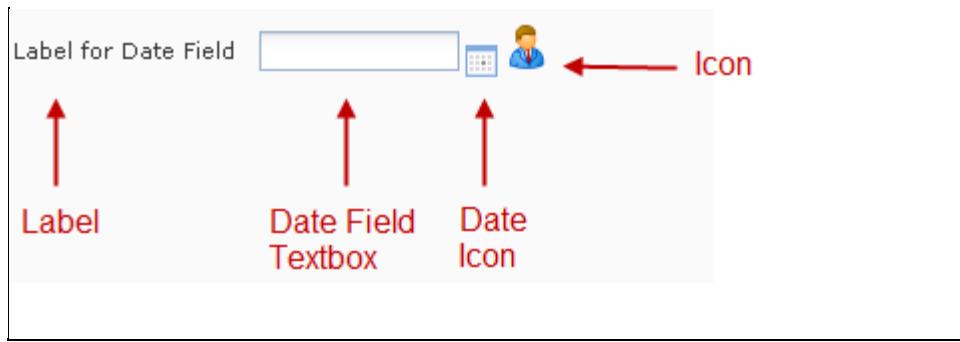
Letter	Date or Time Component	Presentation	Examples	Limitation
y	Year	<a href="#">Year</a>	1996; 96	
M	Month of the year	<a href="#">Month</a>	July; Jul; 07	Only numeric value is supported in editable form.
d	Day in month	<a href="#">number</a>	10	
E	Day in week	<a href="#">text</a>	Tuesday; Tue	Only numeric value is supported in editable form.
a	am/pm marker	<a href="#">text</a>	PM	
H	Hour in day (0-23)	<a href="#">number</a>	0	
h	Hour in am/pm (1-12)	<a href="#">number</a>	12	When using the <b>hh</b> format, use the <b>a</b> format to specify am/pm (for example, <b>dd/MM/yyyy hh:mm a</b> )
m	Minute in hour	<a href="#">number</a>	30	
s	Second in minute	<a href="#">number</a>	55	
S	Millisecond	<a href="#">number</a>	978	

Some examples of the display date and date time formats are:

Display Format Value	Description
MM/dd/yyyy	month/day/year
MM/dd/yyyy HH:mm	month/day/year hours:minutes
dd/MM/yyyy	day/month/year
dd/MM/yyyy HH:mm	day/month/year hours:minutes
yyyy/MM/dd	year/month/day
yyyy/MM/dd HH:mm	year/month/day hours:minutes

## Notes

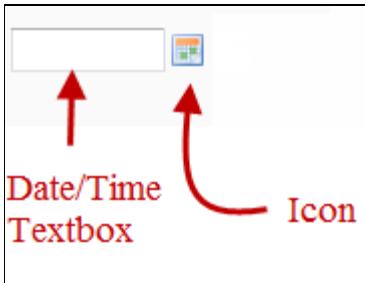
- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- The Date Field has a Calendar icon to the right that allows the user to select a date using a Calendar pop-up widget. The alignment of the Date Field properties are displayed in the graphic below.



- If Date Field is parented by a Grid Layout, the label orientation of the Date Field follows Grid Layout's **Label Orientation** property.
- Non-numeric formats are supported as read-only fields. Formats containing the MMM or E are only supported as read-only fields. These formats require internationalization that can only be achieved with server-side code. There is an automatic rendering of non-editable text fields in place of date fields using the formats specified. When these formats are used, the value is automatically translated to the browser language set provided the metadata has specified this language as being supported.

## Date Time Field

The Date Time Field can be enabled or disabled, and displayed or hidden. The Date Time Field can be used in conjunction with the Finder object and Table elements to allow the search results data of the Date Field to be sorted or grouped, or both. The format of the field is controlled by the Display Format property to allow various date formats. The following is an example of the Date Time Field.



**Note:** Selecting a time from the picker is not supported. You can only input the time by directly typing it in the text field.

### Summary

references Variable?	Yes
references Method?	Yes
references Form?	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Date Field.
Label	Optional	Visible label of the Date Field. Default is empty which results in no label appearing.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Date Field in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Date Field occupies in the Grid Layout, to the right of current cell. Default is empty, which occupies one column.
Cell Row Span	Optional	This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.  By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.  To use this property, see the <a href="#">example</a> for details.
Date Icon	Optional	Specify a specific Date Icon to use. Default is empty, which renders a default calendar icon.

<b>Display Format</b>	Optional	Specify the format displayed for the Date Field element. A listing of supported formats is outlined in the Notes section below. If seconds is indicated in this property then the Seconds textbox will be displayed (For example, MM/dd/yyyy HH:mm:SS) otherwise, the Seconds textbox is hidden (MM/dd/yyyy HH:mm). Default is empty field but the system will use the protected static string "DF_DEFAULT_DATETIME = "MM/dd/yyyy HH:mm:ss"" as the default. Note that the S (uppercase) is the fractional second number, while the s (lowercase) is the second in minute number.
<b>Dynamic Label</b>	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
<b>Dynamic Label Style</b>	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
<b>Dynamic Tooltip</b>	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
<b>Dynamic Tooltip Style</b>	Optional	When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.
<b>Editable</b>	Optional	This property determines whether the Date Field is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be enabled or disabled).
<b>Error Icon Gap</b>	Optional	<b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.
<b>Error Icon Orientation</b>	Optional	Select this property's checkbox to make a gap for the validation icon. By default, this property is false.  <b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.
<b>Group By In Table</b>	Optional	This property allows the user to group the Finder results by the elements value. In the case of a Date Field, the results can be grouped by Day, Week, Month, Quarter, Year, Day of Month and Upcoming.
<b>Icon</b>	Optional	Choose an image file that appears as an icon to the right of the Date Field. Default is NONE.
<b>Include Icon in Tab Order</b>	Optional	The property allows you to optionally exclude the icon button from the tab order.
		This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <u>suppress tabbing</u> configuration setting and can take the following values: <ul style="list-style-type: none"> <li>• When set to &lt;Default&gt;, the element is included in tab order according to global rules and its focusable state.</li> </ul>

<b>Include in Tab Order</b>	Optional	<ul style="list-style-type: none"> <li>If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>Label Align</b>	Optional	<p>Click the <b>Value</b> field and select from one of the following alignment options for your label:</p> <ul style="list-style-type: none"> <li><b>Left</b></li> <li><b>Centre</b></li> <li><b>Right</b></li> </ul>
<b>Label Orientation</b>	Optional	Specifies where the label of the Date Field appears in relation to the Date Field area; can be <i>left</i> , <i>right</i> or <i>top</i> of the Date Field area. The default orientation is <i>right</i> . However, in a Grid Layout, the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout.
<b>Label Style</b>	Optional	Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the Label such as font color. Default is NONE.
<b>On Enter</b>	Optional	The system calls any script defined in this property. This method is invoked when the Date Field is focused on the Web page and the user presses the ENTER key.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Run Trigger</b>	Optional	<p>The method assigned to this property is invoked whenever the value of the Date Field changes. The method can be a User Action Method or a Script.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>You click a menu.</li> <li>You change your selection in a navigation tree.</li> <li>You click another tab of a tabset.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>Field triggers do not fire on save. However, document triggers do.</li> <li>As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
<b>Show Label</b>	Optional	This property determines whether the label is displayed on the Date Field. Choose from <i>True</i> or <i>False</i> . Default is True which means that the label is displayed.
<b>Start Row</b>	Optional	Applicable only when the parent element is a Grid Layout. If true, the Date Field appears in a new row of the grid. If <i>False</i> , it is simply placed in the next available cell. Default is <i>False</i> , which means that the Date Field element does not appear on a new row.
<b>Start Year</b>	Optional	This property sets the minimum year that the user can select from the calendar. If you click the drop-down Year on the calendar you can see that the range of available years starting at the Start Year. The default value is 1995, and the minimum year specified has to be 1901 or greater.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the button such as positioning and color. Refer to the information on <a href="#">Style Sheets</a> for details on configuration. Default is none.
<b>Text Align</b>	Optional	Specifies the text alignment of the Date Field entry at runtime; it can be <i>left</i> , <i>center</i> , or <i>right</i> . Default is empty, which is right-aligned.
		Choose from styles in the selected stylesheet at the toolbar, which defines the styling of

		the textbox of the Date Field such as color and font. Default is NONE.
<b>Textbox Style</b>	Optional	To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the MyStyle style to the <b>Textbox Style</b> property, the stylesheet must have MyStyleDisabled to represent the disabled style.  To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet. <b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Date Field. Default is empty, which means no tooltip for the Date Field.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the <b>Date Time</b> field. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Use TextField</b>	Optional	Boolean, when true, renders text field to present the date, or when false, renders three dropdown listbox to present month, day of month and year respectively. Default is true.
<b>Variable</b>	Optional	Choose from the list of elementary Variables of the types <i>DateTime</i> , <i>DateMin</i> and <i>DateSec</i> available in the User Interface to bind to this Date Field. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Determines whether the Date Field is visible or not. Choose true (visible) or false (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this field. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE or True, which means no permissions assigned and field is visible..
<b>Width</b>	Optional	Width to be taken up by the Date Field; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element. If this field is empty, the Width of the field should be taken up by that defined in the Column Width property of the Grid Layout. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Supported Formats

Below is a listing of supported display formats for date and date time:

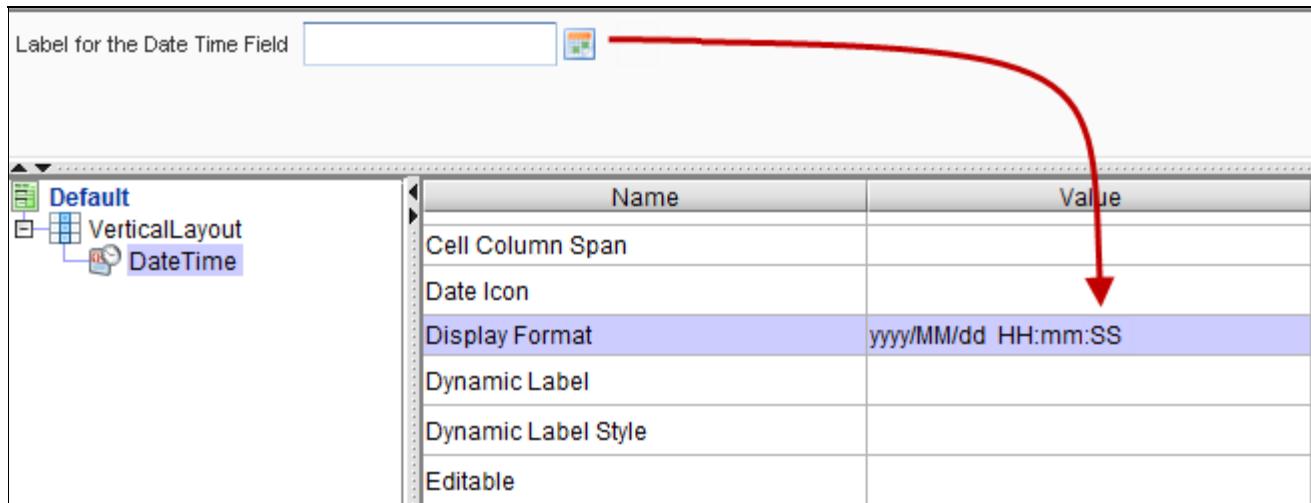
Letter	Date or Time Component	Presentation	Examples	Limitation
y	Year	<a href="#">Year</a>	1996; 96	
M	Month of the year	<a href="#">Month</a>	July; Jul; 07	Only numeric value is supported in editable form.
d	Day in month	<a href="#">number</a>	10	
E	Day in week	<a href="#">text</a>	Tuesday; Tue	Only numeric value is supported in editable form.
a	am/pm marker	<a href="#">text</a>	PM	
H	Hour in day (0-23)	<a href="#">number</a>	0	
h	Hour in am/pm (1-12)	<a href="#">number</a>	12	When using the <b>hh</b> format, use the <b>a</b> format to specify am/pm (for example, <b>dd/MM/yyyy hh:mm a</b> )
m	Minute in hour	<a href="#">number</a>	30	
s	Second in minute	<a href="#">number</a>	55	
S	Millisecond	<a href="#">number</a>	978	

Some examples of the display date and date time formats are:

Display Format Value	Description
MM/dd/yyyy	month/day/year
MM/dd/yyyy HH:mm	month/day/year hours:minutes
MM/dd/yyyy HH:mm:ss	month/day/year hours:minutes:seconds The seconds will be displayed in the seconds textbox.
dd/MM/yyyy	day/month/year
dd/MM/yyyy HH:mm	day/month/year hours:minutes
dd/MM/yyyy HH:mm:ss	day/month/year hours:minutes:seconds The seconds will be displayed in the seconds textbox.
yyyy/MM/dd	year/month/day
yyyy/MM/dd HH:mm	year/month/day hours:minutes
yyyy/MM/dd HH:mm:ss	year/month/day hours:minutes:seconds The seconds will be displayed in the seconds textbox.

## Notes

- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- The Date Time field has a Calendar icon to the right that allows the user to select a date using a Calendar pop-up widget. Specify the Date Time field's format using the **Display Format** property.



- This Property is used for recording Date/Times. However, setting the current time is not supported.
- Supported delimiter characters is not limited to "/". Delimiters can be a range of characters from whitespace to commas.
- Validation: The values entered in the DateTime field are validated on the browser side when leaving the field. Any validation errors that exist are displayed beside the DateTime element. Any actions requiring updates to the server (such as save) are not allowed when a datetime value is invalid. For example, a Date format of MM/dd/yy will cause a validation error when a value of 1/1/2001 is entered.
- Non-numeric formats are supported as read-only fields. Formats containing the MMM or E are only supported as read-only fields. These formats require internationalization that can only be achieved with server-side code. There is an automatic rendering of non-editable text fields in place of date fields using the formats specified. When these formats are used, the value is automatically translated to the browser language set provided the metadata has specified this language as being supported.
- The format currently used is the GWT's DateFormat to format date time. These formats are for read-only purposes for table element date-time columns and cannot be used with the date picker icon.

<http://google-web-toolkit.googlecode.com/svn/javadoc/1.6/com/google/gwt/i18n/client/DateTimeFormat.html>

## Dialog

The Dialog element displays a dialog box that can be configured to display a warning message and user action buttons. The Dialog element is used in the Default form of the Confirmation Dialog User Interface.

The following user action buttons can be configured to display:

- Yes
- No
- OK
- Cancel

Using the Variable property, you can configure the dialog to display different types of confirmation dialogs.



### Summary

references Variable	Yes
references Method	Yes
references Form	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Dialog.
Variable	Mandatory	This property can either be <i>String</i> , <i>Integer</i> or <i>Boolean</i> (for example, String values equal to <i>True</i> , <i>False</i> , 1 (one) or 0 (zero), Integer values equal to 1 (one) or 0 (zero), and Boolean values equal to <i>True</i> or <i>False</i> . Use the property's <a href="#">lookup button</a> to view property details.
Auto Expire(s)	Optional	Setting this property indicates that the dialog automatically closes after a certain period in seconds, without an user action to close it. The default value is 0 seconds.
Can Resize	Optional	The <b>Can Resize</b> property is automatically selected by default, indicating that dialog resizing is enabled. To disable dialog resizing, deselect the checkbox.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
Cell Alignment	Optional	Controls the horizontal and vertical positioning of the dialog in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the dialog occupies in the Grid Layout, to the right of current cell. Default is empty, which occupies one column.
Dynamic Height	Optional	This property allows the label on the dialog to dynamically change at runtime. The value of this property is a variable of type <i>String</i> or a user action method that returns a <i>String</i> .
Dynamic Tooltip	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.  When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.
Dynamic Tooltip	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.

Style		
Dynamic X Coordinate	Optional	You can enable repositioning of your dialog by setting both the <b>Dynamic X Coordinate</b> and <b>Dynamic Y Coordinate</b> properties. Click the drop-down menu and select from the list (for example, <code>dialogX</code> ).
Dynamic Y Coordinate	Optional	You can enable repositioning of your dialog by setting both the <b>Dynamic X Coordinate</b> and <b>Dynamic Y Coordinate</b> properties. Click the drop-down menu and select from the list (for example, <code>dialogY</code> ).
Dynamic Width	Optional	The value of this property is a method with a Return type <i>String</i> . This method dynamically changes the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current CSS file that is being used by the application.
Editable	Optional	This property determines whether the checkbox is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user this field is enabled or disabled).  <b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's <code>editablePerm</code> permission takes over. If this method is not overwritten, the model's <code>editablePerm</code> permission is used. User interfaces displayed under another user interface with <code>editablePerm</code> are also be inherited if the embedded user interfaces do not have their own <code>editablePerm</code> permission.
Height	Optional	Height to be taken up by the dialog area; can be an integer which is in pixels, or in percentage (that is, "50%") of page height or parent element.
Label Orientation	Optional	Specifies where the label of the Dialog appears in relation to the checkbox area; can be <i>left</i> , <i>right</i> or <i>top</i> of the checkbox area. The default orientation is <i>right</i> . However, in a Grid Layout , the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout.
Label Style	Optional	Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the Label such as font color. Default is NONE.
Label Variable	Optional	Choose from a list of available <i>string</i> variables. This variable can be used to dynamically to set, change and use the title that is displayed on the tab bar. Similar to the Image Variable it can be initialized or dynamically changed by any method within the local user interface. This property is dependent on the Form property and is only available if the Form property value is given a value.
Modal	Optional	If set to <i>True</i> , when displayed, this Window intercepts and blocks events to all other existing components on the page.
On Enter	Optional	The system calls any script defined in this property. This method is invoked when the Grid Layout is focused on the Web page and the user presses the ENTER key.
Show Close Button	Optional	To remove the close button, deselect this property.
Show Header	Optional	This property determines whether the header is displayed on the Dialog. Choose from <i>True</i> or <i>False</i> . Default is <i>True</i> which means that the label is displayed
Show Label	Optional	This property determines whether the label is displayed on the Checkbox. Choose from <i>True</i> or <i>False</i> . Default is <i>True</i> which means that the label is displayed.
Show Maximize Button	Optional	To remove the maximize button, deselect this property.
Show Minimize Button	Optional	To remove the minimize button, deselect the <b>Show Minimize Button</b> checkbox.
Show Modal Mask	Optional	If set to <i>True</i> , the page behind the dialog is unavailable.
Start Row	Optional	Applicable only when the parent element is a Grid Layout. If true, the dialog appears in a new row of the grid. If <i>False</i> , it is simply placed in the next available cell. Default is <i>False</i> , which means that the Dialog element does not appear on a new row.
Style	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the button such as positioning and color. Default is none.
Tooltip	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers over the Dialog. Default is empty, which means that a tooltip has not been defined.
Tooltip Style	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the dialog. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
Tooltip Width (px)	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
		Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as <code>addPerm</code> ,

<b>Visible</b>	Optional	deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
<b>Width</b>	Optional	Width to be taken up by the Dialog; can be in integer which is in pixels, or in percentage (that is, "50%") of page width or parent Element.

## Notes:

- For the messageDefault Form of the Confirmation Dialog, the properties of the Dialog element are editable when you perform a *copy and replace* action.
- A Dialog box is restricted to the scope of its parent window. The parent window can only have one dialog, but the dialog can have its own dialog and so forth. This ensures a clean UI, restricting multiple browser windows open at once. It is also encouraged to create more complex user interfaces without the use of popups, which can be achieved by using a choice of available dynamic elements.

## Control Dialog Window

You can control the built-in dialog window functionality, which includes resizing and moving its location:

1. In Velocity Studio, create a new User Interface. For the **Extends** field, select **com.conceptwave.system.Dialog**.
2. Right-click the **Default** form and select **Override**.
3. Right-click the **dialog** element and select the **Copy & Replace...** option.
4. Navigate to the dialog property panel. You can do the following:
  - a. To remove the minimize button, deselect the **Show Minimize Button** checkbox.
  - b. The **Can Resize** property is automatically selected by default, indicating that dialog resizing is enabled. To disable dialog resizing, deselect the checkbox.
  - c. To enable repositioning of dialog, set both the **Dynamic X Coordinate** and **Dynamic Y Coordinate** properties by selecting **dialogX** and **dialogY** from the drop-down menu, respectively.
5. In a user interface, create a new method and select the **Show in popup** checkbox, which displays size properties. To move the popup's location, specify an x- and y-coordinate for the **Dialog X Coordinate** and **Dialog Y Coordinate** fields, respectively.

New Metadata Object

### New User Action

Name: testUserActionMethod  Private  Final  Instance  Type

Description:

Action Category:

Script:

```
function () {  
}  
}
```

Object: <NONE>

Form: <NONE>

Validate  Allow invalid object  Autosave  Show in popup

Dialog Width:  Dialog Height:

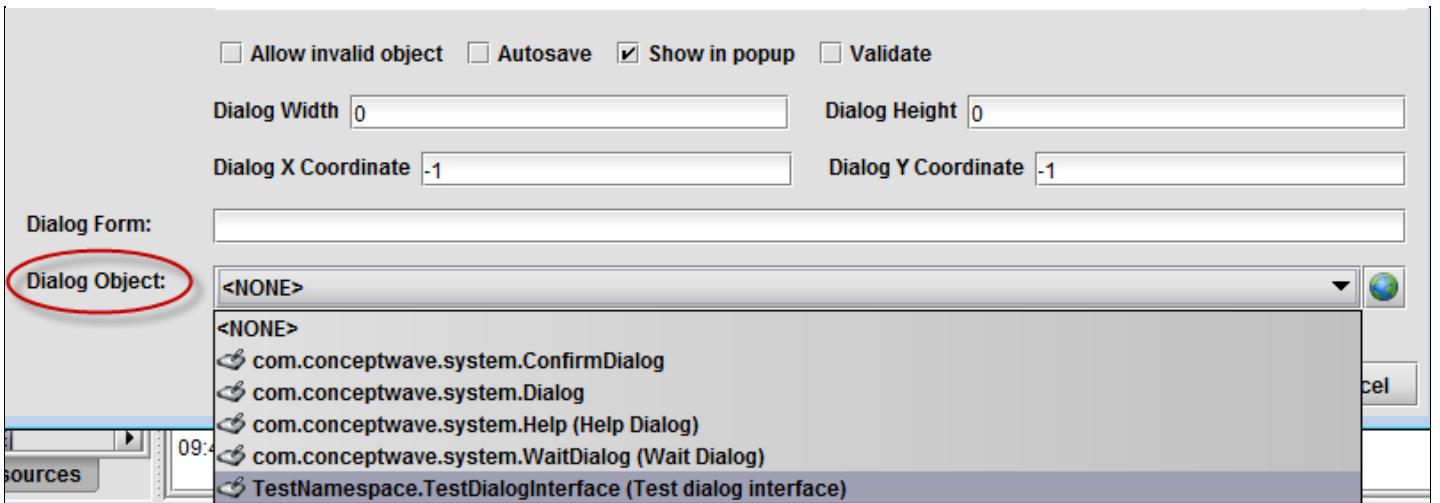
Dialog X Coordinate:  Dialog Y Coordinate:

Dialog Form:

Dialog Object: <NONE>

**Note:** By default, popups are centered.

6. To gain functionality of the dialog that you created in step 1, select it as a **Dialog Object** of the user action, which will tell the product to use that dialog to display your object.



7. To test your dialog, write a script in the method created in step 5 that returns the event log finder. Set the user action as the click method of a menu. In runtime, you can see that the properties you have set have taken effect.

#### Allow a Reference Finder to Pop Up from a Popup Dialog

A menu item pops up as a confirmation dialog, which has a reference finder popup as one of the input fields. If the dialog is presented as the top window, the reference finder pops up fine. If the the dialog is a popup, to also allow the reference to pop up, complete these steps:

1. Create a new User Interface that both extends and overrides com.conceptwave.system.ConfirmDialog.
2. Set the **Variable** property to **dialog**.
3. Set the **Unmanaged** property to **true**.
4. Save and run your metadata, which allows the finder to appear.

## Confirmation Dialog

The Confirmation Dialog is used to execute a user action and display a message in a dialog for the end user to acknowledge or dismiss. The User Interface contains several forms (Default, Help, Menu, and messageDialog) that when configured display a combination of confirmation dialog, message and user action buttons including, **Yes**, **No**, **OK**, and **Cancel**.

There are a number of variables and methods available to assist you in implementing the Confirmation Dialog configuration. You can add a Confirmation Dialog by adding a User Action Method. From this method you can configure the Confirmation field by selecting a script method that appears in the drop-down list. The Confirmation Dialog is triggered by a user action; when the script is run, it displays the contents of the confirmation dialog. You can setup an action in the User Interfaces objects, such as a Document, Finder or Order. For example, the user action is represented in the interface by a button action on a Document which in turn displays the Confirmation Dialog.

You can create a Confirmation Dialog by extending a user Interface using the `com.conceptwave.system.ConfirmDialog` object.

**Note:** The Confirmation Dialog is available from the Template folder.

### Variables

Several system-generated variables are available when a User Interface of Confirmationdialog type is added.

Variable	Type	Description
<b>confirmObject</b>	Object	Returns the confirmed object by executing the confirm action.
<b>model</b>	model	The object representing the data for this user interface (for example, Document, Finder, Order).
<b>content</b>	User Interface	Populates the Confirmation Dialog with the contents of the contentForm when the contentVisible method is invoked.
<b>contentForm</b>	String	The name of the Form that belongs to the content object being used to display the content.
<b>context</b>	User Interface	Used for backward compatibility. Used to display two different objects on the screen.
<b>contextForm</b>	String	The name of the Form which belongs to the context object used to display the context.
<b>dialog</b>	User Interface	The current dialog user interface being displayed. Used by the Form Frame.
<b>owner</b>	User Interface	Detail document of the Confirmation Dialog. It is populated when a selection is performed on the Confirmation Dialog and the XXXX method is invoked.
<b>title</b>	String	The title to display in the Web Browser.
<b>dialogHeight</b>	Integer	The height of the dialog box.
<b>dialogWidth</b>	Integer	The width of the dialog box.
<b>isModal</b>	Boolean	Allows the dialog box to move positioning.
<b>dialogOwner</b>	Object	The user interface that invokes the dialog.

The following variables are available to configure the confirmation dialog. Examples are provided in the [Library](#) tab of Velocity Studio.

Variable	Type	Description
<b>ifContent</b>	Boolean	Indicates whether to display the content form of the message form.
<b>ifYes</b>	Boolean	Indicates whether the Yes button is visible.
<b>ifNo</b>	Boolean	Indicates whether the No button is visible. Depending upon how the user responds to the warning message in the confirmation dialog a confirmation script may be triggered. The confirmation script is executed based on the action of the user.
<b>ifOK</b>	Boolean	Indicates whether the OK button is visible.
<b>ifCancel</b>	Boolean	Indicates whether the Cancel button is visible.
<b>userAction</b>	String	Contains the name of the user action.
<b>message</b>	String	Contains a User Message the format is UU0100:YN. The Y (Yes), N (No) represents buttons
<b>scriptConfirm</b>	Object	This object contains a user message and confirm object.

## User Action Method

The User Action Method enables you to configure the settings and create a script to invoke a user action in a Form. Add a New User Action Method and configure the fields. The Confirmation field enables you to set the confirmation action that displays the message or the Form.

Name:

Is private

Description:

Parameters:

Name	Type

Script:

```
function Y_NVisibleDoc() { // Metadata type method. Can be called by scripts.
```

}

Return:

Object:

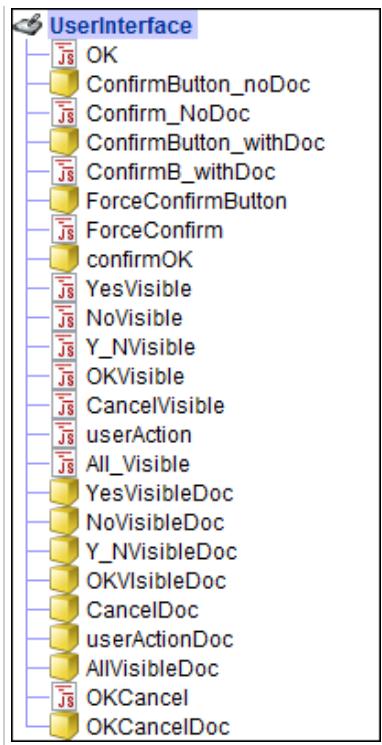
Form:

Confirmation:

Allow invalid object    Autosave    Show in popup    Validate

Field	Description
Name	Name of the Method. Must be unique within the User Interface.
Description	Descriptive text for documentation purposes.
Parameters	Adds parameters in your script. The last parameter must be defined as the object. For example, if the User Action Method is invoked by the User Interface (object), the only parameter that is used is this object.
Script	Any method, including Permission Methods, is considered to be a script Method when it includes a custom JavaScript that is invoked as an action. The User Action script can return a value (a Document, Finder, Order, User Interface, Navigation Tree) and that returned object is set as the content object at the page or pop-up dialog (returning a value overrides the object property).
Return	The object type to be returned by the script.
Object	The object of this type is instantiated when this method is invoked.
Form	The Form to present to user when the object is instantiated. Choices are available and limited only to Forms of the selected object.
Confirmation	Configure a confirmation message that displays the message or the Form before performing the user action.

For each action that you want to display in the Confirmation Dialog, you need to create a Using User Action methods. You can configure the Confirmation field to use a script that triggers the user action.



When a method is triggered, different types of dialogs display and certain variables in the ConfirmDialog are set.

Script Example	Returned Value From Script	Confirmation Dialog Displayed
var doc = new Document("VelMethods:TestConfirm"); return Global.confirmOK(doc);	user message	The contents of the messageDialog Form is displayed in the dialog.
return new Confirm("UU0117:Y");	scriptConfirm object	The contents of the messageDialog Form is displayed in the dialog.
var doc = new Document("VelMethods:TestConfirm"); doc.TestStr = Global.translateText("UU0117", null, null); return new Confirm("UU0117",doc);	confirmObject	The default Form of the confirmObject is displayed.
var string = "Global.getForceConfirm(document) Returned. "; if (this.confirmObject!=null) string += " There was a confirmObject " + this.confirmObject + "."; else string += " There was no confirmObject."; Global.showUserMessage(string);	ForceConfirm object	No dialog is displayed, and the user action is executed.

## Configuring Forms

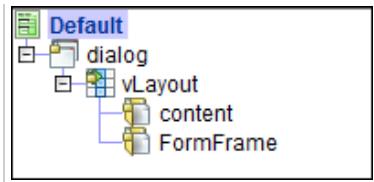
The Confirmation Dialog User Interface contains by default four forms: Default, Help, Menu and messageDialog.



### Default Form

The Default Form contains the Dialog element, and two Form Frames. The dialog element displays a confirmation dialog and the Form Frame elements allows you to configure contents of the dialog.





## Help Form

This default Help Form allows you to configure a link to an external help document or site.

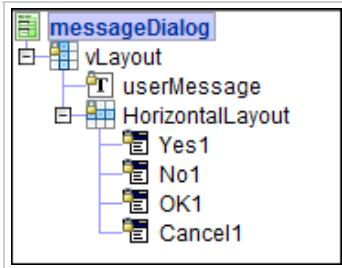
## Menu Form

This default Menu Form allows you to configure additional menu items when displaying the Configuration dialog.

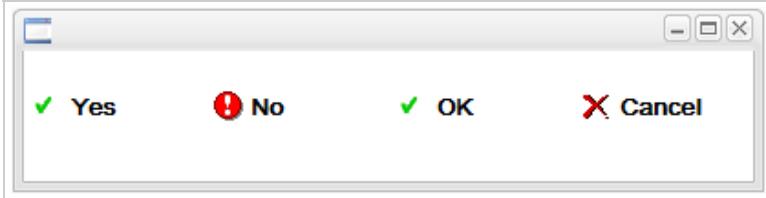
## messageDialog Form

The messageDialog Form consists of:

- a text element that contains the content of the user message
- and menu elements for the user action buttons



The user action buttons, **Yes**, **No**, **OK**, and **Cancel** are executed using a Click Method.



The visibility of the buttons are controlled by the *Visible* property, which is permission based.

## Implement a Confirmation Dialog

You can configure a confirmation dialog by doing the following.

1. In the **Navigation** pane, click the User Interface object that you want to open.
2. On the Variables tab, ensure that this is an object called **ConfirmObject**.
3. Next, click the **Methods** tab.
4. Right-click the **User Interface** node and select **New User Action**.
5. In the **New User Action** dialog, configure the settings for the action that you want to take place.
6. In the **Confirmation** drop-down list box, select the script that you want to point to for the user action.
7. Click **Finish**.

## Libraries

In the **Library** tab, you can view the available layout, including the variables and methods, for the confirmation dialog by navigating to the following path:  
**Library > com > conceptwave > system > Presentation > User Interface > ConfirmDialog**.

## Dynamic Document

Dynamic Documents enable variables or table columns, or both to be created dynamically at runtime. For example, a catalog application the user would create a new category and item for a catalog. Each item might have different attributes that need to be captured at runtime. These attributes can be modeled by a [Dynamic Document](#).

**Note:** Dynamic documents can also be viewed in a form with a predefined layout.

Name	Value
Cell Alignment	
Column Width(s)	
Height	
Label Orientation	top
Name	DynamicDocument
Number of Columns	1
On Enter	
Style	
Tooltip	
Variable	dynDoc
Visible	
Width	

### Summary

<b>references Variable?</b>	Yes
<b>references Method?</b>	Yes, On Enter.
<b>references Form?</b>	No
<b>Allowable Child Elements</b>	Checkbox Date Field Date Time Hyperlink Password Field Radio Button Group Select Field Text Area Text Field Upload File Variable Button Cell Spacer Group Header Image Label Row Spacer Separator

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Dynamic Document
Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Dynamic Document in the page or child elements (that is, Group element). Default is undefined, which behaves as if it is top-left.
Column Width	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Dynamic Document occupies in the Grid Layout, to the right of current cell. Default is empty, which occupy one column.
Dynamic Tooltip	Optional	<p>Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i>. The static Tooltip property is considered as default until the action returns a value.</p> <p>When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.</p>
Dynamic Tooltip Style	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
Height	Optional	Height to be taken up by the Dynamic Document; can be in integer which is in pixels, or in percentage (for example, "50%") of page height or parent element.
Label Orientation	Optional	Specifies where the label of the Dynamic Document child elements appear in relation to the Dynamic Document area; can be <i>left</i> , <i>right</i> or <i>top</i> of the Dynamic Document area. The default orientation is <i>right</i> . However, in a Grid Layout, the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout.
Number of Columns	Optional	Defines the number of columns within the Grid Layout. If this parameter is left undefined, the system creates a grid with 2 columns.
On Enter	Optional	The system calls any script defined in this property. The onEnter property is only met if there is an input field (for example, a textfield, selectfield, and so on) that has focus when the <b>Enter</b> key is pressed. This onEnter property is invoked only if the input field does not have an OnEnter property set. This feature can be used with multiple input fields, but only one onEnter method to fire for all fields.
Style	Optional	Choose from styles within the selected style sheet (located in the Velocity Studio toolbar), which defines the styling of the Dynamic Document such as positioning and color. Default is none.
Tooltip	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Dynamic Document. Default is empty, which means no tooltip for the Dynamic Document.
Tooltip Style	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the dynamic document. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
Tooltip Width (px)	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
Variable	Optional	Choose from the list of <i>Document Variables</i> of the types available in the User Interface to bind to this Dynamic Document. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
Visible	Optional	Determines whether the Dynamic Document is visible or not. Choose true (visible) or false (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this field. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE or True, which means no permissions assigned and field is visible..
		Width to be taken up by the Dynamic Document; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element. If this field is empty,

<b>Width</b>	Optional	the Width of the field should be taken up by that defined in the Column Width property of the Grid Layout.
--------------	----------	--

There are a number of ways to create a Dynamic Document:

- [Create a Dynamic Document with a completely dynamic layout](#)
- [Create a Dynamic Document with a predefined, static layout](#)
- [Create a Dynamic Document with leaf exclusion](#)

## Show Permissions and Validation Errors When Binding a UI Element to a Dynamic Document Variable

To show permissions (for example, Optional, Editable, Visible, and so on) and validation errors on a target field when binding a UI element to a dynamic document variable, override one or all of the following methods in the document:

Method	Description
cwOnValidateDynamicLeaf(leafName)	This method returns either CwfError or null. This method is called when the validate() method calls the document.
cwlIsDynamicLeafEditable(leafName)	This method returns either true or false. If null, the field is editable. This method is called just before the UI update is sent to the Web browser.
cwlIsDynamicLeafVisible(leafName)	This method returns either true or false. If null, the field is visible. This method is called just before the UI update is sent to the Web browser.
cwlIsDynamicLeafOptional(leafName)	This method returns either true or false. If null, the field is optional. This method is called as part of the validation and just before the UI update is sent to the Web browser.

## Specify Date and Time Formats Different from those in Metadata for Dynamic Documents

You can specify date, and date and time formats that are different than those defined in your metadata project for dynamic documents by using [event handlers](#), without the need to either override or change existing objects.

While creating the data for dynamic documents, publishing of these [events](#) occur:

- CW\_GET\_DYNAMIC\_DATE\_FORMAT
- CW\_GET\_DYNAMIC\_DATETIME\_FORMAT

Your application then provides handlers for these events to return a string representation of the formats. In turn, these formats are used in place of the existing display formatting for the date, and date and time fields.

**Note:** It is recommended that you reuse the event results for the duration of processing your dynamic documents, to avoid any recalculation.

To specify these formats using event handlers, do the following:

1. Add the following event handlers for date, and date and time formats, respectively:
  - DYNDOC\_DATE\_FORMAT
  - DYNDOC\_DATETIME\_FORMAT
2. Return some date, and date and time formatting in those event handlers.
3. Create a dynamic document with some fields that include a Date and DateTime data types.
4. Create all relevant metadata that allows showing your dynamic document.
5. At runtime, make sure that the dynamic document's Date and DateTime values are using the display formatting that the event handlers are returning.

# Dynamic Variable

## Create a Dynamic Variable

The createLeaf API of DynamicDocument class can be used to create of a **dynamic variable** (or **dynamic leaf**). The type parameter of createLeaf can be used to specify either an integer leaf data type or a metadata data type name.

The following are examples where an integer leaf data type is passed into createLeaf:

```
var dyn=new DynamicDocument("ns.DynDocMetadata");
dyn.createLeaf("decLeaf","decLeaf",0,0,8,2,null);
dyn.createLeaf("intLeaf","intLeaf",1,0,8,0,null);
dyn.createLeaf("codeTableLeaf","Code Table",3,0,0,0,"codeTableName");
```

The following are examples of using the createLeaf API to create a dynamic variable using a metadata data type name:

```
dyn.createLeaf("strLeaf","strLeaf","ns.stringDataType"); // overrides the data type label
dyn.createLeaf("strLeaf", null, "ns.stringDataType"); // gets label from the data type
```

These statements will throw an exception if the data type is not specified or the string type is not found or not a data type.

Label, length, precision and code table parameters of createLeaf are optional if data type name is used. If these parameters are specified and not null/0, then the parameter value overrides the value from the data type. For example:

```
dyn.createLeaf("strLeaf", "Label", "ns.stringDataType"); // gets specified label
```

## Trigger for a Dynamic Variable

The last parameter of createLeaf can be used to optionally specify a trigger method name. For example:

```
dyn.createLeaf("leafName","Label","leafType",null,null,null,null, "triggerMethod");
```

where triggerMethod is the name of the method of the dynamic document.

**NOTE:** This method must be defined in the document metadata that is used to create the dynamic document. In the provided example, ns.DynDocMetadata metadata must have a script method triggerMethod. The method should have 1 parameter: leafName.

```
function triggerMethod(leafName) {
...
}
```

## How Dynamic Variable Triggers Work

Dynamic variable triggers work very similarly to regular document triggers. The trigger for a dynamic variable is invoked if the data of that dynamic variable is changed and the runTriggers method is fired for the document. runTriggers is a native Java method of the document instance and it is fired every time a dynamic form is submitted to the server (e.g., on every user action).

If a trigger is specified for a checkbox, radio group or dropdown dynamic field, any change of the field in the UI fires the submission of the form and the running of the document triggers for all the changed dynamic variables.

It is also possible to run triggers manually, as shown in the following example:

```
dyn.runTriggers();
```

## Data Type Properties Applied to Dynamic Field

When a data type name is used to create a dynamic variable, the following properties of the data type (if defined) will apply to the dynamic variable:

- Label
- Length
- Precision
- Database code table
- Default value
- Element properties
- Reference
- Display Format
- Auto Format (adding grouping comma for numbers)
- Visible: TRUE/FALSE
- Editable: TRUE/FALSE
- Mandatory style (!Nullable || Optional=FALSE) applies bold style and \*
- Hint

## Supported Element Types

The following is a list of element types which are supported by dynamic variables.

- Text field
- Text area
- Select
- Checkbox
- Radio Button Group
- Date
- Date Time
- Reference

**Note:** The available element types will depend on the data type being used. Not all of these elements types will be available for all data types.

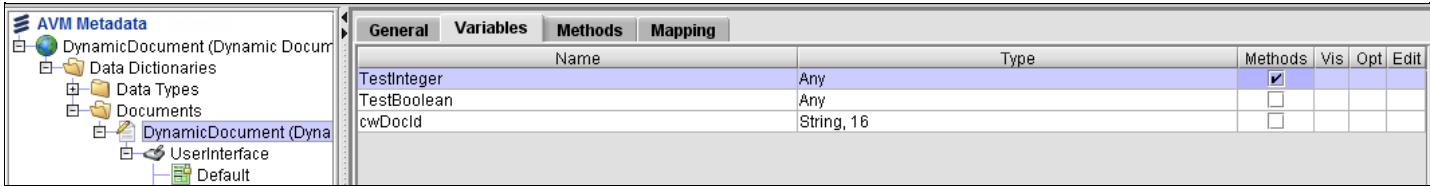
## Dynamic Document with a Static Form Layout

[Dynamic documents](#) can be viewed in a form with either a predefined, static layout or a completely dynamic layout. A predefined layout means that a form can be designed the same way as a regular document. The dynamic document element accepts child elements that can be bound to a dynamic leaf.

For child elements to be bound to a dynamic leaf, the created document needs to either have variables of type Any or cwf.dynamicDatasource. Each variable must have the same name as a dynamic leaf found in the dynamic document's database table. Without this name lookup, the value will not be shown at runtime.

To create a dynamic document with a predefined layout, complete these steps:

1. Create a new document and select the **Dynamic Document** checkbox. For the **Dynamic table name** field, enter the database table that you want to use at runtime.
2. Click the document's **Variables** tab and add variables of type Any with the same names as those found in the database tables. If the type has already been determined, set the proper type that extends cwf.dynamicDatasource.

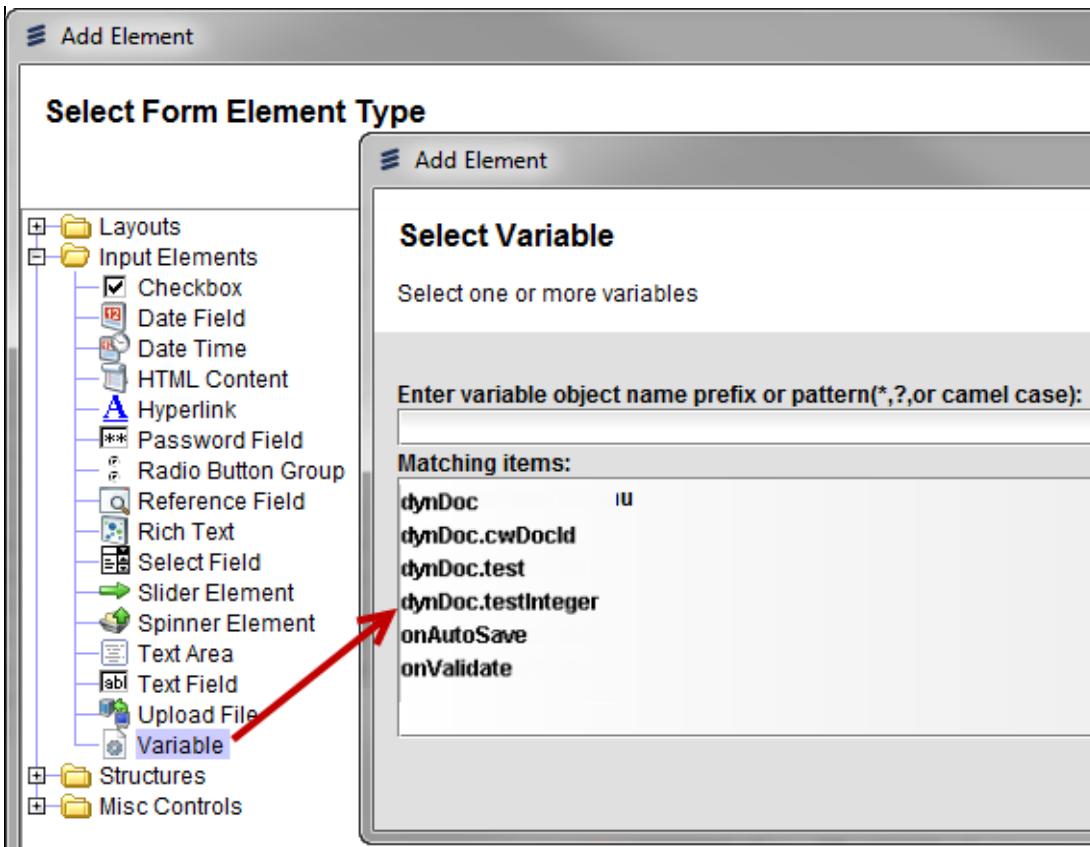


3. Create a new top-level user interface and add a variable for your dynamic document.
  4. Override the `onInit()` method and initialize your variable.  
**Example:** `this.dynamicDocNew = new DynamicDocument( "DynamicDoc.dynDocNew2", testTable, "test5", false, false );`
  5. Navigate to the Default form, right-click it, and then select **Override**. Proceed to add a **Vertical Stack Layout** to the form.
  6. Under this layout, add a Dynamic Document element.
  7. Right-click the Dynamic Document element and add the input elements that you want.
- Note:** These elements must match the variable name and the dynamic leaf.
8. You may choose to display other elements under the dynamic document, such as images or buttons, that are not associated with the dynamic document.
  9. Continue to design the form as required.
  10. Save your changes and then run your metadata.

## Display Static Variables with a Dynamic Document

To display static variables with a dynamic document, complete the following steps:

1. Using the application that you have created in the previous example, navigate to the document that you created in step 1.
2. In the Variables tab, add the necessary static variables. Ensure that you map the document. The static part of this document requires a separate table in the database.
3. Navigate to the user interface Default form that you created in step 5 of the previous section.
4. Right-click the Vertical Stack and add the static fields either through clicking **Input Elements > Variables** from the Select Form Element Type dialog, or by selecting the form element and then specifying the variable to use with it.



5. Save and run your metadata. In runtime, both dynamic and static variables are present. When you have saved your metadata, all input is present in the databases in the corresponding tables.

## Dynamic Document with a Dynamic Form Layout

A completely dynamic form is the simplest method to display a dynamic document and its dynamic leaves. The product automatically gathers all dynamic leaves found in the NVP table and displays them based on the data type assigned.

To create a [dynamic document](#) with a dynamic layout, complete these steps:

1. Create a static document and define it as dynamic in the general folder.

The screenshot shows the 'Variables' tab of a document configuration interface. The 'Name' field is set to 'dynDoc', and the 'Label' field is also 'dynDoc'. Under 'Extends', it says 'com.conceptwave.system.Document'. The 'Overrides' field is '**<NONE>**'. The 'Dynamic document' checkbox is checked, and the 'Dynamic table name' input field contains 'dynTable'. Below these fields are several checkboxes: 'Audit trail', 'Attachments', 'Generated key' (which is checked), 'Strict valid', and 'Timestamp sync'. A red box surrounds the 'Dynamic document' checkbox and the 'Dynamic table name' input field. A yellow arrow points from the text 'Database Table' to this red box. The 'Keys' field contains 'cwDocId'.

2. Define your dynamic table (*Dynamic table name* (for example, dynTable as shown in the previous screenshot)) in your database with the following format:

Column Name	Type	Description
DOC_ID	VARCHAR2(36)	Contains the GUID or ID of the document.
PARENT_ID	VARCHAR2(16)	This column stores the basket item ID and is used for the catalog basket item only. However, this column is mandatory, even for a general dynamic document, due to an update database statement that includes this column.
DOCUMENT_TYPE	VARCHAR2(38)	GUID of the document type.
LEAF_NAME	VARCHAR2(64)	The dynamic Document variable name.
ARRAY_INDEX	NUMBER(4)	The array size. 0 for single values. 1 to 9999 for arrays.
DATA_TYPE	NUMBER(2)	Data type of the dynamic Document variable. <ul style="list-style-type: none"><li>o 0 - Decimal</li><li>o 1 - Integer</li><li>o 3 - String</li><li>o 4 - Boolean</li><li>o 5 - Date_Time</li><li>o 6 - Date</li><li>o 9 - Code Table</li></ul>
VALUE	NVARCHAR2(256)	The value stored as string. Details are defined in below section.

3. Perform any necessary database mappings.
4. To display the dynamic document, create a top-level User Interface.
5. In the **Variables** tab, add a variable of type DynamicDoc as created in step 1.
6. Click the **Methods** tab and override the `onInit()` method.

7. The syntax for the [DynamicDocument](#) constructor is as follows:

```
new DynamicDocument(docFullName, nvpTableName, docId, optionalThrowException, runLoadScript)
```

**Note:** The nvpTableName parameter is optional. If you want to use a different database table from that set in the document object in the metadata, you can pass it in the constructor. Otherwise, a null value is suitable.

Initialize the document variable as follows:

```
var docVar = new DynamicDocument("DynamicDoc:dynamicDoc", testTable, "5", false, false);
```

8. The syntax for the createLeaf constructor is as follows:

```
createLeaf(leafName, leafLabel, type, arraySize, length, precision, codeTable, triggerMethod);
```

The last parameter, trigger method, is optional.

The type parameter of createLeaf is used to specify either an integer leaf data type or a metadata data type name. When specifying an integer for the type parameter, the following integer data types are supported:

- o Decimal = 0
- o Integer = 1
- o String = 3
- o Boolean = 4
- o Date\_Time = 5
- o Date = 6
- o Void = 7
- o Timestamp = 8
- o Code Table = 9

Create the following document leaves:

```
docVar.createLeaf("testDecLeafName", "testDecLeaf", 0, 0, 4, 0, null);
docVar.createLeaf("testIntLeafName", "testIntLeaf", 1, 0, 4, 0, null);
docVar.createLeaf("testStringLeafName", "testStringLeaf", 3, 0, 4, 0, null);
docVar.createLeaf("testBooleanLeafName", "testBooleanLeaf", 4, 0, 4, 0, null);
docVar.createLeaf("testDateLeafName", "testDateLeaf", 6, 0, 4, 0, null);
```

The following is an example of how a metadata data type name can be passed for the type parameter:

```
dyn.createLeaf("strLeaf", "strLeaf", "ns.stringDataType"); // overrides the data type label
dyn.createLeaf("strLeaf", null, "ns.stringDataType"); // gets label from the data type.
```

This will throw an exception if the data type is not specified or the string type is not found or not a data type.

For more information on creating and using a dynamic variable (or dynamic leaf), see [Dynamic Variable](#).

9. Override the user interface's default form in [step 4](#).

10. Add a **Vertical stack layout** element. Right-click the element and click **Add**.

11. Select **Structures > Dynamic Document**, and then click the **Finish** button.

12. Set the dynamic document element's **Variable** property to the variable you added in step 5.

13. Add a menu element in your application menu to display this user interface.

14. Save your metadata, and click **Runtime > Run** from the menu bar.

The following is an example of an onInit script that calls the DynamicDocument object and the createLeaf method:

onInit Script	Runtime
<pre>this.doc=new Document("DynamicDoc:dynamicDoc"); var dyn=new DynamicDocument("DynamicDoc:dynamicDoc",testTable,this.doc.cwDocId,false,false);</pre>	

```

dyn.createLeaf("decLeaf","decLeaf",0,0,8,2,null); // Decimal leaf
dyn.createLeaf("intLeaf","intLeaf",1,0,8,0,null); // Integer leaf
dyn.createLeaf("strLeaf","strLeaf",3,0,10,0,null); // String leaf
dyn.createLeaf("booLeaf","booLeaf",4,0,1,0,null); // Boolean leaf
dyn.createLeaf("datLeaf","datLeaf",6,0,10,0,null); // Data leaf
dyn.save();
this.dynDoc=dyn;
this.dynDoc.save();
this.dynDoc.decLeaf=100.2;
this.dynDoc.intLeaf=12345678;
this.dynDoc.strLeaf="Dynamic ";
this.dynDoc.booLeaf=true;
this.dynDoc.datLeaf=Calendar.parseDate("2010-06-06","yyyy-MM-dd");
this.dynDoc.save();

```

## Dynamic Document

### Dynamic Leaves

decLeaf	intLeaf
100.20	12345678
strLeaf	booLeaf
Dynamic	<input checked="" type="checkbox"/>
datLeaf	
6/6/2010	

## Save a User Entry with a Dynamic Document

This section continues with the example used in the previous section. To save a user entry with a dynamic document, do the following:

1. Navigate to the user interface created in [step 4](#).
2. In the **Methods** tab, create a new user action called **save**. In this script, only the following line of code is required:

```
this.docVar.save();
```

3. Navigate to the default form and add a new menu element. Set its **Click method** property to the **save** method from step 2.

**Note:** Set the label or icon of this menu to see it during runtime.

4. Save your metadata, and click **Runtime > Run** from the menu bar.
5. In runtime, edit the values of the fields and then click the **Save** button. The database contains the new data created in [step 2](#) of the previous section.

## Dynamic Document with Leaf Exclusion

---

[Dynamic document](#) elements can be designed so that not all dynamic leaves in the NVP table are displayed at runtime. A script that accepts a parameter type of String and a return type of Boolean may be used to determine which leaves are to be displayed and which should remain hidden.

To create a dynamic document with leaf exclusion, complete these steps:

1. Create the same scenario as described in the [Dynamic form](#) example, including the saving process, except without the `createLeaf` calls in the script.
2. Navigate to the **Methods** tab of the user interface and create a new script that returns type Boolean (mandatory to select this script). Proceed to add a parameter of type string to represent the leaf name.
3. Create a sample script that ensures that only some dynamic leaves are displayed at runtime:

```
if("testDecLeafName" == leafNameParam || "testIntLeafName" == leafNameParam ){  
    return true;  
}  
return false;
```

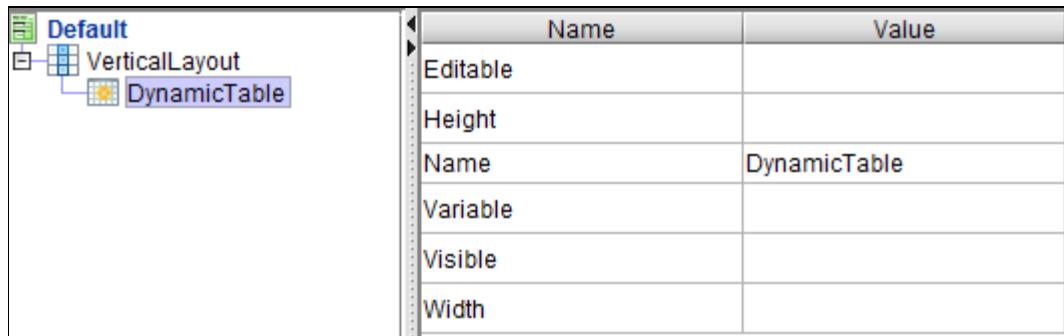
The script must return a Boolean value. A true value excludes that leaf; otherwise, an error occurs.

4. Navigate back to the dynamic document element and set its **Filter method** property to the script you created in step 3.
5. Save your metadata, and click **Runtime > Run** from the menu bar. The output shows three of the five dynamic leaves.
6. Before the dynamic document is loaded during runtime, the script created in step 3 is called for all available leaves in the NVP table. Ensure that this logic is what you need for your application form.

## Dynamic Table

The Dynamic Table element allows data columns to be created dynamically at runtime. The dynamic table element is used in conjunction with the Catalog Designer's Info Table object, to work with dynamic tables at runtime. You can use the *DynamicTable* object and related methods to add, edit, or remove rows in the dynamic table without any need of Catalog Designer's Info Table. Refer to [JavaScript Documentation](#) for more details.

Dynamic table element uses proper UI controls for the columns when editing. For example, if catalog Info table has a column of type code table, dynamic table element displays the drop-down list with the options described in code table.



The following table describes the properties of dynamic table:

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Dynamic Table element.
Editable	Optional	<p>This property determines whether user input is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user, editing the table will be enabled or disabled).</p> <p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.</p>
Header Height	Optional	This property is of type Integer. This value is written as part of the binding since table creation is on demand. Additionally, the text in the table's header row wraps automatically, if needed.
Height	Optional	Height to be taken up by the Table area; can be an integer which is in pixels, or in percentage (for example, "50%") of page height or parent element.
Include in Tab Order	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"><li>When set to &lt;Default&gt;, the element is included in tab order according to global rules and its focusable state.</li><li>If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li><li>If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li></ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
Variable	Optional	Choose from the list of Variables available in the User Interface to bind to this Table. Default

		is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Determines whether the Dynamic Table is visible or not. Choose true (visible) or false (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this field. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE or True, which means no permissions assigned and field is visible.
<b>Width</b>	Optional	Width to be taken up by the Dynamic Table; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element.

## Form Frame

Form Frame is a reference Element that displays a Form.

Form Frame is a special and important Element. Use Form Frame to include Document, Order, and Finder Forms, and other Forms of pre-built common UI objects such as Menus and Trees, to assemble your Web page, analogous to stitching UI widget components together. A Form Frame does not add any decoration to the user interface.

The following is an example of a Form Frame referencing a Document Form. Notice that the Form Frame itself, indeed, has no visible frame or border around the Form.

The screenshot shows a form element with the title "Handsets". It contains several input fields and dropdown menus:

- A text input field labeled "name" with a placeholder.
- A dropdown menu labeled "manufacturer" with options: Nokia, Ericsson, Apple, and Research in Motion. The option "Research in Motion" is selected.
- A text input field labeled "modelNumber" with a placeholder.
- A checkbox labeled "enabled3G".

### Summary

references Variable	Yes
references Method	No
references Form	Yes
Allowable Child Elements	<i>None</i>

The following table describes the properties of form frame element:

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Form Frame.
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Section Stack in the page or parent Element. Default is undefined, which behaves as if it is top-left.
Dynamic Height	Optional	This property allows you to dynamically assign a variable or a script to the <b>Dynamic Height</b> property. For a variable, the data type must be either an integer or string. For the script, only the script action can be used and its return type must be defined as either integer or string.  At runtime, by default, the regular width and height are used. However, if <b>Dynamic Width</b> and <b>Dynamic Height</b> are defined, they replace the regular width and height, and the values are calculated based on either the variable or script.
Dynamic Style	Optional	This property allows you to set a dynamic style for a form frame element, which avoids multiple layers of layouts.
Dynamic Tooltip	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.  When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display

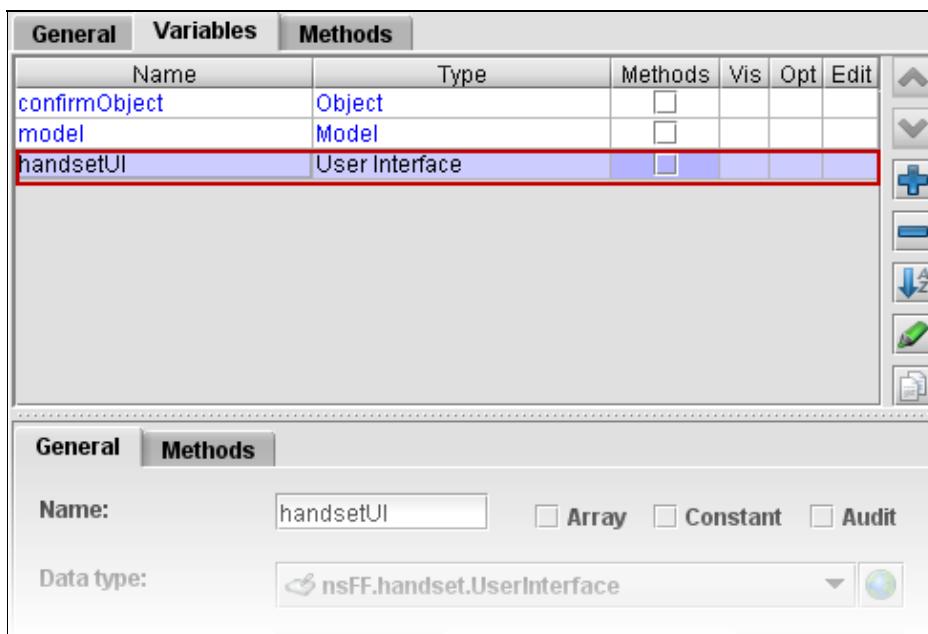
		the tooltip.
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Dynamic Width</b>	Optional	<p>This property allows you to dynamically assign a variable or a script to the <b>Dynamic Width</b> property. For a variable, the data type must be either an integer or string. For the script, only the script action can be used and its return type must be defined as either integer or string.</p> <p>At runtime, by default, the regular width and height are used. However, if <b>Dynamic Width</b> and <b>Dynamic Height</b> are defined, they replace the regular width and height, and the values are calculated based on either the variable or script.</p>
<b>Error Icon Gap</b>	Optional	<p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>External URL</b>	Optional	Type the URL for the external site. Note, the Form Frame width and height must be defined to implement this property. Form Frames with variables and form property settings cannot have an External URL.
<b>Form</b>	Optional	<p>Choose from a list of available forms that this form frame presents. If the form frame is assigned to a Variable property, the choices are limited to theForms of that Variable. If Variable property is not assigned, the choices are limited to forms in the current User Interface.</p> <p>Default is not assigned where no form is displayed. Alternatively, a local string variable can be used to dynamically change the form to be used at runtime. This string variable should have <i>only</i> the name of the form such as "Default". Setting this property to a string variable automatically uses the user interface of the Variable property to look for the form if it is provided. Otherwise, it will use the current user interface for the form lookup. This property also allows you to specify a method that returns a String.</p> <p><b>Note:</b> If metadata is migrated from different version 5.x releases and property is set to the <i>Form.Default</i>, the migration process will reset the value to <i>Default</i>.</p>
<b>Height</b>	Optional	Height to be taken up by the Form Frame; can be in integer which is in pixels, or in percentage (for example, "50%") of page height or parent Element.
<b>Icon Height</b>	Optional	Height of the icon used or consumed within the element. The value is defined as the number of pixels (integer) or percentage. The default value is 0.
<b>Icon Tooltip</b>	Optional	Click the ellipsis button and specify a tooltip description within the <b>Value</b> field for the icon.
<b>Icon Width</b>	Optional	Width of the icon used or consumed within the element. The value is defined as the number of pixels (integer) or percentage. The default value is 0.
<b>On Enter</b>	Optional	The system calls any script defined in this property. The onEnter property is only met if there is an input field (for example, a textfield, selectfield, and so on) that has focus when the <b>Enter</b> key is pressed. This onEnter property is invoked only if the input field does not have an OnEnter property set. This feature can be used with multiple input fields, but only one onEnter method to fire for all fields.
<b>Overflow</b>	Optional	<p>Defines how overflow text is displayed based on the height and width of the Form Frame. Choose from auto, hidden, or visible options in the toolbar. The auto option displays text overflow as a child of the Form Frame. The default is NONE.</p> <p>See the <a href="#">Set form frame height dynamically based on height</a> section for more details on this property and an example.</p>
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Show Resize Bar</b>	Optional	If true, a resize bar is created for the form frame that allows you to resize the form frame by dragging it, and show/hide layout by clicking it. The orientation of the resize bar for the form frame

		yields an up-and-down resize bar. The length of the resize bar is dictated by the width of the parent. Specifically, it is the height if either the vertical layout parent or the width of the horizontal layout parent that determines the resize bar's length. Default is <i>default</i> , which does not show the resize bar.
<b>Style</b>	Optional	Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the Form Frame such as color and font. Default is NONE.
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Form Frame. Default is empty, which means no tooltip for the Form Frame.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the form frame. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Unmanaged</b>	Optional	Specifies whether the Form Frame is displayed under a dialog element. The default is set to false.
<b>Visible</b>	Optional	Determines whether the Form Frame is visible or not. Defaults to NONE, which means visible.
<b>Variable</b>	Optional	<p>Choose from the list of Variables of type <i>User Interface</i> in the current User Interface to bind to this Form Frame. Once a Variable is chosen, you can choose a Form to present the Variable in the Form property. If Form property is not selected with a chosen Variable, the Form with name <i>Default</i> (that is, &lt;Variable's data type&gt;.UserInterface.Form.Default) is assumed. Default is not assigned.</p> <p>You can also specify a method returning either a UserInterface or any particular UI. The <b>Form</b> drop-down list is then populated according to the return type of the method.</p> <p>Use the property's <a href="#">lookup button</a> to view property details.</p>
<b>Width</b>	Optional	Width to be taken up by the Form Frame; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element.

## Notes:

The **Form** property is the most significant property in a Form Frame, which defines which Form to display. You may display a Form that is contained within the current User Interface, or a Form that is outside the current User Interface.

- To display a Form of the same User Interface, simply choose that Form in the **Form** property. Because the Form Frame and the chosen Form are in same User Interface, they share, and thus have access to, the same Variables and Methods in the User Interface.
- To display a Form outside the User Interface, you must first add the target Form's User Interface as a Variable of the current User Interface.



Then assign that Variable to the Form Frame's **Variable** property, and assign the target Form to Form Frame's **Form** property.

The field elements of the Form in the Form Frame may be bound to Variables. At preview time, the value of these variables are null. At runtime,

the assignment of these Variables will be reflected in the rendered page.

Form Frame at preview time		Form Frame at runtime	
<b>Handsets</b>		<b>Handsets</b>	
name	<input type="text"/>	name	<input type="text" value="iPhone 3G"/>
	<input checked="" type="radio"/> Nokia		<input type="radio"/> Nokia
	<input type="radio"/> Ericsson		<input type="radio"/> Ericsson
manufacturer	<input type="radio"/> Apple	manufacturer	<input checked="" type="radio"/> Apple
	<input type="radio"/> Research in Motion		<input type="radio"/> Research in Motion
modelNumber	<input type="text"/>	modelNumber	<input type="text" value="12345678912345"/>
enabled3G	<input type="checkbox"/>	enabled3G	<input type="checkbox"/>

At runtime, if the referenced Form is located outside the User Interface, the Form cannot be rendered (and thus does not show up) without instantiation of the *User Interface* Variable that is added to your User Interface. (Note: that Variable is null by default.) For example, if the *User Interface* Variable is setup as per above screen shot, you may instantiate the Variable by script

```
this.handsetUI = new nsFF.handset.UserInterface(null, this);
```

at the Variable's Initialization method. The Form Frame would then appear at runtime. To populate any data to the Form, assign values to the corresponding Variables under the *User Interface* Variable. In our illustration, an example of the assignment statement may be:

```
this.handsetUI.model.name = "iPhone 3G";
```

See [cascading User Interface objects](#) in Scripting for details.

\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

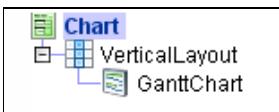
### Set Form Frame's Height Dynamically Based on Content

The form frame is automatically expanded based on the size of its content when the **Overflow** property is set to **DEFAULT**. To automatically expand it to at least one level of children in this form frame, the size must be defined. Otherwise, the form frame does not know how much to expand by.

As an example, if the form frame contains a tree element and you want to automatically expand it based on the number of records in the tree, set the [tree element's Auto Fit Data](#) property to be **Vertical**.

## Gantt Chart

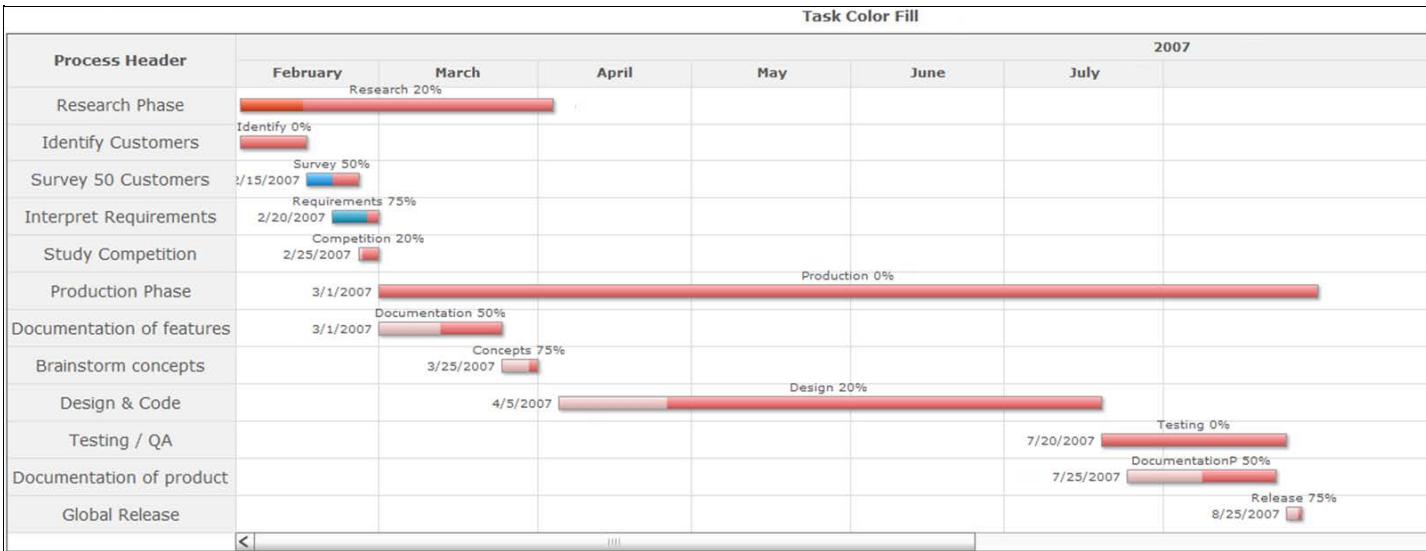
A Gantt chart is a type of bar chart that shows a project schedule, including the start and finish dates of each task belonging to a project. The Gantt Chart element enables you to configure the overall look and feel of the chart display at runtime. The Gantt Chart exists under a finder user interface, relying on its result variable of type document array. Each document in the array represents a new task in the Gantt chart.



Velocity Studio contains the GanttDocument system variable, which includes seven variables that form a template structure for your Gantt chart. The following table shows the required columns that comprise your document:

Name	Description	Nullable?	Unique?
processName	The name of the process displayed in the Gantt chart.	No	Yes
taskName	The name or label of each task in the chart.	No	Yes
startDate	The start date of the task	No	No
endDate	The end date of the task	No	No
percentComplete	The completion percentage of the task	Yes	No
connectToTask	Another task to draw a line starting from this task	Yes	No
taskColorFill	The colour to fill the completed part of the task	Yes	No

The following is an example of a Gantt chart at runtime:



### Display Multiple Charts at Runtime

You can display multiple charts in a Form by creating multiple Form Frame Form elements and then referring the variable of each element to the Finder User Interface Chart Form that contains the chart that you want to display. Refer to [Displaying multiple charts at runtime](#) for details.

### Summary

references Variable	Yes
references Method	No
references Form	No

### Properties

The following are the properties for the Gantt chart element:

Property	Mandatory/Optional	Comment
----------	--------------------	---------

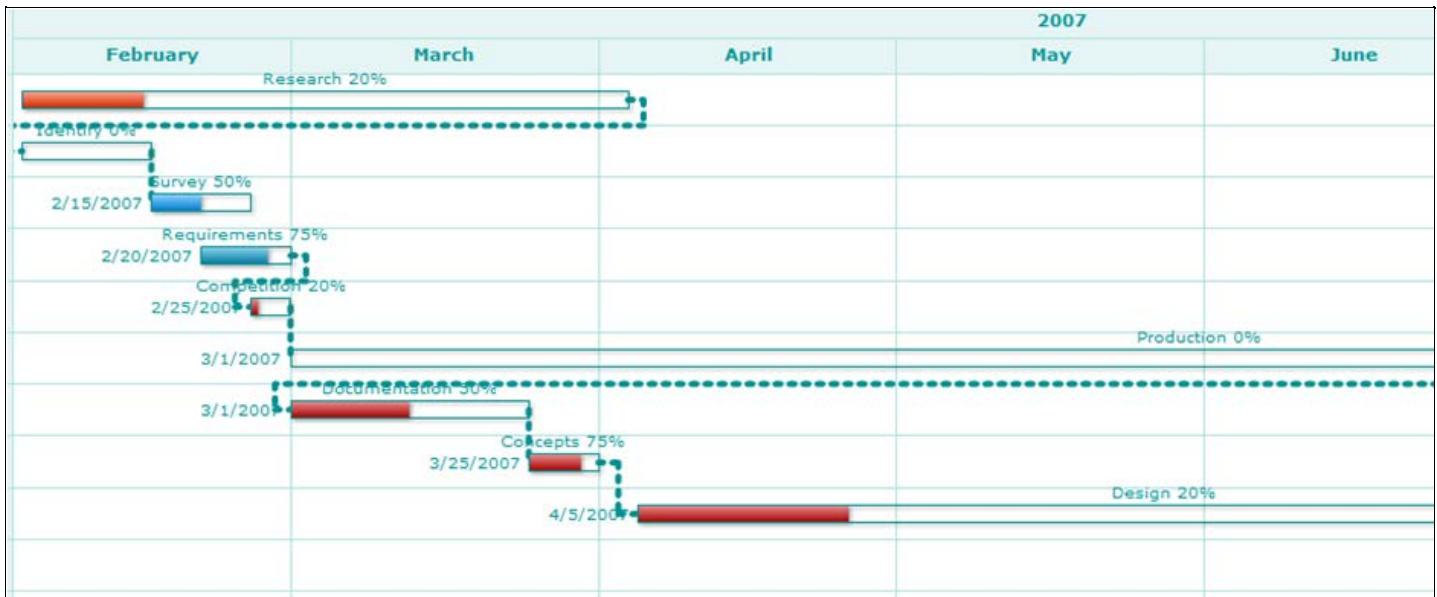
<b>Name</b>	Mandatory	The Gantt chart name.														
<b>Variable</b>	Mandatory	Finder result form for the Gantt chart. Use the property's <a href="#">lookup button</a> to view property details.														
<b>Caption</b>	Optional	The title (label) of the Gantt chart.														
<b>Click Method</b>	Optional	When you specify this property, chart elements are clickable. Clicking these elements invokes the method. The method receives two parameters: <ul style="list-style-type: none"> <li>The data object that the chart node represents</li> <li>The variable name, if applicable, that the chart node represents</li> </ul>														
<b>Connector Thickness (max 10)</b>	Optional	The line thickness that connects two tasks in the Gantt chart. The higher the number, the thicker the connection line is between tasks. This property's default value is 2. <p><b>Note:</b> If you enter a number that exceeds 10, the Velocity Studio automatically overrides this value and sets this field to 10.</p> <p>See the <a href="#">Set Up Connectors</a> section for details.</p>														
<b>Date Format</b>	Optional	The date format used in the Gantt chart. The default format is <i>mm/dd/yyyy</i> . Click the drop-down menu in the <b>Value</b> field to change the date format.														
<b>Dynamic Height</b>	Optional	This property allows you to dynamically assign a variable or a script to the <b>Dynamic Height</b> property. For a variable, the data type must be either an integer or string. For the script, only the script action can be used and its return type must be defined as either integer or string. <p>At runtime, by default, the regular width and height are used. However, if <b>Dynamic Width</b> and <b>Dynamic Height</b> are defined, they replace the regular width and height, and the values are calculated based on either the variable or script.</p>														
<b>Dynamic Width</b>	Optional	This property allows you to dynamically assign a variable or a script to the <b>Dynamic Width</b> property. For a variable, the data type must be either an integer or string. For the script, only the script action can be used and its return type must be defined as either integer or string. <p>At runtime, by default, the regular width and height are used. However, if <b>Dynamic Width</b> and <b>Dynamic Height</b> are defined, they replace the regular width and height, and the values are calculated based on either the variable or script.</p>														
<b>Dynamic Pane Duration Unit</b>	Optional	This dynamic property can be set to a method returning either a String or a String type variable, which allows you to set this property from runtime. This property gets translated into the <code>ganttPaneDurationUnit</code> property, which takes the following values: <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Code</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>y</td> <td>Year</td> </tr> <tr> <td>m</td> <td>Month</td> </tr> <tr> <td>d</td> <td>Day</td> </tr> <tr> <td>h</td> <td>Hour</td> </tr> <tr> <td>mn</td> <td>Minute</td> </tr> <tr> <td>s</td> <td>Second</td> </tr> </tbody> </table> <p>These values must be used as the code table.</p> <p>The following is an example of using this property:</p> <ol style="list-style-type: none"> <li>Create a data type that extends String. As the Enumeration, create the aforementioned code table.</li> <li>In the UserInterface, create a variable with this data type.</li> <li>Override the Menu form.</li> <li>Add a Select Field element and assign the newly created variable.</li> <li>Set the <b>Run Trigger</b> property to <b>showChart</b>.</li> <li>Go to the Gantt chart Form and set the <b>Dynamic Pane Duration Unit</b> property to the same variable.</li> </ol>	Code	Description	y	Year	m	Month	d	Day	h	Hour	mn	Minute	s	Second
Code	Description															
y	Year															
m	Month															
d	Day															
h	Hour															
mn	Minute															
s	Second															
<b>Dynamic Suppress Dependencies</b>	Optional	You can set this dynamic property to either a Boolean or a method returning a Boolean type. This property can be set during runtime. It suppresses displaying dependencies connectors in the Gantt chart. <p>The following is an example of using this property:</p> <ol style="list-style-type: none"> <li>Create a Boolean type variable in the UserInterface.</li> <li>Override the Menu form.</li> <li>Add a Checkbox element and assign the newly created variable.</li> <li>Set the <b>Run Trigger</b> property to <b>showChart</b>.</li> </ol>														

		5. Go to the Gantt chart Form and set the <b>Dynamic Suppress Dependencies</b> property to the same variable.
<b>Gantt Width Percent</b>	Optional	<p>This static property gets translated as <code>ganttWidthPercent</code>, which can be set as a percentage between 0 and 100. The Gantt chart consists of two parts:</p> <ul style="list-style-type: none"> <li>• The Gantt chart</li> <li>• The data table, including the process name</li> </ul> <p>This attribute allows you to set the width of the Gantt part, in percentage, with respect to the entire chart. If the Gantt takes less space, the grid automatically gets more space for the process part. As a result, it is inversely allowing you to control the area width for process names.</p>
<b>Height</b>	Optional	<p>The height of the Gantt chart. Click the <b>Value</b> field, and then select whether you want the height represented in pixels or a percentage using the drop-down menu.</p> <p><b>Note:</b> Although the Height property does not have a maximum limit, specifying a large pixel or percentage in this field may show a black screen during runtime. As a result, it is recommended that you select an appropriate height value for your Gantt chart.</p>
<b>Palette Color</b>	Optional	<p>The base colour for the entire Gantt chart. Based on the colour specified for this property, the Gantt chart's border colours, font colours, connector colours, and more are automatically changed. Click the <b>Ellipsis</b> button in the <b>Value</b> field and select a colour from the palette provided. To remove an existing colour selection, click the <b>Delete</b> button. See the <i>Notes</i> section for additional details about palettes.</p>
<b>Pane Duration</b>	Optional	<p>This field specifies how many of the duration units specified in the <b>Pane Duration Unit</b> field to show in the Gantt chart's main screen without having to scroll. For example, if you specify a duration of one month, the Gantt chart only shows the first month and scrolls the rest. By default, this field contains the value of 1 year.</p>
<b>Pane Duration Unit</b>	Optional	<p>Click this field's drop-down menu allows you to specify the following duration units:</p> <ul style="list-style-type: none"> <li>• Year</li> <li>• Month</li> <li>• Day</li> <li>• Hour</li> <li>• Minute</li> <li>• Second</li> </ul>
<b>Process Font Style</b>	Optional	<p>The font style used for process names (labels). Click the <b>Ellipsis</b> button and select from the following font styles:</p> <ul style="list-style-type: none"> <li>• <b>Font family</b> (for example, Arial, Verdana, and so on)</li> <li>• <b>Colour</b> (click the <b>Ellipsis</b> button to select a colour from the palette)</li> <li>• <b>Size</b> in pixels</li> <li>• <b>Bold</b>, <b>Italic</b>, and <b>Underline</b> styles (click the drop-down menu for each option to select <b>Yes</b>; the default value for each option is <b>No</b>).</li> </ul> <p>Click the <b>Save</b> button to save your font settings.</p>
<b>Process Header Label</b>	Optional	The process header title (label) that appears above the process names.
<b>Show Percent Label</b>	Optional	Select this checkbox to display the percentComplete value for each task in the Gantt chart.
<b>Show Slack As Fill</b>	Optional	Select this checkbox to show the percentage of work that has not yet been completed for each task in the Gantt chart.
<b>Show Task End Date</b>	Optional	Select this checkbox to display the end date of each task in the Gantt chart.
<b>Show Task Name</b>	Optional	Select this checkbox to show the name of each task in the Gantt chart.
<b>Show Task Start Date</b>	Optional	Select this checkbox to display the start date of each task in the Gantt chart.
<b>Show Validation</b>	Optional	This boolean property has values of either <i>True</i> or <i>False</i> , which determines whether a validation warning message displays at runtime. By default, this property is unchecked, meaning that validation errors do not appear under the chart at runtime and supports backward compatibility. Otherwise, selecting this property allows validation errors to appear under the chart at runtime.
<b>Slack Fill Color</b>	Optional	If provided, the percentage provided as the <code>percentCompleted</code> value is displayed with the specified fill colour.
<b>Visible</b>	Optional	This field defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> , such as <code>addPerm</code> , <code>attachPerm</code> , <code>deletePerm</code> , <code>editablePerm</code> , and <code>visiblePerm</code> can be assigned as well as Object

		Permissions. This field defaults to NONE, which means that no permissions are assigned.
Width	Optional	The width of the chart. Click the <b>Value</b> field, and then select whether you want the width represented in pixels or a percentage using the drop-down menu.

## Set Up Connectors

Gantt charts can be set up to have connections between tasks. These connections are data-driven and can be set by providing a value for the `connectToTask` variable. You must ensure that a task with the exact name exists in the chart; otherwise, the connector will not show. If the proper data is provided, the connector will be displayed from the end of the current task to the beginning of the specified task. The following is an example of task connectors within a Gantt chart:

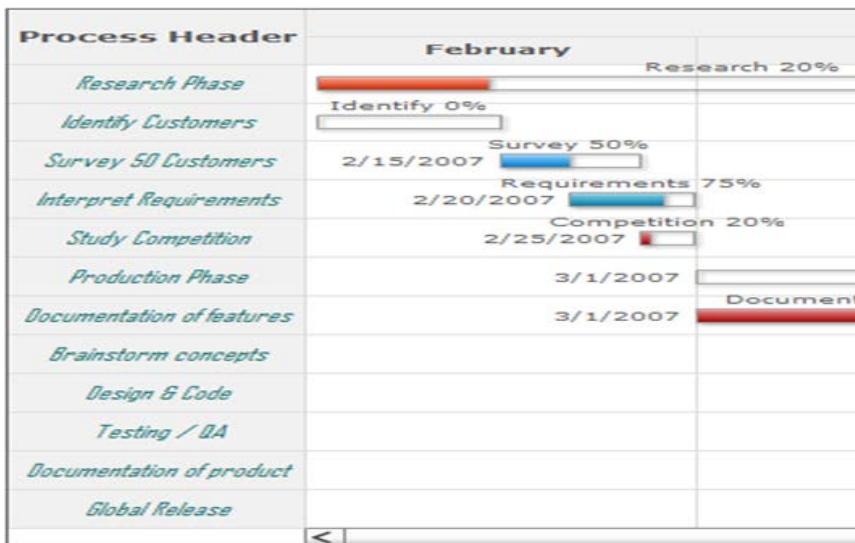


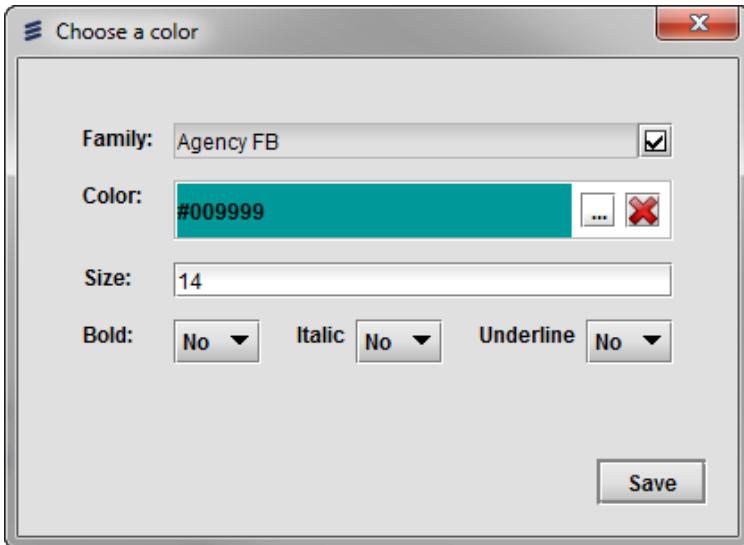
The thickness of each connector can be set using the Gantt chart's **Connector Thickness** property.

**Note:** Connectors can also be set up from the first task to any other task within the chart, meaning that they are not limited to one task jump.

### Notes:

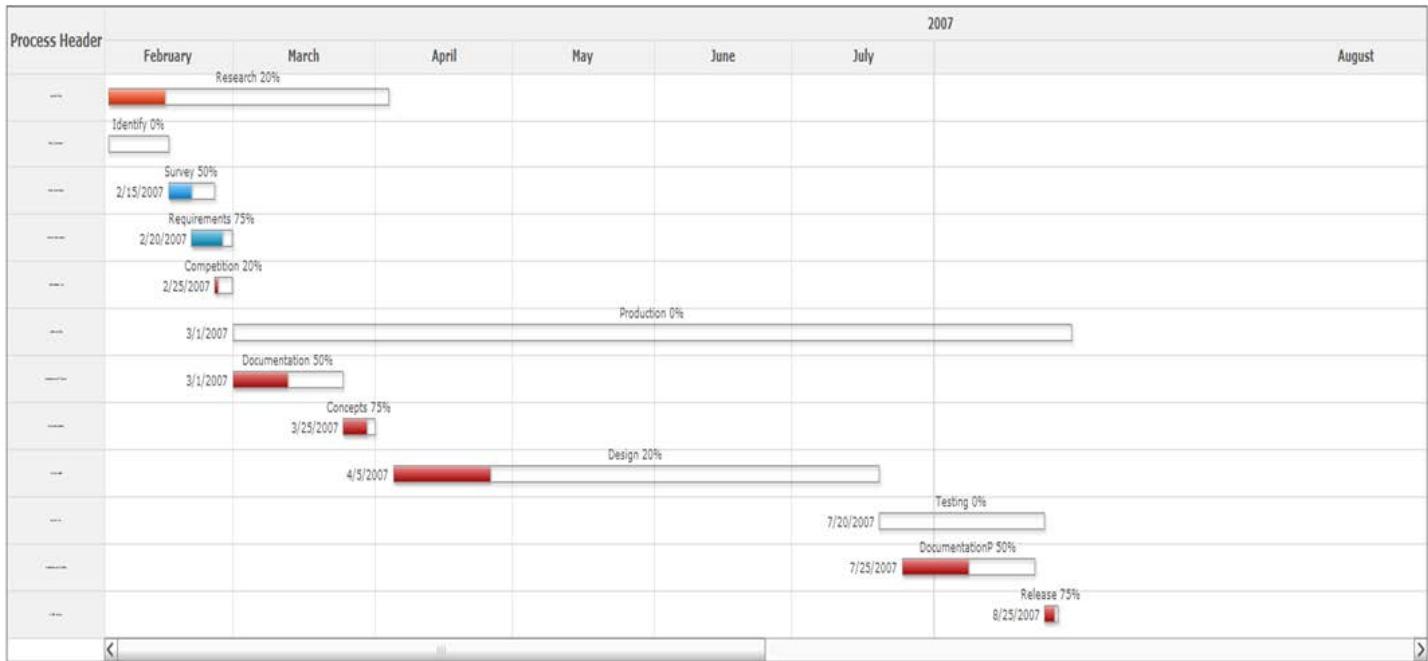
- The font style of process labels can be changed using the Gantt chart's **Process Font Style** property, as shown in these examples:



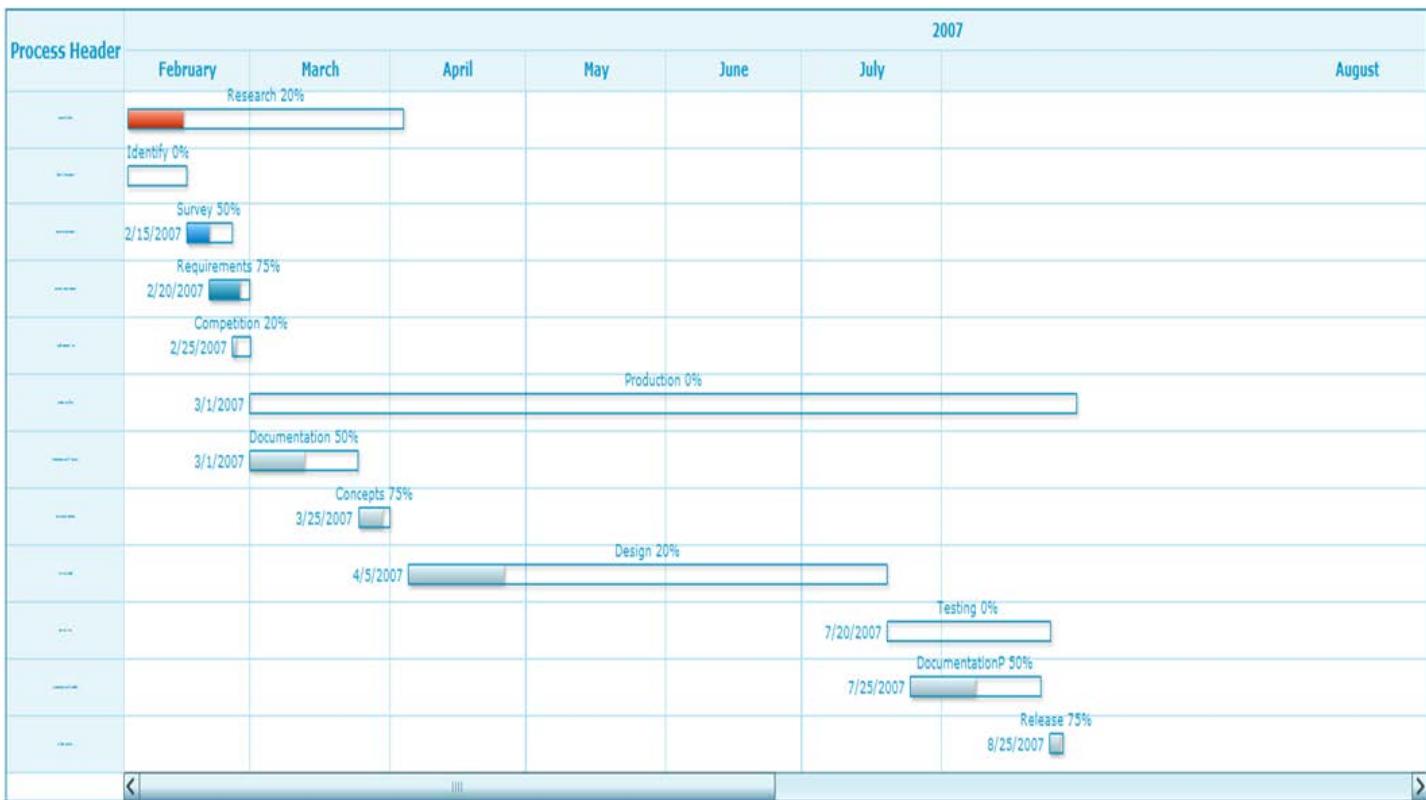


- Each Gantt chart can have its base colour customized using the **Palette Color** property. Based on the colour specified for this property, the Gantt chart's border colours, font colours, connector colours, and more are automatically changed.

This example shows the Gantt chart using a base palette with modified task colour fills:



This example shows the Gantt chart with a modified palette:



- To view charts, Adobe Flash Player 8.0 or higher must be installed for the Firefox browser. Adobe Flash Player 8.0 or higher as an Active X object must be installed for Internet Explorer. Adobe Flash Player can be downloaded from Adobe's Web site at [www.adobe.com](http://www.adobe.com).
- To view JavaScript-based charts, FusionCharts automatically renders them on computers that do not have a Flash player installed. The only client-side requirement is a browser that supports JavaScript.

## Grid Layout

---

There are three layout directives for user interface design:

- Horizontal
- Vertical
- Grid

Grid Layout presents child elements in a grid. Child elements are placed from top-left, shifting across, then down. The property **Number of Columns** defines the column span of the Grid All other Grid Layout properties are defined below

**Customer Profile**

First Name	Last Name	
Phone Number	Email Address	Website
Customer Type	Member Since Date	Active Customer?
Notes: <div style="border: 1px solid #ccc; height: 100px; width: 100%;"></div>		

## Summary

<b>references Variable?</b>	No
<b>references Method?</b>	Yes
<b>references Form?</b>	No
<b>Allowable Child Elements</b>	Button Cell Spacer Checkbox DateField Group Header Hyperlink Image Label Password Field Radio Button Group Reference Field Rich Text Editor Row Space Select Field Separator Text Area Text Field Translation Upload File

The following table describes the properties of grid layout:

Property	Mandatory/Optional	Comment
<b>Name</b>	Mandatory	Name of the Grid Layout.
<b>Can Accept Drop</b>	Optional	<p>This property allows you to drop objects to layouts from a table, tree, or tile grid. The following example shows how to use this property.</p> <ol style="list-style-type: none"> <li>1. Create a layout and set the <b>Can Accept Drop</b> property to either <b>TRUE</b> or a permission method.</li> <li>2. Set On Drop method to a method that you previously created (that is, in the Methods tab, right-click the method node and select <b>Create Drop Method</b> from the menu, which provides the required parameter list).</li> <li>3. Implement this method to suit your needs.</li> </ol> <p><b>Note:</b> There is no default implementation for dropping objects into a layout.</p>
<b>*Cell Alignment</b>	Optional	Specifies the horizontal positioning of its children within the Grid Layout (left, center, right). You can override the cell alignment of individual's Child Element by setting it at the Child Element level. Default is undefined, which behaves as if it is left.
<b>Cell Border Style</b>	Optional	Click the drop-down menu and select the style of your cell border from the list.
<b>Cell Padding</b>	Optional	Defines the cell padding of the grid layout. The default value is 2.
<b>Column Width(s)</b>	Optional	<p>Specifies in the individual column widths, comma delimited, of the grid layout. A column width can be in integer which is in pixels or in percentage of the Grid Layout's width, or a combination of pixels and percentages. The following are valid combinations:</p> <ul style="list-style-type: none"> <li>• Numbers only indicates px: 100, 100, 100</li> <li>• Numbers with or without px means they are in pixels. With or without a space between the number and px is acceptable: 100px, 100 px, 100 px</li> <li>• Percentage sign (%), with or without a space between it and the percent value 50%, 50 %</li> <li>• Combinations of px and % values: 60, 50%, 60, 50%</li> </ul> <p>This property uses a simple text editor with designer validation.</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• The width of form items width cannot be set in a percentage.</li> <li>• Form elements under a grid layout accept * as a width value, meaning that the width of the field is set by the <b>Column Width(s)</b> property.</li> </ul>
<b>Dynamic Tooltip</b>	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Edge</b>	Optional	Select this checkbox to show an edge around a layout element. See the other Edge properties in this table to customize your edged layout.
<b>Edge: Content Offset</b>	Optional	This property denotes the amount the contained elements are offset to. The

		property defaults to <b>Edge: Size</b> if the value of the <b>Edge: Content Offset</b> is 0. Set this property to less than <b>Edge: Size</b> to allow the contained elements to overlap the edge and corner media.
<b>Edge: Custom Sides</b>	Optional	Use this property to list custom sides for the border, such as <code>left</code> , <code>top</code> , <code>bottom</code> , and <code>right</code> . Some examples include the following: <ul style="list-style-type: none"> <li>• <code>left, top, bottom</code></li> <li>• <code>top, bottom</code></li> </ul>
<b>Edge: Dynamic Image</b>	Optional	This property contains either a method or variable that sets the edge image dynamically (starts with a <code>/</code> ). If the dynamic image is null, a static image is used for the edge instead.  <b>Note:</b> If neither a dynamic, nor static image is specified, the default <code>[SKIN]/edge.gif</code> image is used.
<b>Edge: Fill Center</b>	Optional	Specify whether to show the image background (with the <code>_center</code> extension) in the centre section (that is, behind the decorated layout).
<b>Edge: Size</b>	Optional	Use this property to define the size in pixels for corners and edges.
<b>Edge: Static Image</b>	Optional	This property contains the base name of images for edges. Extensions for each corner or edge piece are added to the image URL before its file extension. For example, a default base name of <code>edge.gif</code> , the top-left corner images is <code>edge_TL.gif</code> .  The list of valid extensions is as follows: <ul style="list-style-type: none"> <li>• <code>_TL</code> (top-left)</li> <li>• <code>_TR</code> (top-right)</li> <li>• <code>_BL</code> (bottom-left)</li> <li>• <code>_BR</code> (bottom-right)</li> <li>• <code>_T</code> (top)</li> <li>• <code>_L</code> (left)</li> <li>• <code>_B</code> (bottom)</li> <li>• <code>_R</code> (right)</li> <li>• <code>_center</code> (centre)</li> </ul>
<b>Fixed Column Widths</b>	Optional	Select this property's checkbox to for the grid layout to use the specified column widths.
<b>Height</b>	Optional	Height used or consumed by the Grid Layout. The value is defined as the number of pixels (integer), or as a percentage of page height or of the parent element.
<b>Include in Tab Order</b>	Optional	This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values: <ul style="list-style-type: none"> <li>• When set to <code>&lt;Default&gt;</code>, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <code>Exclude</code>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <code>Include</code>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.
<b>Label Align</b>	Optional	Click the <b>Value</b> field and select from one of the following alignment options for your label: <ul style="list-style-type: none"> <li>• <code>Left</code></li> </ul>

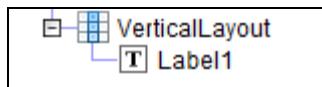
		<p><b>Centre</b>  • <b>Right</b></p> <p>This property can be overridden by the Grid Layout element's (for example, text field, radio button, and so on) <b>Label Align</b> property. If Label Align property is set to left or right, the label take its own column. This means that the grid layout appear as using two columns even if the <b>Number of Columns</b> property is set to 1.</p>
<b>Label Orientation</b>	Optional	Specifies where the label of Child Elements in the Grid Layout appear in relation to the Element. The available options are Top, Left and Right. Defaults to top.
<b>Members Margin</b>	Optional	Specifies the number of spaces allowed between grid layouts in the layout stack. See the <b>Notes</b> section on using two Grid layouts next to each other.
<b>Mouse Over Method</b>	Optional	Click this property's <b>Value</b> field and select a method to invoke when users hover their mouse over a specified element.
<b>Number of Columns</b>	Optional	Defines the number of columns within the Grid Layout. If this parameter is left undefined, the system creates a grid with 2 columns.
<b>On Enter</b>	Optional	The system calls any script defined in this property. The onEnter property is only met if there is an input field (for example, a textfield, selectfield, and so on) that has focus when the <b>Enter</b> key is pressed. This onEnter property is invoked only if the input field does not have an OnEnter property set. This feature can be used with multiple input fields, but only one onEnter method to fire for all fields.
<b>Redraw on resize</b>	Optional	<p>By default, this property is unchecked. When selected, this property works when form layouts have either a width or height set at 100%. When you resize the browser window, the layouts in the current form are redrawn so that it fits the current window's size.</p> <p><b>Note:</b> For complex layouts, it is not recommended that you use this property, as the entire page is redrawn. As a result, redrawing the entire page may take a while. Depending on the complexity of the page and the slowness of your computer, you may see a delay in seeing the page being redrawn.</p>
<b>Show Error Icon</b>	Optional	By default, this property is checked, indicating that the error icon is visible. To hide the error icon, deselect this property.
<b>Show Error Style</b>	Optional	This property allows you to set a different style when the field has a validation error by selecting this property. See the <a href="#">example</a> that follows on how to use this property with the <b>Show Error Icon</b> property.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Grid Layout such as positioning and color. Default is NONE.
<b>Tooltip</b>	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers over the Grid Layout. Default is empty, which means that a tooltip has not been defined. During runtime, the tooltip may be difficult to find as it is not the only layout on the screen and may cover a large area.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the Grid Layout. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
<b>Width</b>	Optional	Width used or consumed by the Grid Layout. The value is defined as the number of pixels (integer) or as a percentage of page width or of the parent element. For example, 30% of the page width or 150 pixels.

## Notes:

- Most common use of Grid Layout is to construct a Form for a Document.
- Elements with labels, such as Text Field, span two cells within the Grid Layout. For example, the a Text Field form element spans two cells: label in one cell and the text field in another cell. With a Label Orientation of either Left or Right - the label will be in text box's adjacent cell and with a Top Label Orientation, the label will appear in the cell above the text box.
- If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.
- If you require drawing a border or box around a label or single form item element, wrap the element with Grid Layout instead of Horizontal Layout or Vertical Layout. The following are examples of drawing a border around a label:
  - It is recommended that you use a Grid Layout with the **Edge** property selected and a Label element under the layout:



- It is not recommended that you use a Vertical Layout with the **Edge** property selected and a Label element under the layout:



The reason for using the Grid Layout is that at runtime, the label and any form element that is not under Grid Layout is already wrapped with Grid Layout.

- To show an actual grid (that is, horizontal and vertical lines) between the elements of the grid, do the following:
  1. To add a Grid Layout beside another one, create a form with a Horizontal Layout.
  2. Add two Grid Layouts.
  3. For the first Grid Layout, set the **Members Margin** property

**Note:** The **Members Margin** is for the entire Grid Layout. If you add another grid layout beside it, the margin acts as the amount of space between the first grid layout and the one beside it.

**Example:** Define an error style and show the error inline

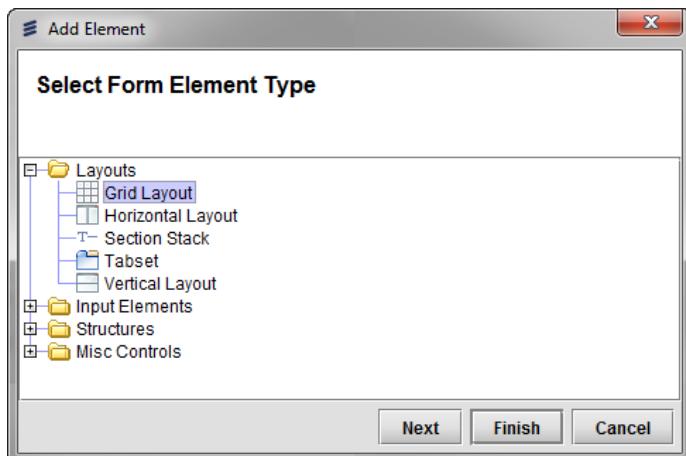
The following is an example of using the **Show Error Icon** and **Show Error Style** properties to customize a field containing a validation error:

1. Create a document with a few variables containing validation errors.
2. In the Default form, add a Grid Layout element with fields bound to the variables.
3. Set the **Show Error Icon** property to false and the **Show Error Style** to true.
4. In your metadata .css file, add the textItemError style, which overrides the default styling.
5. In runtime, invoke the validation error. The error icon does not appear and the styling of the fields have changed.
6. Clear the validation error. The style is back to normal.

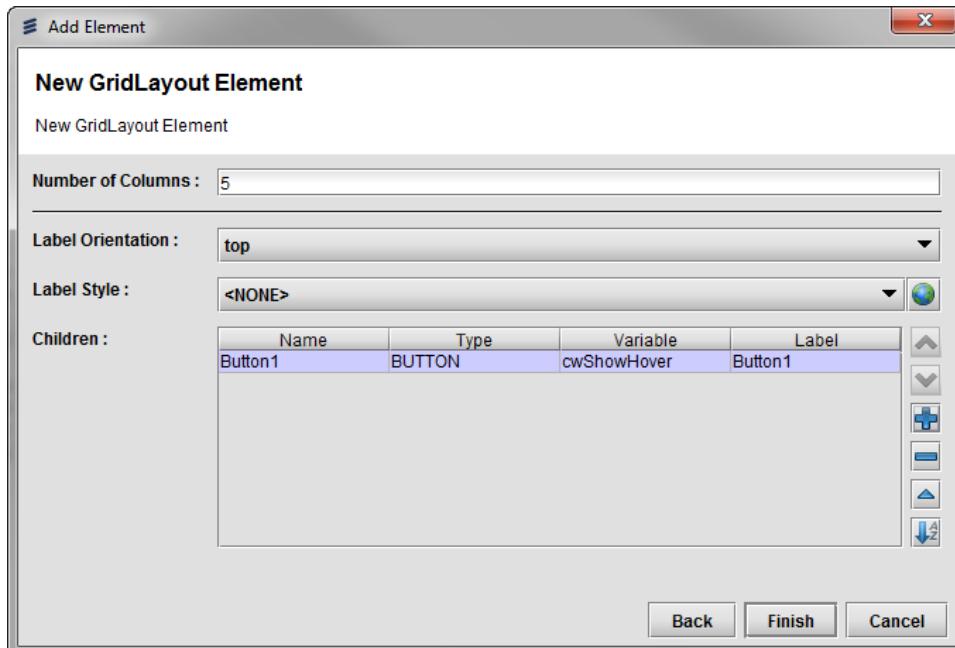
## Grid Layout Wizard

Velocity Studio contains a wizard for the Grid Layout element, allowing you to present child elements on a grid, specifying the number of columns, and more. To use the Grid Layout wizard, complete these steps:

1. When right-clicking to Add an element to a page or form, select **Grid Layout** from the **Layouts** folder, and then click the **Next** button.



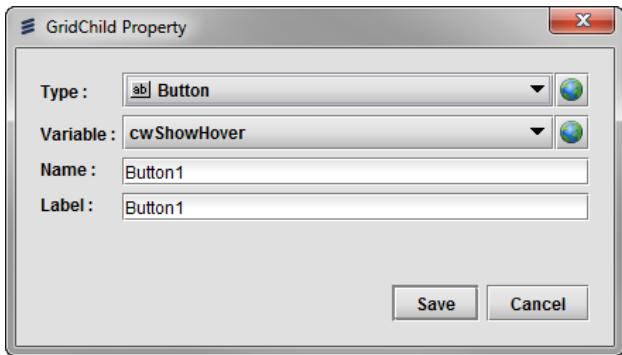
2. The New Grid Layout Element page displays.



Specify the following fields:

Property	Description
<b>Number of Columns</b>	This property defines the number of columns within the Grid Layout. If this parameter is left undefined, the system creates a grid with one column.
<b>Label Orientation</b>	Click the drop-down menu and specify where child element's label in the Grid Layout appear in relation to the element. The available options are Top, Left and Right. This property defaults to top.
<b>Label Style</b>	Click the drop-down menu and select from the styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Grid Layout, such as positioning and colour. The default is NONE.

3. For the **Children** field, click the **Add** button (+) to launch the Grid Child Property dialog.



Specify the following fields:

Button	Description
Type	Click the drop-down menu and select the child element type.
Variable	Click the drop-down menu and specify the variable from the list. The default value is NONE.
Name	Enter the name of your child element.
Label	Enter the label name that displays in the user interface.

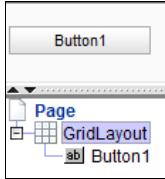
Click the **Save** button to continue.

4. If you have more child elements to add, click the **Add** button and repeat the process.

The New Grid Layout Element page contains the following buttons:

Button	Description
Up	Moves the selected child element up in the list.
Down	Moves the selected child element down in the list.
Add	Opens the Grid Child Property dialog so that you can add a new child element. The new child element will be placed below the selected row.
Remove	Removes the selected child elements from the list.
Properties	Opens a Grid Child Property dialog, allowing you to view or change the selected child element.
Sort	Sorts the list of child elements in ascending order.

5. Click the **Finish** button to create your Grid Layout and child elements.



## Group

Group is a container that allows for the grouping of data or elements on a page in a logical fashion such that the child data elements can be collapsed or expanded at runtime. The Group element works in the same fashion as a [Section Stack](#) in that the section can be expanded and collapsed. Unlike the [Section Stack](#) element, the Group element can be parented by the [Grid Layout](#) element to take the advance of the Grid Layout properties such as alignment and column spans. The Group element contains a header and the "real estate" containing the child elements can be expanded or collapsed.

The example below shows the relationship between a Group element within a Grid Layout.

**Element Tree and Properties**

Name	Value
Cell Alignment	
Expand	<input checked="" type="checkbox"/>
Header Height	
Header Style	
Label	Internet Connection Options
Name	Group
Tooltip	
Visible	
Width	350px

**Preview Pane**

Internet Connection Options	
Cable connection	\$25/mntrh
DSL	\$40/mntrh
Analog	\$15/mntrh
ISDN	\$20/mntrh
T1 Line	\$140/mntrh

## Summary

references Variable?	No
references Method?	No
references Form?	No
Allowable Child Elements	Button Checkbox Cell Spacer Date Field Date Time Header Hyperlink Image Label Password Field Radio Button Group Row Spacer Select Field Separator Slider Element Spinner Element Text Area Text Field Upload File Variable

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Group.
Label	Optional	Visible label of the header. Displayed over the header banner in a left-justified alignment. Default is empty.

<b>Cell Alignment</b>	Optional	Controls the horizontal and vertical positioning of the Group within the Grid Layout . Default is undefined, which behaves as if it is top-left.
<b>Expand</b>	Optional	This property determines whether the child field elements of the Group appear by default in a collapsed or expanded state at runtime. This property behaves similarly the Expand property of the Section Header element.
<b>Header Height</b>	Optional	Height used or consumed by the Header of the Group. The value is defined as the number of pixels (integer).
<b>Header Style</b>	Optional	Choose from styles within the selected style sheet (located in the Velocity Studio toolbar), which defines the styling of the header such color. The selected style must contain three associated style names that correspond with the collapsed and expanded state of the Group element. These styles must be present in the cascading style sheets (css) with consistent naming convention: <i>styleName</i> , <i>styleNameopened</i> and <i>styleNameclosed</i> . Select the style <i>styleName</i> for this property. When the state of the Group is collapsed - the style that will be used is <i>styleNameclosed</i> and when expanded it is <i>styleNameopened</i> .
<b>Tooltip</b>	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers on top of the Group. Default is empty, which means that a tooltip has not been defined. During runtime, the tooltip may be difficult to find as it is not the only layout on the screen and may cover a large area.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the group. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Visible</b>	Optional	Defines the user permissions assigned to this element. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
<b>Width</b>	Optional	Width used or consumed by the Group. This property takes an integer value and is defined as the number of pixels (px).

**Note:**

- You cannot have a Group element under a Table element.

## Header

---

The Header element inserts a header within a layout. The font, size and color of the Header can be changed as shown in the Vertical layout below with four different Header elements, each containing four different styles.



### Summary

references Variable?	No
references Method?	No
references Form?	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Header.
Label	Optional	Visible label of the Header. If empty, the label assumes the Name of the Header. Default is empty.
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Header in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Row Span	Optional	<p>This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
Dynamic Tooltip	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
Dynamic Tooltip Style	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
Error Icon Gap	Optional	<p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
Error Icon		This property allows you to configure how an icon are displayed. This property takes the

<b>Orientation</b>	Optional	values left (default) and right.
<b>Height</b>	Optional	Height to be taken up by the Header as defined as an integer value in pixels. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Show Validation</b>	Optional	This boolean property has values of either <i>True</i> or <i>False</i> , which determine if a validation warning message displays at runtime.  Setting this property to <i>True</i> : If you try to save a document at runtime with validation errors, the system displays a Warning Icon on the Header and validation messages appear when you hover over the Warning Icon.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the header such as size and color. Refer to the information on <a href="#">Style Sheets</a> for details on configuration. Default is none.
<b>Tooltip</b>	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers over the Header. Default is empty, which means that a tooltip has not been defined.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the header. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
<b>Width</b>	Optional	Width to be taken up by the Header; can be in integer which is in pixels, or in percentage (for example, "50%") of page width or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Notes

- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- Use Header in Vertical Layout to visually paragraph your Form. Create different styles in your stylesheet for Headers can achieve the appearance of multi-level Headers (for example, style H1, H2, H3, etc).
- If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Horizontal Layout

---

There are three layout directives for user interface design:

- Horizontal
- Vertical
- Grid

Horizontal Layout presents child form elements in a horizontal stack. Each element is presented adjacent to one another horizontally. The example below shows the customer information fields: Customer Name, Active and Member Since Date in a horizontal layout.

**Customer Info**

Customer Name	Active	Member Since Date
<input type="text"/>	<input checked="" type="radio"/> True <input type="radio"/> False	<input type="text"/> 

### Summary

<b>references Variable?</b>	No
<b>references Method?</b>	Yes
<b>references Form?</b>	No
<b>Allowable Child Elements</b>	Grid Layout Horizontal Layout Vertical Layout Layout Spacer Form Frame Button Checkbox Date Field Header Hyperlink Image Label Large Text Password Field Radio Button Group Reference Field Rich Text Editor Section Stack Select Field Separator Text Area Text Field Translation Upload File Variable HTML Content Iterator Menu Item Tabset Table Dynamic Table Dynamic Document

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Horizontal Layout.
Can Accept Drop	Optional	<p>This property allows you to drop objects to layouts from a table, tree, or tile grid. The following example shows how to use this property.</p> <ol style="list-style-type: none"> <li>1. Create a layout and set the <b>Can Accept Drop</b> property to either <b>TRUE</b> or a permission method.</li> <li>2. Set On Drop method to a method that you previously created (that is, in the Methods tab, right-click the method node and select <b>Create Drop Method</b> from the menu, which provides the required parameter list).</li> <li>3. Implement this method to suit your needs.</li> </ol> <p><b>Note:</b> There is no default implementation for dropping objects into a layout.</p>
*Cell Alignment	Optional	The cell alignment property functions in conjunction with the height/width properties. The values in the height/width properties must be set for the cell alignment property to function. This property specifies the horizontal and vertical positioning of the children form elements within the Horizontal Layout. An undefined value results in a "top-left" alignment.
Dynamic Tooltip	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
Dynamic Tooltip Style	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
Edge	Optional	Select this checkbox to show an edge around a layout element. See the other Edge properties in this table to customize your edged layout.
Edge: Content Offset	Optional	This property denotes the amount the contained elements are offset to. The property defaults to <b>Edge: Size</b> if the value of the <b>Edge: Content Offset</b> is 0. Set this property to less than <b>Edge: Size</b> to allow the contained elements to overlap the edge and corner media.
Edge: Custom Sides	Optional	<p>Use this property to list custom sides for the border, such as <code>left</code>, <code>top</code>, <code>bottom</code>, and <code>right</code>. Some examples include the following:</p> <ul style="list-style-type: none"> <li>• <code>left, top, bottom</code></li> <li>• <code>top, bottom</code></li> </ul>
Edge: Dynamic Image	Optional	<p>This property contains either a method or variable that sets the edge image dynamically (starts with a <code>/</code>). If the dynamic image is null, a static image is used for the edge instead.</p> <p><b>Note:</b> If neither a dynamic, nor static image is specified, the default <code>[SKIN]/edge.gif</code> image is used.</p>
Edge: Fill Center	Optional	Specify whether to show the image background (with the <code>_center</code> extension) in the centre section (that is, behind the decorated layout).
Edge: Size	Optional	Use this property to define the size in pixels for corners and edges.
Edge: Static Image	Optional	<p>This property contains the base name of images for edges. Extensions for each corner or edge piece are added to the image URL before its file extension. For example, a default base name of <code>edge.gif</code>, the top-left corner images is <code>edge_TL.gif</code>.</p> <p>The list of valid extensions is as follows:</p> <ul style="list-style-type: none"> <li>• <code>_TL</code> (top-left)</li> <li>• <code>_TR</code> (top-right)</li> <li>• <code>_BL</code> (bottom-left)</li> <li>• <code>_BR</code> (bottom-right)</li> <li>• <code>_T</code> (top)</li> <li>• <code>_L</code> (left)</li> </ul>

		<ul style="list-style-type: none"> <li>• <code>_B</code> (bottom)</li> <li>• <code>_R</code> (right)</li> <li>• <code>_center</code> (centre)</li> </ul>
<b>Fill Space</b>	Optional	<p>Set this property to true if the parent layout is needed to fill empty spaces with its children. For example, if the parent layout is 100% height and 100% width, and its children combined either use less than or more than 100%, the parent resizes its children to use 100%.</p> <p>This property should not be set to true if the parent layout requires a centered alignment. In general, when a parent has one child and the parent's alignment is set to center, the child should only be moved to the centre position. However, if the parent is instructed to <i>fill space</i>, its child is resized to fill the rest of the space that is unoccupied.</p>
<b>Height</b>	Optional	Height used or consumed by the Horizontal Layout. The value is defined as the number of pixels (integer) or as a percentage of page height or of the parent element. For example, 50% of the page height or 200 pixels.
<b>Include in Tab Order</b>	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>• When set to <code>&lt;Default&gt;</code>, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <code>Exclude</code>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <code>Include</code>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>Members Margin</b>	Optional	Specifies the number of spaces allowed between grid layouts in the layout stack.
<b>Mouse Over Method</b>	Optional	Click this property's <b>Value</b> field and select a method to invoke when users hover their mouse over a specified element.
<b>On Enter</b>	Optional	The system calls any script defined in this property. The <code>onEnter</code> property is only met if there is an input field (for example, a textfield, selectfield, and so on) that has focus when the <b>Enter</b> key is pressed. This <code>onEnter</code> property is invoked only if the input field does not have an <code>OnEnter</code> property set. This feature can be used with multiple input fields, but only one <code>onEnter</code> method to fire for all fields.
<b>Overflow</b>	Optional	<p>Defines how overflow child form elements are displayed based on the height/width of the layout. The default is Auto.</p> <p><b>Auto</b> - The child elements that do not fit within the height of the layout defined are shown via a scroll bar. The user can scroll across to see the children at runtime. In other words, the layout will scroll if the child members exceeds its specified size.</p> <p><b>Hidden</b> - The child elements that do not fit within the height of layout are NOT displayed at runtime.</p> <p><b>Visible</b> - If the child field element members size exceeds the specified layout size, the layout will grow to accommodate the child element members.</p>
<b>Redraw on resize</b>	Optional	<p>By default, this property is unchecked. When selected, this property works when form layouts have either a width or height set at 100%. When you resize the browser window, the layouts in the current form are redrawn so that it fits the current window's size.</p> <p><b>Note:</b> For complex layouts, it is not recommended that you use this property, as the entire page is redrawn. As a result, redrawing the entire page may take a while. Depending on the complexity of the page and the slowness of your computer, you may see a delay in seeing the page being redrawn.</p>
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Horizontal Layout such as positioning and color. Default is

		NONE.
<b>Tooltip</b>	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers over the Horizontal Layout. Default is empty, which means that a tooltip has not been defined. During runtime, the tooltip may be difficult to find as it is not the only layout on the screen and may cover a large area.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the Horizontal Layout. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions are assigned.
<b>Width</b>	Optional	Width used or consumed by the Horizontal Layout. The value is defined as the number of pixels (integer) or as a percentage of page width or of the parent element. For example, 30% of the page width or 150 pixels.

\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment.

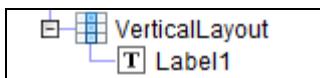
If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Notes:

- Horizontal Layout will set Child Elements that do not have explicit heights to match the layout.
- If you require drawing a border or box around a label or single form item element, wrap the element with Grid Layout instead of Horizontal Layout, Vertical Layout, Horizontal Stack Layout, or Vertical Stack Layout. The following are examples of drawing a border around a label:
  - It is recommended that you use a Grid Layout with the **Edge** property selected and a Label element under the layout:



- It is not recommended that you use a Vertical Layout with the **Edge** property selected and a Label element under the layout:



The reason for using the Grid Layout is that at runtime, the label and any form element that is not under Grid Layout (is already wrapped with Grid Layout).

- By default, horizontal layouts have a height of 1 px. If your dynamic forms, such as ones displayed through a form frame, are not appearing runtime, ensure that there is a height specified on at least the parent layout of that form frame, or on the form frame itself.
- To show your Web browser's scroll bar when your browser's window is too slow to view the entire application page, do the following:
  - In your main layout (that is, the top layout of the page form, such as the Horizontal Layout element), set the following properties:
    - Both the **Height** and **Width** properties to **100%**
    - The **Overflow** property to **Auto**
  - For all other child layouts, set the **Overflow** property to overflow to **DEFAULT**.
  - The scroll bar in your application works fine if the layout overflows or when you decrease your browser's size.

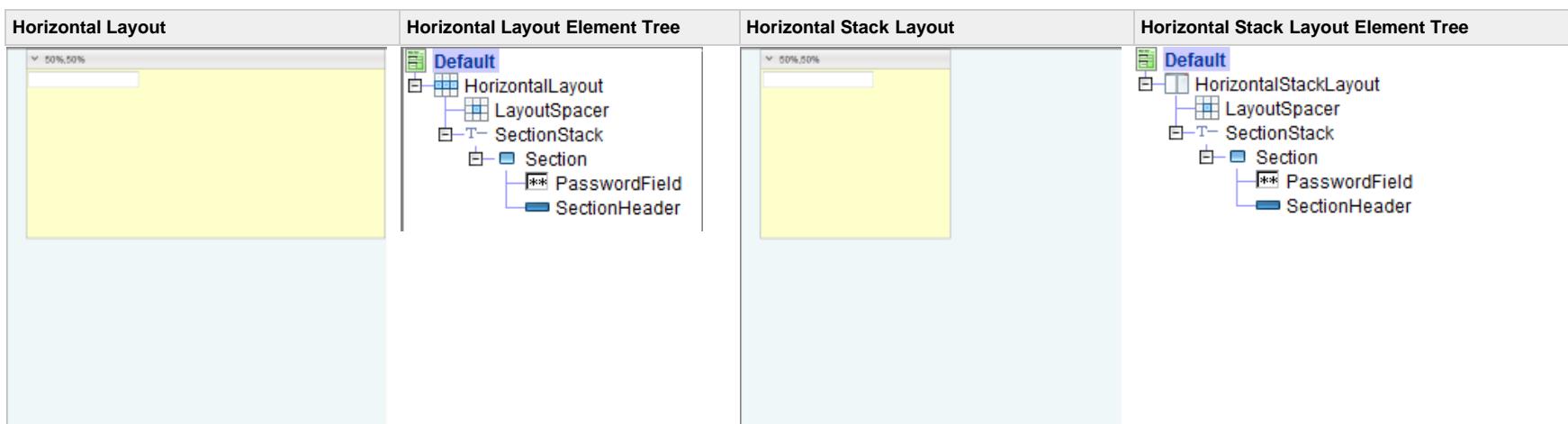


## Horizontal Stack Layout

**Note:** The Horizontal Stack layout is deprecated. Use the [Vertical](#) or [Horizontal](#) layout.

Horizontal Layout and a Horizontal Stack Layout present child form elements in a horizontal orientation. Each element is presented adjacent to one another horizontally. However, for the two horizontal layout directives, the width percentage property of their respective child elements results in a different presentation behaviour. The child form elements of the Horizontal Layout takes on the percentage of the available empty space. Whereas the Horizontal Stack Layout's child elements will present the child elements as a percentage of the parent element.

Below is an example of both Layouts and child elements. This example show the area that the parent element (Horizontal Layout and Horizontal Stack Layout) in blue and the child Section Stack element in the yellow area. These two examples shows the Horizontal Layout directives with a child SectionStack element and a LayoutSpacer. The SectionStack element has a width percentage property value of 50% and a LayoutSpacer width property value of 25px in both layout directives. For the Horizontal Layout, the SectionStack element (area displayed in yellow) takes on the width of the remaining space of its parent (Horizontal Layout width - the LayoutSpacer width). Whereas for the Horizontal Stack Layout, the child SectionStack width is 50% of the Horizontal Stack Layout width regardless of the LayoutSpacer element's width.



The table below shows the property values for the above example:

Parent Layout Properties		Child Layout Spacer Properties		Child Section Stack Properties	
		Name	Value		
<b>Default</b>	<b>HorizontalStackLayout</b>	Default		<b>Default</b>	
HorizontalStackLayout		Height		HorizontalStackLayout	
LayoutSpacer		Name	LayoutSpacer	LayoutSpacer	
SectionStack		Style		SectionStack	
Section		Visible		Section	
PasswordField		Width	25px	PasswordField	
SectionHeader				SectionHeader	
On Enter				Header Style	
Overflow				Height	50%
Show Resize Bar	<input type="checkbox"/>			Name	SectionStack
Style	CwBanner			Visibility Mode	mutex
Tooltip				Visible	
Visible				Width	50%
Width	500px				

Although both layout properties and child element properties have exactly the same values, the way in which the child elements are displayed differ. The difference is in the handling of the child sizing

percentage. The Horizontal Layout will adjust its children to fill the available space, whereas the stack layouts will always respect the width/height values of the children.

## Summary

<b>references Variable?</b>	No
<b>references Method?</b>	Yes
<b>references Form?</b>	No
<b>Allowable Child Elements</b>	Button Checkbox Date Field Date Time Dynamic Table Dynamic Document Form Frame Grid Layout Header Horizontal Layout Horizontal Stack Layout HTML Content Hyperlink Image Iterator Label Layout Spacer Menu Item Password Field Radio Button Group Section Stack Select Field Separator Slider Spinner Tabset Text Area Text Field Tree Upload File Variable Vertical Layout Vertical Stack Layout

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Horizontal Stack Layout.
*Cell Alignment	Optional	The cell alignment property functions in conjunction with the height/width properties. The values in the height/width properties must be set for the cell alignment property to function. This property specifies the horizontal and vertical positioning of the children form elements within the Horizontal StackLayout. An undefined value results in a "top-left" alignment.
Dynamic Style	Optional	The value of this property is a method with a Return type <i>String</i> . This method will dynamically change the element's style at runtime based on the instructions returned by the method. The style returned by the method must exist within the current cascading style sheet (css) file that is being used by the application.
Dynamic Tooltip	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
Dynamic		

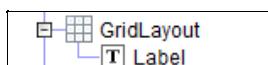
<b>Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Edge</b>	Optional	Select this checkbox to show an edge around a layout element. See the other Edge properties in this table to customize your edged layout.
<b>Edge: Content Offset</b>	Optional	This property denotes the amount the contained elements are offset to. The property defaults to <b>Edge: Size</b> if the value of the <b>Edge: Content Offset</b> is 0. Set this property to less than <b>Edge: Size</b> to allow the contained elements to overlap the edge and corner media.
<b>Edge: Custom Sides</b>	Optional	Use this property to list custom sides for the border, such as <code>left</code> , <code>top</code> , <code>bottom</code> , and <code>right</code> . Some examples include the following: <ul style="list-style-type: none"> <li>• <code>left, top, bottom</code></li> <li>• <code>top, bottom</code></li> </ul>
<b>Edge: Dynamic Image</b>	Optional	This property contains either a method or variable that sets the edge image dynamically (starts with a <code>/</code> ). If the dynamic image is null, a static image is used for the edge instead. <b>Note:</b> If neither a dynamic, nor static image is specified, the default <code>[SKIN]/edge.gif</code> image is used.
<b>Edge: Fill Center</b>	Optional	Specify whether to show the image background (with the <code>_center</code> extension) in the centre section (that is, behind the decorated layout).
<b>Edge: Size</b>	Optional	Use this property to define the size in pixels for corners and edges.
<b>Edge: Static Image</b>	Optional	This property contains the base name of images for edges. Extensions for each corner or edge piece are added to the image URL before its file extension. For example, a default base name of <code>edge.gif</code> , the top-left corner images is <code>edge_TL.gif</code> .  The list of valid extensions is as follows: <ul style="list-style-type: none"> <li>• <code>_TL</code> (top-left)</li> <li>• <code>_TR</code> (top-right)</li> <li>• <code>_BL</code> (bottom-left)</li> <li>• <code>_BR</code> (bottom-right)</li> <li>• <code>_T</code> (top)</li> <li>• <code>_L</code> (left)</li> <li>• <code>_B</code> (bottom)</li> <li>• <code>_R</code> (right)</li> <li>• <code>_center</code> (centre)</li> </ul>
<b>Height</b>	Optional	Height used or consumed by the Horizontal Stack Layout. The value is defined as the number of pixels (integer) or as a percentage of page height or of the parent element. For example, 50% of the page height or 200 pixels.
<b>Include in Tab Order</b>	Optional	This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values: <ul style="list-style-type: none"> <li>• When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.
<b>Members Margin</b>	Optional	Specifies the number of spaces allowed between grid layouts in the layout stack.
<b>Mouse Over Method</b>	Optional	Click this property's <b>Value</b> field and select a method to invoke when users hover their mouse over a specified element.
<b>On Enter</b>	Optional	The system calls any script defined in this property. The <code>onEnter</code> property is only met if there is an input field (for example, a textfield, selectfield, and so on) that has focus when the <b>Enter</b> key is pressed. This <code>onEnter</code> property is invoked only if the input field does not have an <code>OnEnter</code> property set. This feature can be used with multiple input fields, but only one <code>onEnter</code> method to fire for all fields.
		Defines how overflow child form elements are displayed based on the height/width of the layout. The default is Auto.

<b>Overflow</b>	Optional	<p><b>Auto</b> - The child elements that do not fit within the height of the layout defined are shown via a scroll bar. The user can scroll across to see the children at runtime. In other words, the layout will scroll if the child members exceeds its specified size.</p> <p><b>Hidden</b> - The child elements that do not fit within the height of layout are NOT displayed at runtime.</p> <p><b>Visible</b> - If the child field element members size exceeds the specified layout size, the layout will grow to accommodate the child element members.</p>
<b>Show Resize Bar</b>	Optional	If true, a resize bar is created for the horizontal stack layout, that allows the user to resize the horizontal layout by dragging it, and show/hide layout by clicking it. The orientation of the resize bar for the horizontal layout yields an up-and-down resize bar. The length of the resize bar is dictated by the width of the parent. Specifically, it is the height if either the vertical layout parent or the width of the horizontal layout parent that determines the resize bar's length. Default is <i>default</i> , which does not show the resize bar.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the layout directive such as positioning and colour. Default is NONE.
<b>Tooltip</b>	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers over the Horizontal Stack Layout. Default is empty, which means that a tooltip has not been defined. During runtime, the tooltip may be difficult to find as it is not the only layout on the screen and may cover a large area.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the Horizontal Stack Layout. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions are assigned.
<b>Width</b>	Optional	Width used or consumed by the Horizontal Stack Layout. The value is defined as the number of pixels (integer) or as a percentage of page width or of the parent element. For example, 30% of the page width or 150 pixels.

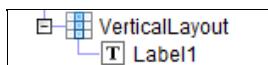
\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout (for example, vlayout or hlayout with the width and height set) this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

#### Notes:

- If you require drawing a border or box around a label or single form item element, wrap the element with Grid Layout instead of Horizontal Layout, Vertical Layout, Horizontal Stack Layout, or Vertical Stack Layout. The following are examples of drawing a border around a label:
  - It is recommended that you use a Grid Layout with the **Edge** property selected and a Label element under the layout:



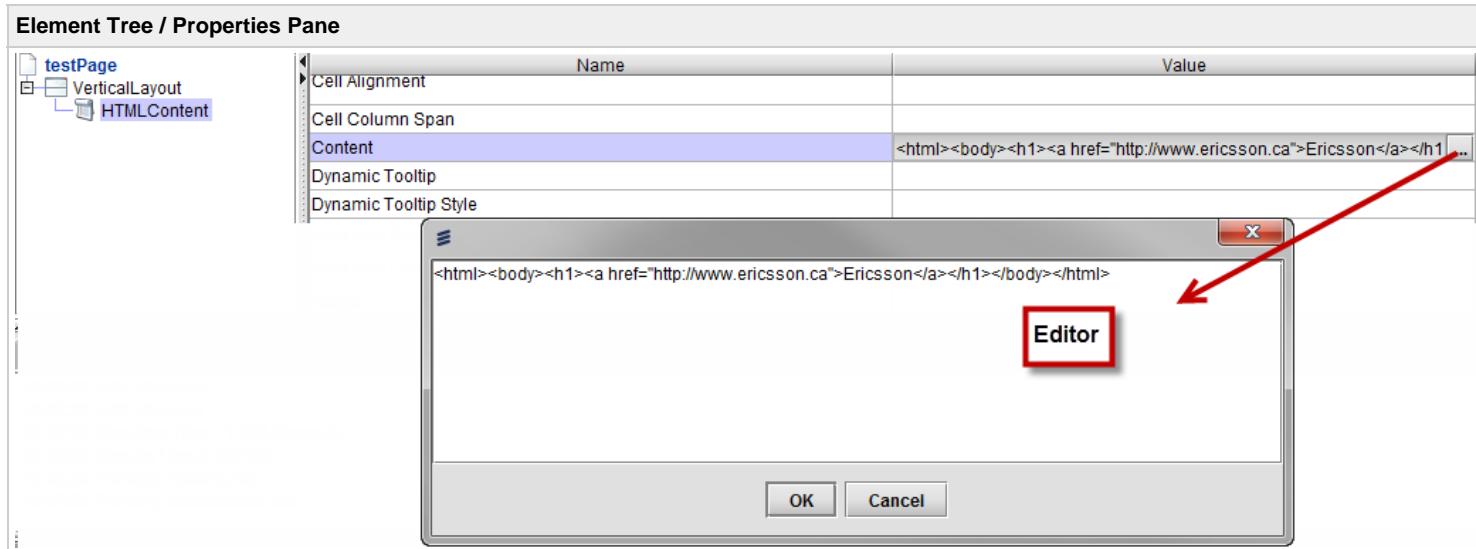
- It is not recommended that you use a Vertical Layout with the **Edge** property selected and a Label element under the layout:



The reason for using the Grid Layout is that at runtime, the label and any form element that is not under Grid Layout (is already wrapped with Grid Layout).

## HTML Content

The HTML Content element allows you to embed HTML code/content into Velocity Studio. The example below shows the HTML Content element and the Content property that has been modified with HTML code. The user can click the Content property to open an editor to enter code which can be viewed at runtime or previewed in Velocity Studio's Preview Pane. The example below shows html hyperlink content that triggers the Web site.



The Preview Pane below displays the Web site as a result of the HTML hyperlink code.

The screenshot shows the Preview Pane displaying the Ericsson website. The header includes the Ericsson logo, the tagline 'Empowering Orders Proven, high performance order and catalog management solutions', a search bar with placeholder 'Enter keyword', and a navigation menu with links for HOME, DOWNLOADS, CAREERS, and CONTACT. Below the header is a blue navigation bar with links for COMPANY, DRIVING YOUR SUCCESS, PRODUCTS, SOLUTIONS, SERVICES, PARTNERSHIPS, and NEWS & EVENTS.

### Summary

references Variable	Yes
references Method	Yes, Variable.
references Form	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the HTML content.
Label	Mandatory	Visible label of the HTML content. Default is empty.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the HTML Content in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the HTML Content occupies in the Grid Layout, to the right of current cell. Default is empty, which means it occupies one column.

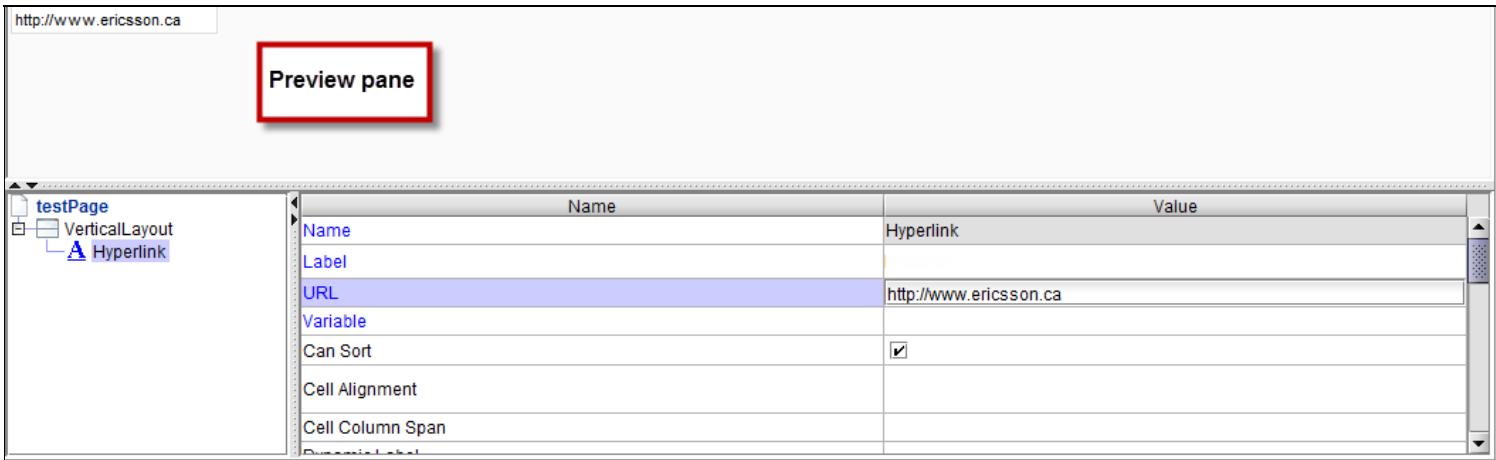
<b>Content</b>	Optional	The Content property opens up an editor that allows the user to enter HTML text.
<b>Dynamic Tooltip</b>	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Error Icon Gap</b>	Optional	Select this property's checkbox to make a gap for the validation icon. By default, this property is false. <b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwf/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Height</b>	Optional	Height to be taken up by the Button; can be an integer defined in pixels.
<b>Icon Height</b>	Optional	Height of the icon used or consumed within the element. The value is defined as the number of pixels (integer) or percentage. The default value is 0.
<b>Icon Tooltip</b>	Optional	Click the ellipsis button and specify a tooltip description within the <b>Value</b> field for the icon.
<b>Icon Width</b>	Optional	Width of the icon used or consumed within the element. The value is defined as the number of pixels (integer) or percentage. The default value is 0.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Start Row</b>	Optional	Applicable only when the parent element is a Grid Layout. If true, the HTML Content appears in a new row of the grid. If False, it is simply placed in the next available cell. Default is False, which means that the HTML Content element does not appear on a new row.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the button such as positioning and color. Refer to the information on <a href="#">Style Sheets</a> for details on configuration. Default is none.
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the HTML content. Default is empty, which means no tooltip for the HTML content.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the HTML content. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Variable</b>	Optional	The variable must be of the type <i>string</i> or methods of the return type <i>string</i> . Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Determines whether the HTML Content is visible or not. Choose true (visible) or false (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this field. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE or True, which means no permissions assigned and field is visible..
<b>Width</b>	Optional	Width to be taken up by the HTML Content; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element. If this field is empty, the Width of the field should be taken up by that defined in the Column Width property of the Grid Layout.

## Notes

\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Hyperlink

This element creates hyperlink text. The example shows the Hyperlink element's preview pane and properties for a Web page hyperlink.



### Summary

references Variable	Yes
references Method	Yes
references Form	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Mandatory. Name of the Hyperlink.
Label	Mandatory	Visible label of the Hyperlink. If empty, the label assumes the URL of the Hyperlink. Default is empty.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Hyperlink element in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Hyperlink shall occupy in the Grid Layout, to the right of current cell. Default is empty, which occupy one column.
Cell Row Span	Optional	<p>This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
Dynamic Label	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
Dynamic	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by

<b>Label Style</b>		the method. The label style returned by the method must exist within the current css file that is being used by the application.
<b>Dynamic Tooltip</b>	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Editable</b>	Optional	<p>This property determines whether the Hyperlink is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be enabled or disabled).</p> <p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's <i>editablePerm</i> permission takes over. If this method is not overwritten, the model's <i>editablePerm</i> permission is used. User interfaces displayed under another user interface with <i>editablePerm</i> are also be inherited if the embedded user interfaces do not have their own <i>editablePerm</i> permission.</p>
<b>Error Icon Gap</b>	Optional	<p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the <i>/cwfv/error.png</i> icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Height</b>	Optional	Height to be taken up by the Hyperlink as defined as an integer value in pixels. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>Include in Tab Order</b>	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>• When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>Label Align</b>	Optional	<p>Click the <b>Value</b> field and select from one of the following alignment options for your label:</p> <ul style="list-style-type: none"> <li>• <b>Left</b></li> <li>• <b>Centre</b></li> <li>• <b>Right</b></li> </ul>
<b>Label Orientation</b>	Optional	Specifies where the label of the Hyperlink appears in relation to the Hyperlink area; can be <i>left</i> , <i>right</i> or <i>top</i> of the Hyperlink area. The default orientation is <i>right</i> . However, in a Grid Layout , the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout.
<b>On Enter</b>	Optional	The system calls any script defined in this property. This method is invoked when the Hyperlink is focused on the Web page and the user presses the ENTER key.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Show Label</b>	Optional	This property determines whether the label is displayed on the Hyperlink. Choose from <i>True</i> or <i>False</i> . Default is <i>True</i> which means that the label is displayed.
<b>Start Row</b>	Optional	Applicable only when the parent element is a Grid Layout. If true, the Hyperlink appears in a new row of the grid. If false, it is simply placed in the next available cell. Default is <i>False</i> , which means that the Hyperlink element does not appear on a new row.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Hyperlink such as positioning and color. Default is none.
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Hyperlink. Default is empty, which means no tooltip for the Hyperlink.

<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the hyperlink. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>URL</b>	Optional	Specify the URL of the hyperlink.
<b>Variable</b>	Optional	Choose from the list of elementary Variables available in the User Interface to bind to this Hyperlink. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
<b>Width</b>	Optional	Width to be taken up by the Hyperlink; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element. If this field is empty, the Width of the field should be taken up by that defined in the Column Width property of the Grid Layout. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Notes

- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Image

Inserts an image into the user interface. This image can change at runtime based on certain user actions such as click down or mouse over. When the user clicks on the image, the user can be directed to a URL or the system will perform instructions defined in a Method.

## Summary

references Variable	Yes
references Method	Yes Click Method, Permissions.
references Form	No
Allowable Child Elements	None

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Image.
Label	Mandatory	Visible label of the Image. Default is empty which does not display a label.
Action Auditor	Optional	Specify the Action Validator object for this button's <b>Action Auditor</b> property. During runtime, when you click this button, the associated logging action is also invoked, provided that the <b>Enable action logging</b> checkbox from the System <a href="#">Logging</a> tab has been selected.
Can Focus	Optional	This property determines whether an image can have focus ( <i>true</i> ) or not have focus ( <i>false</i> ). For example, in the Show Help functionality, this property is used for keeping focus on a previously clicked element and then clicking the help image to generate help text associated with the focused element.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Image element in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the image will occupy in the Grid Layout, to the right of current cell. By default, the image takes the available columns in a Grid Layout, unless specified the column span property.  For example, if the Grid Layout has specified the number of columns as 3 and the images is the first field under it, the image will take 3 columns. If the image is the second field and the first field only takes one column, the image will take 2 columns.
Cell Row	Optional	This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.  By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if

<b>Span</b>		the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.  To use this property, see the <a href="#">example</a> for details.
<b>Click Method</b>	Optional	Choose from the list of methods available in the User Interface to bind to this Image. The method is invoked when a user clicks the Image. Use the property's <a href="#">lookup button</a> to view property details.
<b>Disabled</b>	Optional	This property can be used by setting either a permission, or static TRUE or FALSE. When this property's value is TRUE, the image is disabled and the click method does not fire when attempting to click this image.
<b>Dynamic Label</b>	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.  When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.
<b>Dynamic Tooltip</b>	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Error Icon Gap</b>	Optional	Select this property's checkbox to make a gap for the validation icon. By default, this property is false.  <b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Height</b>	Optional	Height to be taken up by the Image as defined as an integer value in pixels.
<b>Image Type</b>	Optional	This property allows you to resize your image and takes one of the following values: <ul style="list-style-type: none"><li>• <b>Default</b> - Takes null as its value, by default.</li><li>• <b>Center</b> - Centres the image. The image is not stretched. If the original size of the image is smaller than the user-set size, the image is shown in the centre of the frame. Otherwise, if the original size is larger than the user-set size, the image appears as its normal size.</li><li>• <b>Stretch</b> - Stretches the image to the user-set width and height</li><li>• <b>Tile</b> - Repeats the image if the original size is smaller than the user-set size. If the original size of the image is greater than the user-set size, only part of the image is shown.</li><li>• <b>Normal</b> - Allows the image to have its original size. The image's height and width do not apply. By default, the image appears in the top left corner.</li></ul>
<b>Image URL</b>	Optional	Select an image graphic from the graphic resources found within the Resources folder. The image entered in this property is the standard image that appears at runtime. However, this image can change based on a user action like clicking the image or an image mouse roll over by using the Show Click Image and the Show Roll Over Image properties. The system uses the following image files to define the image used in the Standard, Click and Roll Over states respectively: <ImageName>, <ImageName>_Down and <ImageName>_Over.  Select the <ImageName> image for this property.
<b>Include in Tab Order</b>	Optional	This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values: <ul style="list-style-type: none"><li>• When set to &lt;Default&gt;, the element is included in tab order according to global rules and its focusable state.</li><li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li><li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li></ul> <b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.
		Specifies where the label of the Image appears in relation to the Image area; can be <i>left</i> , <i>right</i> or <i>top</i> of

<b>Label Orientation</b>	Optional	the Image area. The default orientation is <i>right</i> . However, in a Grid Layout , the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout.
<b>Label Style</b>	Optional	Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Label within the Tab Frame such as color and font. To make changes to the styles and stylesheet refer to the <a href="#">Navigation Pane - Resources</a> for further details. Default is NONE.
<b>Link URL</b>	Optional	Specify a URL (for example, http://www.google.com). If this property is given a value, then the URL will be triggered at runtime when the user clicks the Image Label.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Show Click Image</b>	Optional	Boolean to indicate whether to show the click image when the image is clicked. Both the standard image and the Click Image must be available in the Resources folder (<ImageName> and <ImageName>_Down and the standard image must be entered in the Image URL property.
<b>Show Label</b>	Optional	This property determines whether the label is displayed on the Image. Choose from <i>True</i> or <i>False</i> . Default is <i>True</i> which means that the label is displayed.
<b>Show Roll Over Image</b>	Optional	Boolean to indicate whether to show the rollover image when cursor hovers above the image. Both the standard image and the Roll Over Image must be available in the Resources folder (<ImageName> and <ImageName>_Over and the standard image must be entered in the Image URL property.
<b>Start Row</b>	Optional	Applicable only when the parent element is a Grid Layout. If true, the Image appears in a new row of the grid. If false, it is simply placed in the next available cell. Default is <i>False</i> , which means that the Image element does not appear on a new row.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Image such as positioning. To make changes to the styles and stylesheet refer to the <a href="#">Navigation Pane - Resources</a> for further details. Default is NONE.
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Image. Default is empty, which means no tooltip for the Image.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the image. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Variable</b>	Optional	Choose from Variables of the type <i>string</i> or methods that return a type <i>string</i> available in the User Interface to bind to this Image. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
<b>Width</b>	Optional	Width to be taken up by the Image; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element. If this field is empty, the Width of the field should be taken up by that defined in the Column Width property of the Grid Layout.

## Notes

Image click may be enabled in two different ways. One way is to set a **Click Method** property. The other is to set a **Link URL** property. Click Method takes precedence if both are provided.

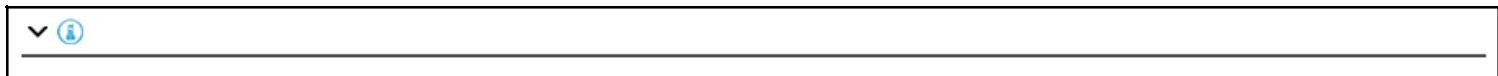
\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Iterator

An Iterator element enables you to create customized User Interface objects at runtime by iterating through Form elements contained in a User Interface array. The Iterator element can create several instances of a User Interface based on the User Interface array. The array consists of one or more element that is initialized using method scripts. By using the Iterator element you can avoid creating separate User Interfaces for similar items that you want displayed at runtime, and instead, create one User Interface that is instantiated multiple times.

### Example

The following example shows the benefit of including the Iterator element in a Form. You can configure a section by adding a section elements to a Form. The section can separate information into grouping making it easier to present information to the end user.



Using the Iterator element, you can configure the User Interface as an array and use variables and methods to *repeat* this element for different package choices including Voice, TV, Internet, Install Options.

A screenshot of a Windows Internet Explorer browser window displaying a web-based form. The form includes sections for 'Customer Information' and 'Billing Address'. On the left side, there are four groups of input fields, each preceded by a collapse/expand icon (triangle). These groups are circled in red. The first group contains fields for 'Voice' (including a dropdown for 'Title' with 'Mrs.' selected, and 'First Name' and 'Last Name' fields), 'Number', 'Street Name', 'Unit', 'City', 'State', and 'ZIP Code'. The second group contains fields for 'TV', 'Customer Type', 'Language Preference', 'Home Phone', 'Cell Phone', 'Drivers Licence', 'State', 'Gender', 'Email', and 'Comments'. The third group contains fields for 'Internet'. The fourth group contains fields for 'Install Options'. Below these groups, there is a section for 'Digital Economy' with 'TV' and 'Install Options' sections, and a section for 'HSD Performance' with 'Install Options'. To the right of the main form area, a 'Shopping Cart' sidebar lists various items with their prices and quantities.

### Summary

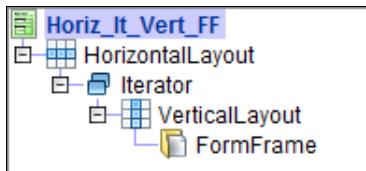
<b>references Variable</b>	Yes
<b>references Method</b>	No
<b>references Form</b>	No
<b>Allowable Child Elements</b>	Reference Form depends upon the parent Form used for the Iterator

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Iterator.
Variable	Optional	Choose from the list of variables in the User Interface (of array type) to bind to the Iterator. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
Visible	Optional	Determines whether the Section Stack is visible or not. Defaults to NONE, which means visible.

## Configuring the Iterator Element

The Iterator element moves through (iterates) elements that are located beneath it in the Form Element Tree. For example, you can configure the Form Element Tree to display a HorizontalLayout element, followed by the Iterator and a Menu Form element. The Menu item defines a click method. That method contains a script (array) that specifies the sequencing iteration of the menu.



Using an Iterator you can cycle through Form elements in sequence, one at a time. The Iterator is a *child* to the following Form elements:

Form Element	Description	Allowable Child Form Elements
Section Stack	When an Iterator is added beneath a <a href="#">Section Stack</a> , you must add a Section Form element beneath the Iterator. The Iterator advances through Form elements that are added beneath the Section Form. Using a next call, the Iterator returns elements that belong to the section.	Section
Tabset	When an Iterator is added beneath a <a href="#">Tabset</a> , you must add a Tab Frame Form element beneath the Iterator. The Iterator advances through Tab Frame elements that are added beneath the Iterator.	Tab Frame
HorizontalLayout	You can also add an Iterator after the <a href="#">HorizontalLayout</a> Form elements. You can then add Form elements beneath the Iterator.	HorizontalLayout, Image, Menu Item, VerticalLayout
VerticalLayout	You can also add an Iterator after the <a href="#">VerticalLayout</a> Form elements. You can then add Form elements beneath the Iterator.	HorizontalLayout, Image, Menu Item, VerticalLayout

**Note:** When configuring element properties, if you have a Form Frame element under the Iterator, the *Form* property of the Form Frame is defined as one of the Iterator variable in the User Interface.

## User Interface Variables

The screenshot shows two panels. The top panel is a table titled 'Methods' with columns for Name, Type, and various permissions. It lists three items: 'confirmObject' (Type Object), 'model' (Type Model), and 'iterated' (Type User Interface, highlighted with a red oval). The bottom panel is a configuration dialog for the 'iterated' variable. It has tabs for General and Methods. Under General, the 'Name' is 'iterated', 'Data type' is 'Iterator.Iterated' (highlighted with a red oval), and 'Type' is 'Array' (also highlighted with a red oval). Other options like 'Constant' and 'Audit' are unchecked. The 'Element properties' dropdown is set to '<Inherited>'. Below this is a table for element properties.

To configure the User Interface variables, first add a variable as type User Interface, then do the following:

- Define the variable's Data type (this refers to the parent + User Interface)
- Assign the User Interface as an array (an array of type User Interface)

## User Interface Methods

To initialize an Iterator, you can configure a method. For example, the following script defines how a section form element will be iterated.

```
function OnInit() { //Metadata type method. Can be called by scripts.

var uis = new Array();

for (var i = 0; i < 10; i++) {
    uis[i] = new Iterator.Iterated;
    uis[i].Form = "Default";
    uis[i].iteratorForm = "Default";
    uis[i].label = "Iterated " + i;
}
this.iterated = uis;

}
```

### Notes:

- In addition to using one Iterator element in a User Interface Form, you can also nest Iterator elements.
- The Iterator Form element is also contained in the HLayoutForm and VLayoutForm of the Navigation Bar (these are preformatted Form elements used only with the Navigation Bar).
- You can add Horizontal, Vertical, and Grid Layouts under an Iterator element.

# Label

---

The Label Element creates a Label that can be configured to display its label properties.

## Display

The Label Element has a number of visual properties, that when applied, changes how a label is rendered in the interface ([see properties](#)). For example you can set the font property of a label (family, color, size and detail) by configuring the Font Properties property.

Referencing Method Scripts in specific Label Element properties such as; Variable is another way of changing the visual properties of a label.

The Dynamic Style property of the Label Element enables the dynamic change of a label in the interface. Use a Method Script (must be defined as a string return type) to define the parameters that will execute in the user interface. For example, you can configure a Label Element to display in red, providing the end-user with appropriate feedback when they perform an action in the interface (typing in a field that requires follow up). The Method Script used with the Label Element specifies the function of the Label Element (turning red) when triggered by the user action.



## Summary

references Variable	Yes
references Method	Yes
references Form	No
Allowable Child Elements	None

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Label.
Label	Mandatory	Visible label of the Label. If empty, there is no Label label. Default is empty.
Click Method	Mandatory	Choose from the list of methods available in the User Interface to bind to this label. The method is invoked when you click the label, which acts as a hyperlink. Use the property's <a href="#">lookup button</a> to view property details.
Variable	Mandatory	Dynamically set this element's label. Use the property's <a href="#">lookup button</a> to view property details.
Can Sort	Optional	Specifies whether the Label can be sorted (true) or unsorted (false). The default is set to true. This option is available when the Label appears in a table.
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Label in the page or parent Element. Default is undefined, which behaves as if it is top-left
Cell Row	Optional	This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.

<b>Span</b>		By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.  To use this property, see the <a href="#">example</a> for details.
<b>Data Length</b>	Optional	Specifies the maximum number of characters that can be entered into the textbox. Default is 0, which means no character restrictions.
<b>Dynamic Label Style</b>	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
<b>Dynamic Tooltip</b>	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.  When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Error Icon Gap</b>	Optional	Select this property's checkbox to make a gap for the validation icon. By default, this property is false.  <b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwf/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Font Properties</b>	Optional	Specifies the font properties of the Label. Click the ellipsis to configure the font-family, font-color and font-size. This overrides the styling assigned, if any, to the Label.
<b>Group by In Table</b>	Optional	Displays the Label in Finder results. The Label is displayed in a column with elements that are grouped with the same value.
<b>HTML Escape</b>	Optional	Specifies if HTML code is used to configure the label. This property is only valid when parented by a Tree or Table element. Default is true.
<b>Height</b>	Optional	Height to be taken up by the Label; can be in integer which is in pixels, or in percentage (for example, "50%") of page height or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>Icon</b>	Optional	If an icon is selected, the icon appears in the Label, besides the label. Default is NONE.
<b>Label Orientation</b>	Optional	Specifies the alignment of the Label; it can be <b>top</b> , <b>left</b> , or <b>right</b> . Default is NONE.
<b>On Enter</b>	Optional	When Label is selected and the user presses ENTER, the action of the selected Method is invoked.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
		Applicable only when parent Element is a Grid Layout. If true, the Label starts in a new row of the grid. If false, it is simply placed in the current cell as-is. Default is <i>/default</i> , which does not start a new row.  The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:

<b>Run Trigger</b>	Optional	<ul style="list-style-type: none"> <li>You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>You click a menu.</li> <li>You change your selection in a navigation tree.</li> <li>You click another tab of a tabset.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>Field triggers do not fire on save. However, document triggers do.</li> <li>As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
<b>Show Label</b>	Optional	Displays the Label. If true, the Label displays. If false, Label does not display.
<b>Start Row</b>	Optional	Applicable only when parent Element is a Grid Layout. If true, the Label starts in a new row of the grid. If false, it is simply placed in the current cell as-is. Default is /default, which does not start a new row.
<b>Style</b>	Optional	Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the Label such as color and font. Default is NONE.
<b>Text Align</b>	Optional	Specifies the alignment of the label; it can be <b>left</b> , <b>center</b> , or <b>right</b> . Default is empty, which is left-aligned.
<b>Textbox Style</b>	Optional	<p>Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the textbox of the Label such as color and font. Default is NONE.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the MyStyle style to the <b>Textbox Style</b> property, the stylesheet must have MyStyleDisabled to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Label. Default is empty, which means no tooltip for the Label.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the label. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Variable</b>	Optional	Choose from the list of variables that you can apply to the label. Default is NONE. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Determines whether the Label is visible or not. Defaults to NONE, which means visible.
<b>Width</b>	Optional	Width to be taken up by the Label; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element.
<b>Wrap</b>	Optional	If <i>True</i> , contents can wrap. If <i>False</i> , all the content appears on a single line. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Notes

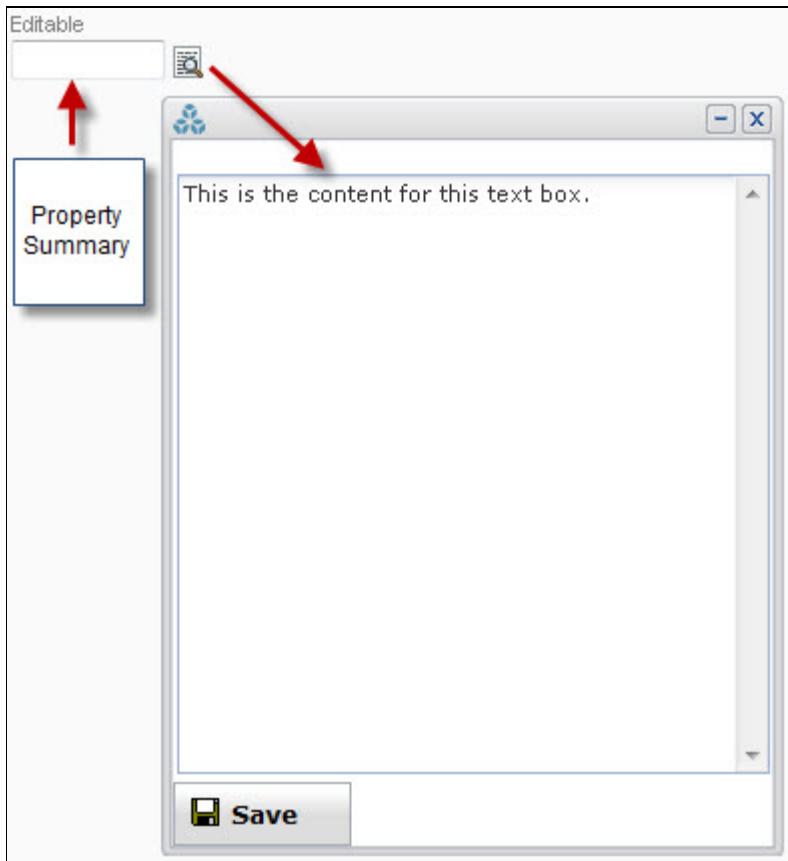
- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- \*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its

alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment.

- If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Large Text

The Large Text Form element enables you to add text into the property summary field, and display it in a dialog box. The property summary displays a read only summary of the text, based on user-defined settings. Users sets the property summary length, which is applied to the text and displayed in preview mode. Based on permission settings, the text displays in the dialog box as read only or editable.



In addition, the Large Text element can be structured inside of a table (see the following properties, Can Sort, Start Row, Cell Row Span, Cell Column Span, Group by In Table).

### Summary

<b>references Variable</b>	Yes, variables must be at the document level.
<b>references Method</b>	Yes Run Trigger, Editable, Icon Action, Visible, Optional, Label Orientation
<b>references Form</b>	No
<b>Allowable Child Elements</b>	None

### Properties

Property	Mandatory/Optional	Comment
<b>Name</b>	Mandatory	Name of the Label
<b>Label</b>	Mandatory	Visible label of the Label. If empty, the label assumes the Name of the Label. Default is empty.
		Applicable only when parent Element is a table. Specifies whether the Large Text Field can

		be sorted (true) or unsorted (false). The default is set to true.
Can Sort	Optional	This property is refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Label in the page or parent Element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Image shall occupy in the Grid Layout, to the right of current cell. Default is empty, which occupies one column.
Cell Row Span	Optional	<p>This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
Dynamic Label	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
Dynamic Label Style	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
Dynamic Tooltip	Optional	<p>Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i>. The static Tooltip property is considered as default until the action returns a value.</p> <p>When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.</p>
Dynamic Tooltip Style	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
Editable	Optional	<p>Specifies whether the Large Text Field is editable (true) or read-only (false). Default is NONE, which is editable. Setting this property to <i>True</i> enables editing of this element when it appear within a Table. The values for this property are either <i>True</i> (editable), <i>False</i> (read-only) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be read-only or editable).</p> <p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.</p>
Error Icon Gap	Optional	<p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
Error Icon	Optional	This property allows you to configure how an icon are displayed. This property takes the

<b>Orientation</b>		values left (default) and right.
<b>Group By In Table</b>	Optional	Displays the Large Text data in text area portion of the Finder results. Each results is grouped in the table based upon its size (from largest to smallest). Default is <i>False</i> .
<b>Height</b>	Optional	Height to be taken up by the Large Text area as defined as an integer value in pixels. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>Icon</b>	Optional	Choose an image file that appears as an icon to the right of the Large Text field. Default is /cwfv/text_view.png icon.
<b>Icon Action</b>	Optional	Choose from the list of Methods available in the User Interface to bind to the Icon. The Method is invoked when a user clicks on the Icon.
<b>Label Align</b>	Optional	Click the <b>Value</b> field and select from one of the following alignment options for your label: <ul style="list-style-type: none"><li>• <b>Left</b></li><li>• <b>Centre</b></li><li>• <b>Right</b></li></ul>
<b>Label Orientation</b>	Optional	Choose the location of the Label. Options include top, left or right.. The default orientation is <i>right</i> . However, in a Grid Layout , the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout.
<b>Label Style</b>	Optional	Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the Label such as font color. Default is NONE.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Run Trigger</b>	Optional	<p>The method assigned to this property is invoked whenever the data in the Large Text changes and the dialog box becomes out-of-focus. The method can be a User Action or a Script.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>• You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>• You click a menu.</li> <li>• You change your selection in a navigation tree.</li> <li>• You click another tab of a tabset.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• Field triggers do not fire on save. However, document triggers do.</li> <li>• As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
<b>Show Label</b>	Optional	This property determines whether the label is displayed on the Large Text. Choose from <i>True</i> or <i>False</i> . Default is True which means that the label is displayed.
<b>Start Row</b>	Optional	Applicable only when the parent element is a Grid Layout. If true, the Large Text appears in a new row of the grid. If <i>False</i> , it is simply placed in the next available cell. Default is <i>False</i> , which means that the Large Text element does not start in a new row.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the button such as positioning and color. Default is none.
<b>Summary Length</b>	Optional	Specifies the maximum number of characters that will display in the preview Large Text Field.
<b>Text Align</b>	Optional	Specifies the text alignment of the Large Text entry at runtime; it can be <i>left</i> , <i>center</i> , or <i>right</i> . Default is empty, which is right-aligned.
		<p>Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the textbox of the Large Text such as color and font. Default is NONE.</p> <p>To change the default disabled style for text fields, set this property with your own style, and</p>

<b>Textbox Style</b>	Optional	<p>make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the MyStyle style to the <b>Textbox Style</b> property, the stylesheet must have MyStyleDisabled to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Large Text. Default is empty, which means no tooltip for the Large Text.
<b>Variable</b>	Optional	Choose from the list of Variables available in the User Interface to bind to this Large Text. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Determines whether the Large Text is visible or not. Choose true (visible) or false (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
<b>Width</b>	Optional	Width to be taken up by the Large Text; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element. If this field is empty, the Width of the field should be taken up by that defined in the Column Width property of the Grid Layout. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

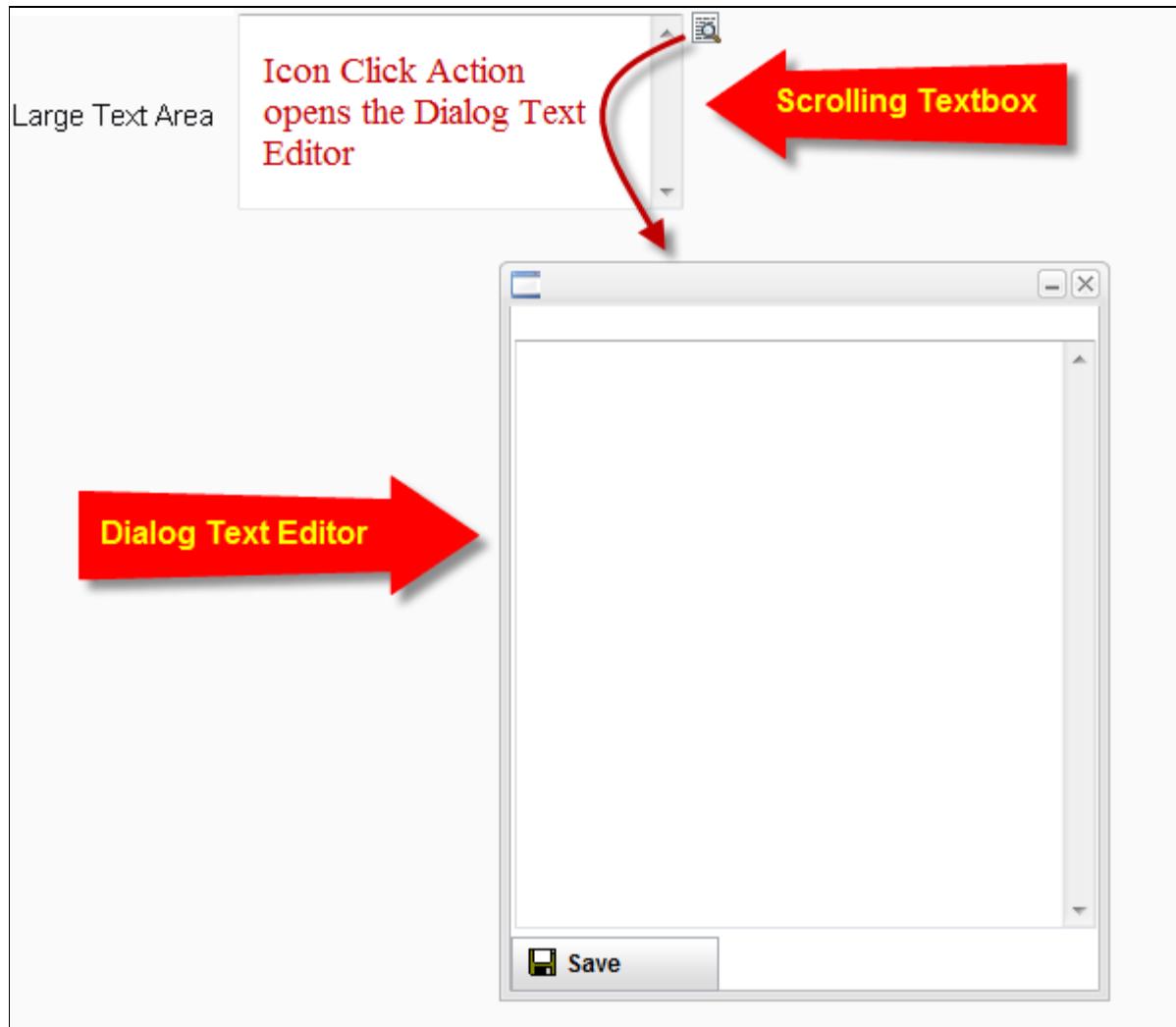
#### Notes:

- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout (for example, vlayout or layout with the width or height set), this element uses these values as a base to set its alignment. If the parent layout does not contain a width or height, then this element does not have a base to which it can set its alignment.
- If the element has a parent that contains width or height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width or height settings.
- If you use a reference or large text field under a finder whose output object does not have a user interface, a validation error occurs.

## Large Text Area

The Large Text Area element enables the user to add text to a scrolling text box via a dialog text editor at runtime. The summary of the content entered in this dialog text box will appear in the Large Text Area as read only. The number of summary characters displayed is based on user-defined value of the Summary Length property.

The Large Text element and the Large Text Area elements contain similar properties. The Large Text element enables content to be displayed in a textbox or a single line of content, whereas the Large Text Area allows content to be displayed in a scrolling textbox (with potentially many lines of content). Both these elements are only available under Document User Interface.



At runtime, the user clicks on the Large Text Area icon to open up the Dialog Text Editor for text entry. Once the user has entered in text into the Dialog Text Editor and save their work, the Textbox will display a summary of the entered text. The summary will contain text entered in the Dialog Text Editor to a maximum number of characters as defined by the value entered in the Summary Length property.

### Summary

references Variable	Yes
references Method	Yes, Icon Action, On Enter and Run Trigger.
references Form	No
Allowable Child Elements	<i>None</i>

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Large Text Area.
Label	Mandatory	Visible label of the Large Text Area. Default is empty which results in no label displayed.
Auto Fit Text	Optional	Select this checkbox to have the text within the <b>Text Area</b> field automatically fit. This checkbox is not selected by default.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Large Text Area in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Large Text Area occupies in the Grid Layout, to the right of current cell. Default is empty, which means it occupies one column.
Cell Row Span	Optional	<p>This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
Dynamic Label	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
Dynamic Label Style	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
Dynamic Tooltip	Optional	<p>Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i>. The static Tooltip property is considered as default until the action returns a value.</p> <p>When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.</p>
Dynamic Tooltip Style	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
Editable	Optional	<p>This property determines whether the Large Text Area is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be enabled or disabled). If the value is set to <i>True</i> then the user will be able to enter text in the Dialog Text Editor. If the value is set to <i>False</i>, then the Dialog Text Editor appears, but the user is unable to enter text in this dialog box.</p> <p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the</p>

		model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.
Error Icon Gap	Optional	<p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
Error Icon Orientation	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
Group By In Table	Optional	This property allows the user to group the Finder results by the elements value.
Icon	Optional	Choose an image file that appears as an icon to the right of the Large Text Area. Default is NONE.
Icon Action	Optional	When a <i>Document</i> is created, this property defaults to onFullTextEdit method. onFullTextEdit method invokes the Dialog Editor when the user clicks on the icon. The Dialog Editor allows content to be entered and saved.
Label Align	Optional	<p>Click the <b>Value</b> field and select from one of the following alignment options for your label:</p> <ul style="list-style-type: none"> <li>• <b>Left</b></li> <li>• <b>Centre</b></li> <li>• <b>Right</b></li> </ul>
Label Orientation	Optional	Specifies where the label of the Large Text Area appears in relation to the Large Text Area area; can be <i>left</i> , <i>right</i> or <i>top</i> of the Large Text Area area. The default orientation is <i>right</i> . However, in a Grid Layout, the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout.
Label Style	Optional	Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Label such as font color. Default is NONE.
On Enter	Optional	The system calls any script defined in this property. This method is invoked when the Large Text Area is focused on the Web page and the user presses the ENTER key.
Remove Label Line	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
Run Trigger	Optional	<p>The method assigned to this property is invoked whenever the value of the Large Text Area changes. The method can be a User Action Method or a Script.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>• You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>• You click a menu.</li> <li>• You change your selection in a navigation tree.</li> <li>• You click another tab of a tabset.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• Field triggers do not fire on save. However, document triggers do.</li> <li>• As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
Show Label	Optional	This property determines whether the label is displayed on the Large Text Area. Choose from <i>True</i> or <i>False</i> . Default is True which means that the label is displayed.
Start Row	Optional	Applicable only when the parent element is a Grid Layout. If true, the Large Text Area appears in a new row of the grid. If <i>False</i> , it is simply placed in the next available cell. Default is <i>False</i> , which means that the Large Text Area element does not appear on a

		new row.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the button such as positioning and color. Refer to the information on <a href="#">Style Sheets</a> for details on configuration. Default is none.
<b>Summary Length</b>	Optional	Specifies the maximum number of characters that will display in the Large Text Area scrolling textbox. This specifies the first characters displayed. For example a value of 15 will display in this property, will display the first 15 characters entered in the Dialog Editor in the scrolling textbox.
<b>Text Align</b>	Optional	Specifies the text alignment of the Large Text Area entry at runtime; it can be <i>left</i> , <i>center</i> , or <i>right</i> . Default is empty, which is right-aligned.
<b>Textbox Style</b>	Optional	<p>Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the textbox of the Large Text Area such as color and font. Default is NONE.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the MyStyle style to the <b>Textbox Style</b> property, the stylesheet must have MyStyleDisabled to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Large Text Area. Default is empty, which means no tooltip for the Large Text Area.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the large text area. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Variable</b>	Optional	The variable must be a <i>model</i> of the type <i>string</i> . Refer to notes below for more details on setup. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Determines whether the Large Text Area is visible or not. Choose true (visible) or false (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this field. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE or True, which means no permissions assigned and field is visible..
<b>Width</b>	Optional	Width to be taken up by the Large Text Area; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element. If this field is empty, the Width of the field should be taken up by that defined in the Column Width property of the Grid Layout. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

#### Notes:

- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- The Large Text Area must be set up as a *Document*:
  - Create a Document: Right-click <NameSpace> and click New > Data Dictionaries > Document
  - Create a Variable of type *string* for the Document (For example, variable called LTString)
  - Create the Large Text Area element: navigate to Documents > UserInterface > Default form. Override the Default form and create a Large Text Area element.
  - Assign the Variable property of the Large Text Area the *string* variable created above (For example, model.LTString)
  - Icon Action property will default to onFullTextEdit method.
- The following figure shows the Large Text Area element and the property values associated with the previous steps

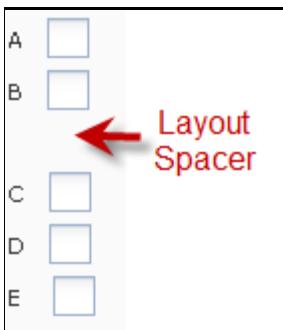
The screenshot shows a software interface with a large text area labeled "Large Text Area" at the top. Below it is a tree view of the element structure under "Default". The "String" node is selected. To the right is a properties grid.

Name	
Dynamic Label	
Dynamic Label Style	
Editable	
Group by In Table	false
Height	
Icon	/cwf/text_view.png
<b>Icon Action</b>	<b>onFullTextEdit</b>
Label	<b>Large Text Area</b>
Label Orientation	left
Label Style	
Name	String
Run Trigger	
Show Label	true
Start Row	false
Style	
Summary Length	10
Text Align	
Textbox Style	
Tooltip	
<b>Variable</b>	<b>model.LTString</b>

- If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout (for example, vlayout or layout with the width or height set), this element uses these values as a base to set its alignment. If the parent layout does not contain a width or height, then this element does not have a base to which it can set its alignment.
- If the element has a parent that contains width or height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width or height settings.
- If you use a reference or large text field under a finder whose output object does not have a user interface, a validation error occurs.

## Layout Spacer

Layout Spacer inserts empty space to Horizontal and Vertical Layouts. Properties **Height** and **Width** determine the size of the empty space.



### Summary

references Variable	No
references Method	Yes
references Form	No
Allowable Child Elements	None

### Properties

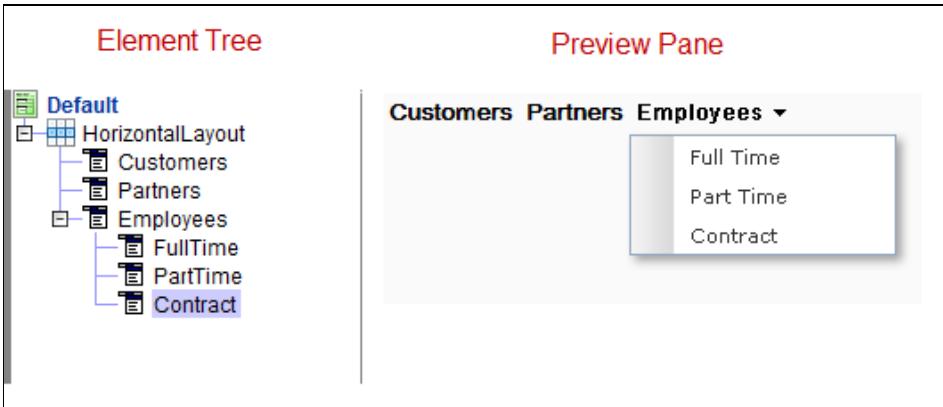
Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Layout Spacer
Height	Optional	Height used or consumed by the Layout Spacer. The value is defined as the number of pixels (integer) or as a percentage of page height or of the parent element. For example, 50% of the page height or 200 pixels.
Width	Optional	Width used or consumed by the Layout Spacer. The value is defined as the number of pixels (integer) or as a percentage of page width or of the parent element. For example, 30% of the page width or 150 pixels.
Style	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Layout Spacer such as positioning and color. Default is NONE.
On Enter	Optional	The system calls any script defined in this property. This method is invoked when the Layout Spacer is focused on the Web page and the user presses the ENTER key.
Visible	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions are assigned.

### Notes:

- A layout spacer that is last child is now the parent's layout RightMargin (if parent is vertical layout, layoutBottomMargin if horizontal layout)
- A layout spacer with a previous element above it is turned into the previous element's (if layout or form frame) layoutRightMargin (horizontal layout) or layoutBottomMargin (vertical layout or form frame)

## Menu

A menu can be created with a hierarchy of Menu Elements. The following is an example of a Menu element with child Menu elements to form a hierarchical structure.



### Summary

references Variable	No
references Method	Yes
references Form	No
Allowable Child Elements	Menu Separator

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Menu.
Label	Optional	Visible label or title of the Menu. If empty, a label or title is not displayed. Default is empty.
Accelerator	Optional	Enter a shortcut key (such as <i>Ctrl + s</i> ) to be used to invoke the Click Method of the menu. When the shortcut key entered in this property is typed by the user, the method for the menu's Click Method will be called. The accelerators are only enabled by the menutem's child.
Action Auditor	Optional	Specify the Action Validator object for this button's <b>Action Auditor</b> property. During runtime, when you click this button, the associated logging action is also invoked, provided that the <b>Enable action logging</b> checkbox from the System <a href="#">Logging</a> tab has been selected.
Can Focus	Optional	Headers on a page consist of a layout, an image button, and a menu item. Menu items are focusable by default (that is, the <b>Can Focus</b> checkbox is already checked). When you deselect this option, pressing the <b>Tab</b> key to tab within the form skips headers.
Click Method	Optional	Choose from the list of Methods available in the User Interface to bind to this Menu. The Method is invoked when a user clicks on the Menu Element. Use the property's <a href="#">lookup button</a> to view property details.  When adding Child Menu Elements, the Menu Element becomes a pure drop-down menu item and this method is no longer editable. The click method of a top-level menu cannot have an assigned click method as its click method purpose is to show the children drop-down menu.
Dynamic Style	Optional	Defines a dynamic style for the menu item. This property is used to select a method which returns the style based on whether the menu action has been called or not.
Dynamic Tooltip	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
Dynamic Tooltip Style	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
Enabled	Optional	When this property is set to false and the <b>Icon</b> property is set, icon_Disabled must be provided. Otherwise, it appears as a broken image.

		See the <a href="#">Apply Enabled Property for Second-level Menu Items and Under Iterators</a> for an example on how to use this property for second-level menu items and iterators.
<b>Height</b>	Optional	This property allows top-level menus to have their height adjusted. Only menus that are not children of other menus can set this property. In other words, top-level menus can set this property, but not menus that are shown as a submenu.
<b>Icon</b>	Optional	Choose from a list of icons. The selected icon appears in the menu item adjacent to the label. Default is NONE.
<b>Icon Height</b>	Optional	Height of the icon used or consumed within the Menu. The value is defined as the number of pixels (integer) or percentage.
<b>Icon Variable</b>	Optional	Choose from a list of available <i>string</i> variables. This value of this variable determines the icon that appears in the menu. Its value is the path where the icon is located. This variable can be initialized or dynamically set in any method within its local User Interface. The method should return a path to the icon (for example, "cwf/Tree.gif"). The icon will be invoked whenever the menu is visible or when a page is being refreshed (on any user action).
<b>Icon Width</b>	Optional	Width of the icon used or consumed within the Menu. The value is defined as the number of pixels (integer) or percentage.
<b>Image on Right</b>	Optional	Specifies whether the icon image is left or right justified within the Menu. Default is "false", which means that the image is left justified.
<b>Include in Tab Order</b>	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>• When set to &lt;Default&gt;, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>On Enter</b>	Optional	The system calls any script defined in this property (including onInit, onClick and toString). This method is invoked when the Menu is focused on the Web page and the user presses the ENTER key.
<b>Show Menu Drop Down</b>	Optional	The showMenuDropdown simply refers to a down arrow which would appear beside the menu element to indicate to the user that there are more menus to choose from under it. The developer can choose not to display this down arrow but still have access to the menu's menu item children.
<b>Show RollOver</b>	Optional	This value set as true would increase the font of the Menu label when the user's mouse hovers over the Menu element.
<b>Show Submenu On Hover</b>	Optional	This property is only available for root menu elements, as second-level menus already have hover capabilities. Selecting this checkbox indicates that you can hover your mouse over the menu item to display its submenus.
<b>Style</b>	Optional	Choose from styles within the selected style sheet (located in the Velocity Studio toolbar), which defines the styling of the Menu such as background color. Default is NONE.
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Menu. Default is empty, which means no tooltip for the Menu.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the element. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>URL</b>	Optional	Enter a Web path or URL (for example, <a href="http://google.com">http://google.com</a> ). This URL is invoked when the user clicks the menu label.
<b>Variable</b>	Optional	Allows you to dynamically set the menu's label. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.

## Notes

- A menu is created by assembling a hierarchy of Menu Elements together. Create expandable sub-menus of a menu item by adding child Menu Elements to it. You can have multiple levels in the menu hierarchy, but should limit the depth to, say two to three, for usability reasons.
- The example below shows three levels of menus. The first level of sub-menu expands vertically down from the root-level menu item. Other deeper levels of sub-menu expands horizontally from parent menu item.

The screenshot shows the Element Tree on the left and the Preview Pane on the right. The Element Tree displays a tree structure under a node named 'Default'. The 'Default' node has a 'HorizontalLayout' child, which contains 'Customers', 'Partners', and 'Employees'. The 'Employees' node has children 'PartTime' and 'FullTime', with 'PartTime' having 'Permanent' and 'Contract' as children. In the Preview Pane, a horizontal menu bar is shown with tabs for 'Customers', 'Partners', and 'Employees'. A dropdown menu is open under the 'Employees' tab, showing 'Part Time' and 'Full Time' as top-level items, with 'Permanent' and 'Contract' listed below 'Full Time'.

- Instead of Horizontal Layout, you can use Vertical Layout to stitch together a vertical menu.
- It is important to spend time on designing a style in CSS and assign it to the menu system to refine its appearance.
- The menu height/width can be controlled by how long the label is or its style. The icon used for the menu can also control the menu's overall size. Menus automatically resize to fit its label as well as its icon.

### Top-level Menus with the Accelerator Property

Using the **Accelerator** property provides added accessibility to both top-level menus and submenus. You can define shortcut keys at design time to access the top-level or submenu you want during runtime.

Accelerators for top-level menus function as follows:

- You can use a combination of the **Alt** key with a character (for example, A)
- When using Firefox as your Web browser, you use the **Alt** key with a character key
- When using Internet Explorer as your Web browser, you use the **Alt + Shift + character** key combination

Submenus with accelerators defined have the shortcut key displayed to the right of the menu label, which you specify at design time. For example, if you want to use the character key A as a shortcut key for the **Save** option, the menu would look as follows:



Accelerators for submenus have at least one character and a combination of one or more of these keys:

- Alt
- Shift
- Ctrl

**Note:** Some shortcut keys may already be used by Web browsers. As a result, the menu you have designed with shortcut keys may not always work. Ensure that you use shortcut keys that will not interfere with your Web browser's behaviour.

To use the **Accelerator** property with a top-level menu, do the following:

1. Create a top-level menu.
2. Locate the **Accelerator** property and click the ... button.
3. Specify a character key to use as your shortcut and then click the **Save** button.

The screenshot shows the Properties panel for a 'MenuItem' object. The 'Name' column lists properties like Accelerator, Action Auditor, Can Focus, Click method, Dynamic Label Style, Dynamic Tooltip, Dynamic Tooltip Style, Enabled, Icon, and Icon Height. The 'Value' column shows the configuration for each property. The 'Accelerator' property is currently selected, and a 'Key Set' dialog box is open. The 'Key Name' field is set to 'A', and the 'Alt Key' checkbox is checked. The 'Save' and 'Cancel' buttons are visible at the bottom of the dialog.

4. Select a **Click Method** for this menu from the drop-down menu.
5. During runtime, depending on the Web browser you are using, press the shortcut key combination. This action puts the focus on the menu item.

6. Press the **Enter** key to fire the click method you have specified for this menu.

To use the **Accelerator** property with a submenu, do the following:

1. Follow the same steps previously used to create a shortcut for a top-level menu. Instead of specifying a click method, add submenus.
2. During runtime, depending on the Web browser you are using, press the shortcut key combination, which puts the focus on the menu.
3. Press the down arrow key to show this menu's children.

## Apply Enabled Property for Second-level Menu Items and Under Iterators

The following example shows how to use the **Enabled** property for second-level menu items and under [iterators](#):

1. Create a top-level UserInterface called **MenuItemUI** with the following methods:
  - o **getLabel()**, which returns String
  - o **getIcon()**, which returns String
  - o **editablePerm()**, which returns either true or false, depending on some condition
  - o **getStyle()**, which returns String

The methods should return dynamic values based on some dynamic condition.

2. Create another top-level UserInterface called **testUI** and do the following:
  - a. Add two variables of *MenuItemUI* type to be used for two different iterators:
    - **iteratorLevel1**
    - **iteratorLevel2**
  - b. Select the **Array** property.
  - c. Add the **isMenuEnabled** method, depending on some condition.
  - d. Add a variable to bind to a checkbox to change the **Editable** condition (for example, **isMenuEnabledCheckbox**).
3. Create a multi-level menu in the **testUI** with the following static items:

```
Static Menu Item
--- Submenu
----- Static Item Level 2
----- Static Item Level 2
--- Static Item Level 1
```

Set the **Enabled** property for all **Static Item Level x** menu items to **isMenuEnabled**.

4. Create a multi-level menu in the **testUI** with iterator items:

```
Iterator Menu Item
--- Submenu
----- Iterator Level 2
----- Menu Item Iterator Level 2
--- Iterator Level 1
----- Menu Item Iterator Level 1
```

5. Set the **Variable** property for **Iterator Level 2** to **iteratorLevel2**, and set **Iterator Level 1** to **iteratorLevel1**.

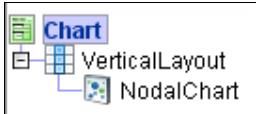
6. For the menu item under **Menu Item Iterator Level**, set the following properties:

- o **Variable** property to **iteratorLevel1.getLabel**
- o **Dynamic Style** property to **iteratorLevel1.getStyle**
- o **Image Variable** property to **iteratorLevel1.getIcon**
- o **Enabled** property to **iteratorLevel1.editablePerm**

7. Use the same properties in the previous step for **Menu Item Iterator Level 2**.

## Nodal Chart

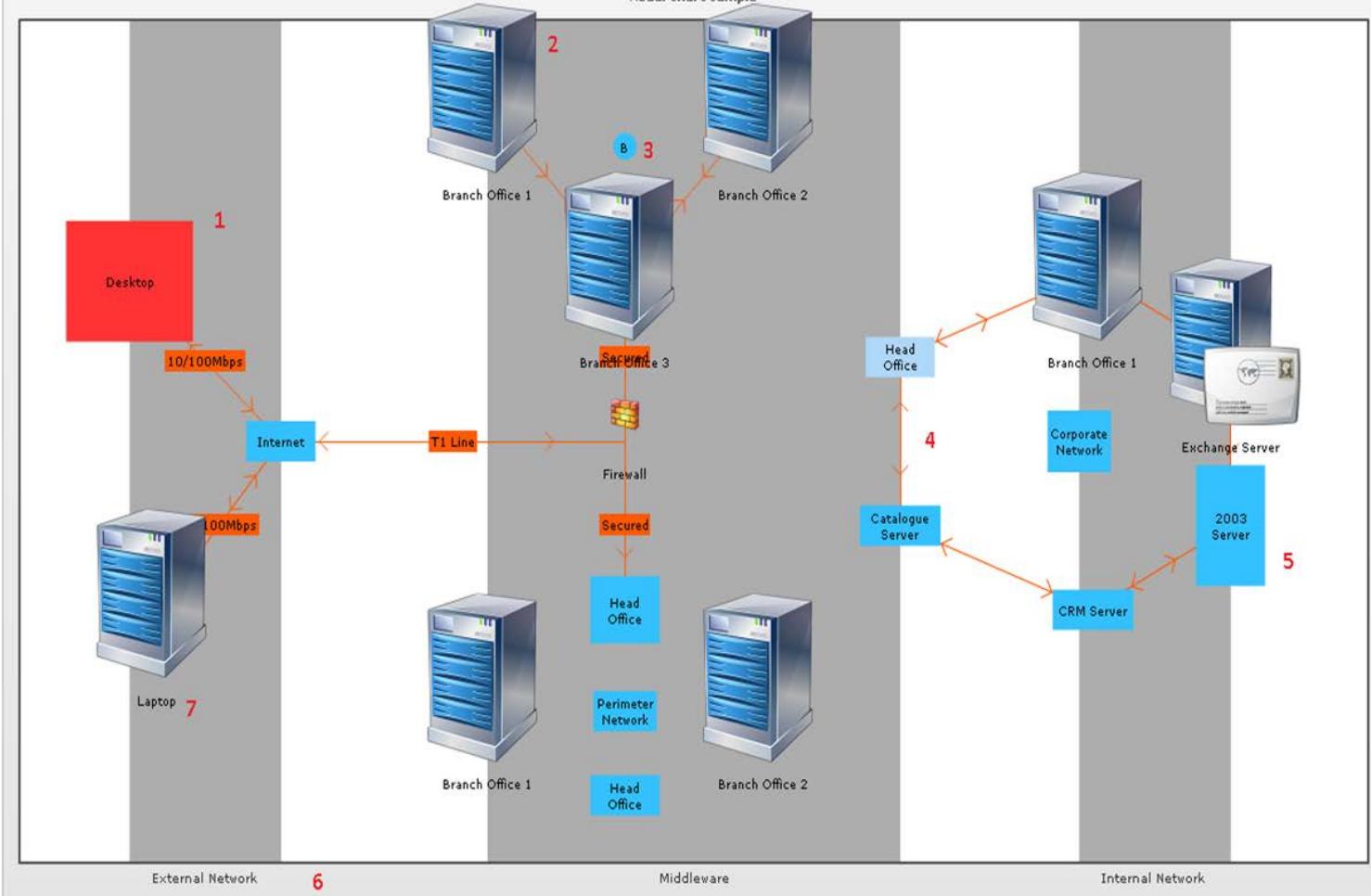
The nodal chart element allows you to build visual diagrams, such as network diagrams and organizational charts. The nodal chart element enables you to configure the overall look and feel of the chart display at runtime. The nodal chart exists under a finder user interface, relying on its result variable of type document array. Each document in the array represents a new node item.



Velocity Studio contains the `NodalChartDocument` system variable, which includes twelve variables that form a template structure for your nodal chart. The following table shows the required columns that comprise your document:

Name	Type	Description
xCoordinate	int; mandatory	The x-coordinate of the node item.
yCoordinate	int; mandatory	The y-coordinate of the node item.
width	int	The overall width of the node item, including space for labels.
height	int	The overall height of the node item, including space for labels.
id	string; mandatory	The unique ID for each node item to use with connectors.
color	string	The colour of the node item's shape if an <code>imageURL</code> is not provided.
imageURL	string	The resource location of the image to use for this node item.
shape	int	The shape index to use for the node item, which can be one of the following: <ul style="list-style-type: none"><li>• Circle</li><li>• Polygon</li><li>• Rectangle (default shape if a shape is not specified)</li></ul>
label	string	The label to show for the node item
trendName	string	The trend that this node item belongs to
connectTo	string	The node item's ID, which is used to connect this node item to another node item.
connectorLabel	string	The colour to fill the completed part of the task

The following is an example of a nodal chart at runtime:



The sample nodal chart shows the following:

1. Node item with a specified colour and label.
2. Node item with an image URL specified.
3. Node item with shape specified as 1 (circle).
4. Connector with no label specified for the node item. Arrows may be removed by changing the node chart element's **Show Arrow at End** and **Show Arrow at Start** properties.
5. Node item with label and its connector specified (defaults to rectangular shape).
6. Trends or groupings. When shown at the bottom of the nodal chart, the x-coordinate range for each trend is calculated based on the minimum and maximum x value out of all node items that have each trend specified under trendName. The trend position can be changed by updating the nodal chart's **Trend At Bottom** property.
7. Node item label. The label's position can be changed using the **Label Align** nodal chart property. Label alignment is always centered if an image is not being used for the node item.
8. Nodal chart caption.

## Display Multiple Charts at Runtime

You can display multiple charts in a Form by creating multiple Form Frame Form elements and then referring the variable of each element to the Finder User Interface Chart Form that contains the chart that you want to display. Refer to [Displaying multiple charts at runtime](#) for details.

## Summary

references Variable	Yes
references Method	No
references Form	No

## Properties

The following are the properties for the nodal chart element:

Property	Mandatory/Optional	Comment
Name	Mandatory	The nodal chart name.
Variable	Mandatory	Finder result form for the nodal chart. Use the property's <a href="#">lookup button</a> to view property details.
Caption	Optional	The title (label) of the nodal chart.
Click Method	Optional	When you specify this property, chart elements are clickable. Clicking these elements invokes the method. The method receives two parameters: <ul style="list-style-type: none"> <li>• The data object that the chart node represents</li> <li>• The variable name, if applicable, that the chart node represents</li> </ul>
Connector Thickness (max 10)	Optional	The line thickness that connects two nodes in the nodal chart. The higher the number, the thicker the connection line is between nodes. This property's default value is 2.  <b>Note:</b> If you enter a number that exceeds 10, the Velocity Studio automatically overrides this value and sets this field to 10.
Dynamic Height	Optional	This property allows you to dynamically assign a variable or a script to the <b>Dynamic Height</b> property. For a variable, the data type must be either an integer or string. For the script, only the script action can be used and its return type must be defined as either integer or string.  At runtime, by default, the regular width and height are used. However, if <b>Dynamic Width</b> and <b>Dynamic Height</b> are defined, they replace the regular width and height, and the values are calculated based on either the variable or script.
Dynamic Width	Optional	This property allows you to dynamically assign a variable or a script to the <b>Dynamic Width</b> property. For a variable, the data type must be either an integer or string. For the script, only the script action can be used and its return type must be defined as either integer or string.  At runtime, by default, the regular width and height are used. However, if <b>Dynamic Width</b> and <b>Dynamic Height</b> are defined, they replace the regular width and height, and the values are calculated based on either the variable or script.
Height	Optional	The height of the nodal chart. Click the <b>Value</b> field, and then select whether you want the height represented in pixels or a percentage using the drop-down menu.  <b>Note:</b> Although the Height property does not have a maximum limit, specifying a large pixel or percentage in this field may show a black screen during runtime. As a result, it is recommended that you select an appropriate height value for your nodal chart.
Label Align	Optional	The position of the label. The label's position defaults to centre.
Show Connector Arrow at End	Optional	Select this checkbox to show the connector arrow at the end of the line.
Show Connector Arrow at Start	Optional	Select this checkbox to display the connector arrow at the start of the line.
Show Node Labels	Optional	Select this checkbox to display the node labels in the nodal chart.
Show Trend Label	Optional	Select this checkbox to show the trend name in the nodal chart.
Show Validation	Optional	This boolean property has values of either <i>True</i> or <i>False</i> , which determines whether a validation warning message displays at runtime. By default, this property is unchecked, meaning that validation errors do not appear under the chart at runtime and supports backward compatibility. Otherwise, selecting this property allows validation errors to appear under the chart at runtime.
Style	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Grid Layout such as positioning and color. Default is NONE.
Trend At	Optional	Select this checkbox to display the trend (grouping) at the bottom of the nodal chart.

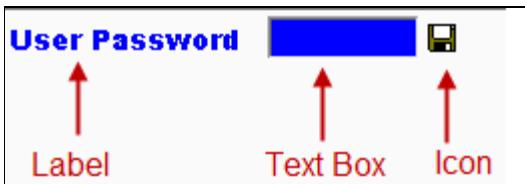
<b>Bottom</b>		
<b>Visible</b>	Optional	This field defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> , such as addPerm, attachPerm, deletePerm, editablePerm, and visiblePerm can be assigned as well as Object Permissions. This field defaults to NONE, which means that no permissions are assigned.
<b>Width</b>	Optional	The width of the chart. Click the <b>Value</b> field, and then select whether you want the width represented in pixels or a percentage using the drop-down menu.

**Notes:**

- To view charts, Adobe Flash Player 8.0 or higher must be installed for the Firefox browser. Adobe Flash Player 8.0 or higher as an Active X object must be installed for Internet Explorer. Adobe Flash Player can be downloaded from Adobe's Web site at [www.adobe.com](http://www.adobe.com).
- To view JavaScript-based charts, FusionCharts automatically renders them on computers that do not have a Flash player installed. The only client-side requirement is a browser that supports JavaScript.

## Password Field

The Password field is a textbox that masks the input at runtime. It is ideal for representing passwords. The input values to this field can be sorted and grouped at runtime when used with the Finder functionality. There are many properties of this element that can be controlled including style of the textbox, label and element area. At runtime, the label can be changed dynamically (using the Variable and Dynamic Style properties), the field can be enable or disabled and methods can be triggered.



### Summary

references Variable	Yes
references Method	Yes - On Enter, Run Trigger, Permissions
references Form	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Password Field.
Label	Mandatory	Visible label of the Password Field. If empty, there is no Password Field label. Default is empty.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Password Field in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Password Field occupies in the Grid Layout, to the right of the current cell. Default is empty, which means it occupies one column.
Cell Row Span	Optional	<p>This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
Data Length	Optional	Specifies the maximum number of characters that can be entered into the password textbox. Default is 0, which means no character restrictions.
Dynamic Label	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.

<b>Dynamic Label Style</b>	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
<b>Dynamic Tooltip</b>	Optional	<p>Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i>. The static Tooltip property is considered as default until the action returns a value.</p> <p>When this element is used under the Table element, the Dynamic Tooltip property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.</p>
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Editable</b>	Optional	<p>This property determines whether the Password Field is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be enabled or disabled).</p> <p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.</p>
<b>Error Icon Gap</b>	Optional	<p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Font Properties</b>	Optional	Specifies the font properties of the content entered in password text/mask at runtime: font-family, font-color, font-size, bold, italic and underline. This overrides the styling assigned, if any, to the textbox.
<b>Height</b>	Optional	Height to be taken up by the Password Field area as defined as an integer value in pixels. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>Icon</b>	Optional	Choose an image file that appears as an icon to the right of the Password Field. Default is NONE.
<b>Include in Tab Order</b>	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>• When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>Label Align</b>	Optional	<p>Click the <b>Value</b> field and select from one of the following alignment options for your label:</p> <ul style="list-style-type: none"> <li>• <b>Left</b></li> <li>• <b>Centre</b></li> </ul>

		<ul style="list-style-type: none"> <li>• Right</li> </ul>
<b>Label Orientation</b>	Optional	Specifies where the label of the Password Field appears in relation to the Password Field area; can be <i>left</i> , <i>right</i> or <i>top</i> of the Password Field area. The default orientation is <i>right</i> . However, in a Grid Layout, the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout..
<b>Label Style</b>	Optional	Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Label such as font color. Default is NONE.
<b>On Enter</b>	Optional	The system calls any script defined in this property. This method is invoked when the Password Field is focused on the Web page and the user presses the ENTER key.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Run Trigger</b>	Optional	<p>The method assigned to this property is invoked whenever the value of the Password Field changes. The method can be a User Action Method or a Script.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>• You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>• You click a menu.</li> <li>• You change your selection in a navigation tree.</li> <li>• You click another tab of a tabset.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• Field triggers do not fire on save. However, document triggers do.</li> <li>• As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
<b>Show Label</b>	Optional	This property determines whether the label is displayed on the Password Field. Choose from <i>True</i> or <i>False</i> . Default is True which means that the label is displayed.
<b>Start Row</b>	Optional	Applicable only when parent Element is a Grid Layout. If true, the Password Field starts in a new row of the grid. If false, it is simply placed in the current cell as-is. Default is <i>false</i> , which does not start a new row.
<b>Style</b>	Optional	Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Password Field such as color and font. Default is NONE.
<b>Text Align</b>	Optional	Specifies the alignment of the text in Password Field; it can be <i>left</i> , <i>center</i> , or <i>right</i> . Default is empty, which is left-aligned.
<b>Textbox Style</b>	Optional	<p>Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the textbox of the Password Field such as color and font. Default is NONE.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the MyStyle style to the <b>Textbox Style</b> property, the stylesheet must have MyStyleDisabled to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the <b>Password</b> field. Default is empty, which means no tooltip for the <b>Password</b> field.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the <b>Password</b> field. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Variable</b>	Optional	Choose from the list of Variables available in the User Interface to bind to this Password Field. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.

<b>Visible</b>	Optional	Determines whether the Password Field is visible or not. Choose true (visible) or false (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this field. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE or True, which means no permissions assigned and field is visible.
<b>Width</b>	Optional	Width to be taken up by the Password Field; can be in integer which is in pixels, or as a percentage (for example, "30%") of the page width or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Notes

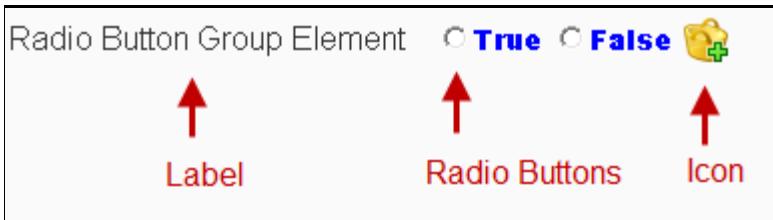
- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- If Password Field is parented by a Grid Layout, the label orientation of the Password Field follows Grid Layout's **Label Orientation** property.
- Separate styles can be assigned to different parts of the Password Field: field area, textbox and label. As a reference of which style governs which part of the field, here's a Password Field with *Style*=grey background, *Label Style*=red font color, *Textbox Style*=blue background:



- If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment.
- If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Radio Button Group

Creates a group of radio buttons based on enumeration of the assigned variable. When user clicks on a radio-button, the button becomes checked, and all other buttons with the same name become unchecked.



### Summary

references Variable	Yes
references Method	No
references Form	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Radio Button Group.
Label	Mandatory	Visible label of the Radio Button Group. If empty, there is no Radio Button Group label. Default is empty.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Radio Button Group in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Radio Button Group occupies in the Grid Layout, to the right of the current cell. Default is empty, which means it occupies one column.
Cell Row Span	Optional	<p>This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
Default Enumeration	Optional	By default, this property is set to <b>yes/no</b> as a radio button (Boolean) in migration. The default product behaviour is <b>true/false</b> .
Dynamic Label Style	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.

<b>Editable</b>	Optional	<p>This property determines whether the Radio Button Group is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be enabled or disabled).</p> <p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's <i>editablePerm</i> permission takes over. If this method is not overwritten, the model's <i>editablePerm</i> permission is used. User interfaces displayed under another user interface with <i>editablePerm</i> are also be inherited if the embedded user interfaces do not have their own <i>editablePerm</i> permission.</p>
<b>Error Icon Gap</b>	Optional	<p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the <i>/cwfv/error.png</i> icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
<b>Error Icon Orientation</b>	Optional	<p>This property allows you to configure how an icon are displayed. This property takes the values <i>left</i> (default) and <i>right</i>.</p>
<b>Group By In Table</b>	Optional	<p>This property allows the user to group the Finder results by the elements value.</p>
<b>Icon</b>	Optional	<p>Choose an image file that appears as an icon to the right of the Radio Button Group. Default is NONE.</p>
<b>Include in Tab Order</b>	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>Label Orientation</b>	Optional	<p>Specifies where the label of the Radio Button Group appears in relation to the Radio Button Group area; can be <i>left</i>, <i>right</i> or <i>top</i> of the Radio Button Group area. The default orientation is <i>right</i>. However, in a Grid Layout, the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout..</p>
<b>Label Style</b>	Optional	<p>Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Label such as font color. Default is NONE.</p>
<b>Remove Label Line</b>	Optional	<p>Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.</p>
<b>Run Trigger</b>	Optional	<p>The method assigned to this property is invoked whenever the value of the Radio Button Group changes. The method can be a User Action Method or a Script.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>You click a menu.</li> <li>You change your selection in a navigation tree.</li> <li>You click another tab of a tabset.</li> </ul>

		<b>Notes:</b> <ul style="list-style-type: none"> <li>Field triggers do not fire on save. However, document triggers do.</li> <li>As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
<b>Show Label</b>	Optional	This property determines whether the label is displayed on the Radio Button Group. Choose from <i>True</i> or <i>False</i> . Default is <i>True</i> which means that the label is displayed.
<b>Start Row</b>	Optional	Applicable only when parent Element is a Grid Layout. If true, the Radio Button Group starts in a new row of the grid. If false, it is simply placed in the current cell as-is. Default is <i>false</i> , which does not start a new row.
<b>Style</b>	Optional	Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Radio Button Group such as color and font. Default is <b>NONE</b> .
<b>Textbox Style</b>	Optional	<p>Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the radio button labels such as color and font. Default is <b>NONE</b>. Refer to example in the Notes section.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the <b>MyStyle</b> style to the <b>Textbox Style</b> property, the stylesheet must have <b>MyStyleDisabled</b> to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Radio Button Group. Default is empty, which means no tooltip for the Radio Button Group.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the radio button group. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Variable</b>	Optional	<p>Choose from a list of variables of the type <i>string</i>, <i>integer</i> or <i>boolean</i> when this variable has enumeration values and exists under a document variable list. Use the property's <a href="#">lookup button</a> to view property details.</p> <p><b>Note:</b> The Radio Button Group variable property cannot bind to data types belonging to a user interface. The data type used must belong to a document. See the notes section for setup.</p>
<b>Vertical</b>	Optional	Specifies the orientation of the radio buttons and their associated labels. Setting this property to <i>True</i> results in a vertical orientation and to <i>False</i> in a horizontal orientation. Refer to the example below for further details.
<b>Visible</b>	Optional	Determines whether the Radio Button Group is visible or not. Choose true (visible) or false (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this field. Pre-defined system <a href="#">Permissions</a> such as <code>addPerm</code> , <code>deletePerm</code> , <code>editablePerm</code> , <code>attachPerm</code> , <code>executePerm</code> , and <code>visiblePerm</code> can be assigned as well as Object Permissions. Defaults to <b>NONE</b> or <i>True</i> , which means no permissions assigned and field is visible.
<b>Width</b>	Optional	Width to be taken up by the Radio Button Group; can be in integer which is in pixels, or as a percentage (for example, "30%") of the page width or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Notes

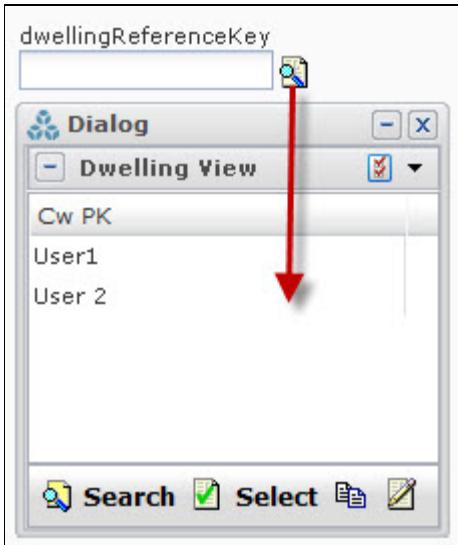
- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- You should always assign a Variable with enumeration defined to a Radio Button Group such a a variable of type *boolean*.
- Radio Button Group can be rendered horizontally or vertically based on the boolean *Vertical* property.

Vertical=false	Vertical=true
Radio Button <input checked="" type="radio"/> <b>True</b> <input type="radio"/> <b>False</b> 	Radio Button <input type="radio"/> <b>True</b> <input checked="" type="radio"/> <b>False</b> 

- In the example above, the **Textbox Style** property changed the style of the radio button labels "True and False" values to a blue font color.
- If Radio Button Group is parented by a Grid Layout, the label orientation of the Radio Button Group follows Grid Layout's **Label Orientation** property.
- If the Radio Button Group is assigned with a boolean Variable, the value cannot be nullified (that is,either *True* or *False* radio button is selected and cannot be de-selected). If a choice of *null* is required, you may choose to use a [Select Element](#).
- If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.
- You should always assign a Variable with enumeration defined to a Radio Button Group. Here are the setup steps:
  - A. Set up an enumeration data type for a Document:
    1. Create a new Data Type that Extends from String, Boolean or Integer: <namespace> > Data Dictionaries > Data Type > New Data Type (For example, Name = EnumString)
    2. Enter enumeration values in the Enumeration Tab for selection at runtime.
  - B. Create a Radio Button Group element in a Document User Interface
    1. Create a User Interface Document: <namespace> > Data Dictionaries > Documents
    2. Add the Variable (EnumString) created above to the Document: Navigate to the Variables tab and click the  icon to add the variable.
    3. In the UserIntefrace, override the Default Form
    4. Add the Form Element *Radio Button Group*
    5. Assign the enumerated variable added in step 2 to the *Variable* property.

## Reference

Implementation of Reference Fields is achieved by first creating a data type of type reference. A finder is then specified under this data type. A visual label for Reference Fields is determined using the result document's `cwOnDocVisualKey` method. If implementation of this method is not provided (the return value is null), the primary key of the document is used.



In addition, the Reference element can be structured inside of a table (see the following properties, Can Sort, Start Row, Cell Row Span, Cell Column Span, Group by In Table).

### Summary

references Variable	Yes
references Method	Yes
references Form	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Label.
Label	Optional	Visible label of the Label. If empty, there is no Label label. Default is empty.
Can Sort	Optional	Specifies whether the Reference Field can be sorted (true) or unsorted (false). The default is set to true. This option is available when the Reference Field appears in a table.
Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Label in the page or parent Element. Default is undefined, which behaves as if it is top-left
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Label shall occupy in the Grid Layout, to the right of current cell. Default is empty, which occupy one column.
Cell Row		This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.

<b>Span</b>	Optional	<p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
<b>Data Length</b>	Optional	Specifies the maximum number of characters that can be entered into the text box. Default is 0, which means no character restrictions.
<b>Editable</b>	Optional	<p>Specifies whether the Reference Field is editable (true) or read-only (false). Default is NONE, which is editable. Setting this property to <i>True</i> enables editing of this element when it appear within a Table.</p> <p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.</p>
<b>Enumeration Reference</b>	Optional	<p>This property provides a list of all data types within the metadata that have either an enumeration or a reference finder defined.</p> <p>At runtime, when the form is displayed, the value is still retrieved from the <b>Variable</b> property. The variable set on the fields can be primitive strictly for storing values. When you click the reference icon, the product uses the enumeration reference data type to find the finder user interface to display in a dialog.</p> <p>See the <a href="#">example</a> on how to use this property for details.</p>
<b>Font Properties</b>	Optional	Specifies the font properties of the label on a pop-up prompt: font-family, font-color and font-size. This overrides the styling assigned, if any, to the label.
<b>Group by In Table</b>	Optional	Displays the Reference Field data in Finder results. The Reference data is displayed in a column with elements that are grouped with the same name.
<b>Height</b>	Optional	Height to be taken up by the Label; can be in integer which is in pixels, or in percentage (for example, "50%") of page height or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>Icon</b>	Optional	If an icon is selected, the icon appears in the Label, besides the label. Default is NONE.
<b>Icon Action</b>	Optional	Only applicable when an Icon is specified. Choose from the list of Methods available in the User Interface to bind to the Icon. The Method is invoked when a user clicks on the Icon.
<b>Include Icon in Tab Order</b>	Optional	The property allows you to optionally exclude the icon button from the tab order.
<b>Include in Tab Order</b>	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>• When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>Label</b>	Optional	Choose the location of the Label. Options include top, left or right.

Orientation		
<b>Label Style</b>	Optional	Choose from styles in the selected style sheet on the toolbar, which defines the styling of the Label such as font color. Default is NONE.
<b>On Enter</b>	Optional	When data in the text box is selected and the user presses ENTER, the action of the selected Method is invoked.
<b>Optional</b>	Optional	Error validation occurs when no data is entered in the large text box.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Run Trigger</b>	Optional	<p>When data is changed in the Reference Field, and the dialog box becomes out-of-focus, the action of the selected Method is invoked immediately after the reference is selected in the popup finder. This property can be bound to any method of the UserInterface data structure.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>• You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>• You click a menu.</li> <li>• You change your selection in a navigation tree.</li> <li>• You click another tab of a tabset.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• Field triggers do not fire on save. However, document triggers do.</li> <li>• As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> <li>• The trigger does not fire for editable visual key types when entering a visual key value immediately in the reference text field.</li> </ul>
<b>Start Row</b>	Optional	Applicable only when parent Element is a Grid Layout. If true, the Label starts in a new row of the grid. If false, it is placed in the current cell as-is. Default is /default, which does not start a new row.
<b>Style</b>	Optional	Defines the styling of the Label such as color and font. Default is NONE.
<b>Text Align</b>	Optional	Specifies the alignment of the label; it can be <i>left</i> , <i>center</i> , or <i>right</i> . Default is empty, which is left-aligned.
<b>Textbox Style</b>	Optional	<p>Choose from styles in the selected style sheet on the toolbar, which defines the styling of the text box of the Textfield such as color and font. Default is NONE.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the MyStyle style to the <b>Textbox Style</b> property, the stylesheet must have MyStyleDisabled to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Label. Default is empty, which means no tooltip for the Label.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the label. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Variable</b>	Optional	Choose from the list of Variables available in the User Interface to bind to this Reference Field. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.

		<b>Note:</b> The Reference variable property cannot bind to data types belonging to a user interface. The data type used must belong to a document.
<b>Visible</b>	Optional	Determines whether the Label is visible or not. Choose true (visible) or false (not visible) or from the list of Methods available in the User Interface to bind to the Field.
<b>Width</b>	Optional	Width to be taken up by the Label; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

#### Notes:

- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- If a Reference Form element is being used as a Reference Finder, the *showDetailAction* method takes the currently selected Document's ID and sets it as this Finder's parent document's variable value.
- If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout (for example, vlayout or layout with the width or height set), this element uses these values as a base to set its alignment. If the parent layout does not contain a width or height, then this element does not have a base to which it can set its alignment.
- If the element has a parent that contains width or height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width or height settings.
- When metadata contains a reference field bound to non-reference data types, a warning message occurs in Velocity Studio. If the data type is incorrect, the value is restored from the database or metadata. testcase verifies that a new warning is added in Velocity Studio when metadata contains a reference field bound to non-reference data types.

**Note:** The functionality of the reference field, such as the icon click, does not work without a reference data type.

- If you use a reference or large text field under a finder whose output object does not have a user interface, a validation error occurs.
- Displaying visual keys for multi-select reference fields does not work with the editable visual key format.

#### Add an Enumeration Type Reference Field in top-level UI

You can add an enumeration type Reference Field element in a top-level UI by completing these steps:

1. Select a form in a top-level UI.
2. Right-click the form and select the **Add..** option.
3. From the Add Element dialog, expand the **Input Elements** node, select the **Reference Field** element, and click the **Finish** button.
4. The Reference Field element appears in the top-level UI.

#### Example: Using the Enumeration Reference property

The following example shows how to use the **Enumeration Reference** property for both Select Field and Reference Field elements:

1. Create a reference data type with a finder property set. Ensure that the finder's output object has a visual key set up either through cwOnDocVisualKey or cwOnStructVisualKey.
2. Create a document with two variables:
  - selectFieldVariable of type String
  - refFieldVariable of type String
3. Override the user interface.
4. Add a grid layout with Select Field element and a Reference Field element. Set the **Variable** property to one of the variables in step 2.
5. Set the **Enumeration Reference** property to the data type created in step 1.

At runtime, when you click the reference icon, a dialog appears, showing the finder attached to the **Enumeration Reference** data type. The select drop-down field contains a list from the finder specified in step 1. The visual key also appears in the user interface.

## Rich Text Editor

The Rich Text Editor element enables you to create an area to perform rich text editing, such as making changes to your text's font, colour, size, alignment, and more. This element is available as an input form element.

The properties of the Rich Text Editor element are similar to the [Text Area](#) element, except for the **Control Options** property, which lets you specify which groups of controls that you want to include in the editor's toolbar.

The following example shows the editing area for editing text. A **Set Canvas HTML** button has also been created in this example. When you click this button, your text that you have worked on in the editor appears in the viewing area under the editor itself.



The HTML generated from this component may vary by Web browser. This component has limited support when viewed with the Safari Web browser.

### Summary

<b>references Variable</b>	Yes
<b>references Method</b>	Yes, Run Trigger, On Enter.
<b>references Form</b>	No
<b>Allowable Child Elements</b>	None

### Properties

Property	Mandatory/Optional	Comment
<b>Name</b>	Mandatory	Name of the Rich Text Editor area.
<b>Variable</b>	Mandatory	Choose from the list of Variables available in the User Interface to bind to this Rich Text Editor. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.  The variable of the Rich Text Editor element stores the editor's HTML text. The text is then stored in the database. The variable must have String as its datatype and this string must have an unlimited size.
		This property indicates whether the focus moves to the next input field when the permitted length of the data is reached. By default, this checkbox property is selected. The next field

<b>Auto-focus to next field</b>	Optional	<p>is considered to be any input field (for example, text, select field, etc.) that is visible and editable (that is, not disabled).</p> <p>The behaviour applies to the following:</p> <ul style="list-style-type: none"> <li>• Masked text field, where the <b>Display Format</b> defines the permitted length</li> <li>• Regular text field, where the <b>Data Length</b> indicates the permitted length</li> </ul>
<b>Can Sort</b>	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
<b>*Cell Alignment</b>	Optional	Controls the horizontal and vertical positioning of the Rich Text Editor area in the page or parent element. Default is undefined, which behaves as if it is top-left.
<b>Cell Column Span</b>	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Rich Text Editor occupies in the Grid Layout, to the right of the current cell. Default is empty, which means it occupies one column.
<b>Cell Row Span</b>	Optional	<p>This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
<b>Control Options</b>	Optional	<p>This property allows you to specify which groups of controls to include in the editor's toolbar by selecting the appropriate checkboxes from the Controls dialog:</p> <ul style="list-style-type: none"> <li>• <b>fontControls</b> When selected, this control displays the <b>Set Font</b> and <b>Set Font Size</b> dropdown menus in the editor's toolbar.</li> <li>• <b>formatControl</b> Select this checkbox to display whether your highlighted text is aligned to the left, right, or centre, or is justified.</li> <li>• <b>styleControls</b> When selected, this control displays the bold, italics, and underline style options for your highlighted text.</li> <li>• <b>colorControls</b> Select this checkbox to specify a colour for your highlighted text or customize a specific colour to suit your needs.</li> </ul> <p><b>Note:</b> At least one control must be selected. Deselecting all controls is not permitted.</p>
<b>Data Length</b>	Optional	Specifies the maximum number of characters that can be entered into the select textbox. Default is 0, which means no character restrictions.
<b>Dynamic Label</b>	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
<b>Dynamic Label Style</b>	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
<b>Dynamic Tooltip</b>	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
		When this element is used under the Table element, the <b>Dynamic Tooltip</b> property

		accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
		This property determines whether the Rich Text Editor is enabled or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be enabled or disabled).
<b>Editable</b>	Optional	<b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.
<b>Error Icon Gap</b>	Optional	Select this property's checkbox to make a gap for the validation icon. By default, this property is false.  <b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Font Editor Width</b>	Optional	This property denotes the font selector field's width in pixels. The field's default value is 150 px.
<b>Font Editor Prompt</b>	Optional	This property represents the text message in the font selector field when no font has been selected. The default text message for this field is <b>Set Font....</b> This field supports translation.
<b>Font Size Editor Width</b>	Optional	This property denotes the font size selector field's width in pixels. The field's default value is 150 px.
<b>Font Size Editor Prompt</b>	Optional	This property represents the text message in the font size selector field when no font size has been selected. The default text message for this field is <b>Set Font Size....</b> This field supports translation.
<b>Group By In Table</b>	Optional	This property allows the user to group the Finder results by the elements value. Default is <i>False</i> .
<b>Height</b>	Optional	Height to be taken up by the Rich Text Editor as defined as an integer value in pixels. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>Icon</b>	Optional	Choose an image file that appears as an icon to the right of the Rich Text Editor. Default is NONE.
<b>Include Icon in Tab Order</b>	Optional	The property allows you to optionally exclude the icon button from the tab order.
<b>Include in Tab Order</b>	Optional	This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <u>suppress tabbing</u> configuration setting and can take the following values: <ul style="list-style-type: none"> <li>When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> </ul>

		<ul style="list-style-type: none"> <li>If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>Label Orientation</b>	Optional	Specifies where the label of the Rich Text Editor appears in relation to the Rich Text Editor area; can be <i>left</i> , <i>right</i> or <i>top</i> of the Rich Text Editor area. The default orientation is <i>right</i> . However, in a Grid Layout, the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout..
<b>Label Style</b>	Optional	Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the Label such as font color. Default is NONE.
<b>On Enter</b>	Optional	The system calls any script defined in this property. This method is invoked when the Rich Text Editor is focused on the Web page and the user presses the <i>Enter</i> key.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Run Trigger</b>	Optional	<p>The method assigned to this property is invoked whenever the value of the Rich Text Editor changes. The method can be a User Action Method or a Script.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>You click a menu.</li> <li>You change your selection in a navigation tree.</li> <li>You click another tab of a tabset.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>Field triggers do not fire on save. However, document triggers do.</li> <li>As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
<b>Show Label</b>	Optional	This property determines whether the label is displayed in the Rich Text Editor. Choose from <i>True</i> or <i>False</i> . Default is True which means that the label is displayed.
<b>Start Row</b>	Optional	Applicable only when parent Element is a Grid Layout. If <i>True</i> , the Rich Text Editor starts in a new row of the grid. If <i>False</i> , it is simply placed in the current cell as-is. Default is <i>False</i> , which does not start a new row.
<b>Style</b>	Optional	Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Rich Text Editor, such as colour and font. Refer to the information on <a href="#">Style Sheets</a> for details on configuration. Default is NONE.
<b>Textbox Style</b>	Optional	<p>Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the textbox of the Rich Text Editor, such as colour and font. Default is NONE.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the MyStyle style to the <b>Textbox Style</b> property, the stylesheet must have MyStyleDisabled to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Rich Text Editor. Default is empty, which means no tooltip for the Rich Text Editor.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the Rich Text Editor. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.

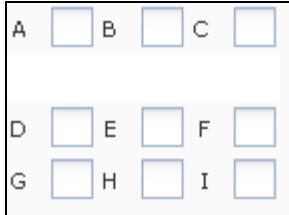
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Visible</b>	Optional	Determines whether the Rich Text Editor is visible or not. Choose true (visible) or false (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this field. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE or True, which means no permissions assigned and field is visible.
<b>Width</b>	Optional	Width to be taken up by the Rich Text Editor; can be in integer which is in pixels, or as a percentage (for example, "30%") of the page width or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Notes

- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- If the Rich Text Editor is parented by a Grid Layout, the label orientation of the Rich Text Editor follows Grid Layout's **Label Orientation** property.
- If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment.
- If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Row Spacer

Row Spacer element is a child element of the Grid Layout. This element inserts a blank row(s) within a Grid Layout that can span multiple rows. Place this Row Spacer element anywhere in the Grid Layout to create a blank row between it and its sibling.



### Summary

references Variable	No
references Method	Yes
references Form	No
Allowable Child Elements	<i>None</i>

### Properties

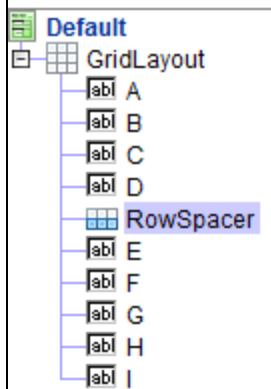
Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Row Spacer.
Cell Row Span	Optional	Defines how many rows the Row Spacer will span. Default is one (1) row.
Style	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Row Spacer such as positioning and color. Default is NONE.
Tooltip	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers over the Row Space. Default is empty, which means that a tooltip has not been defined.
Tooltip Style	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the row space. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
Tooltip Width (px)	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
On Enter	Optional	The system calls any script defined in this property. This method is invoked when the Row Spacer is focused on the Web page and the user presses the ENTER key.

### Notes

Row Spacer must be parented by a Grid Layout.

Below is an example to the Row Spacer Element that spans 2 rows. The white box within the Preview Pane is the Row Spacer defined.

## Element Tree Pane

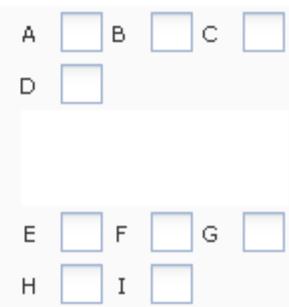


## Element Properties Pane

Name	Value
Cell Row Span	2
Name	RowSpacer
Style	CwRow1
Tooltip	

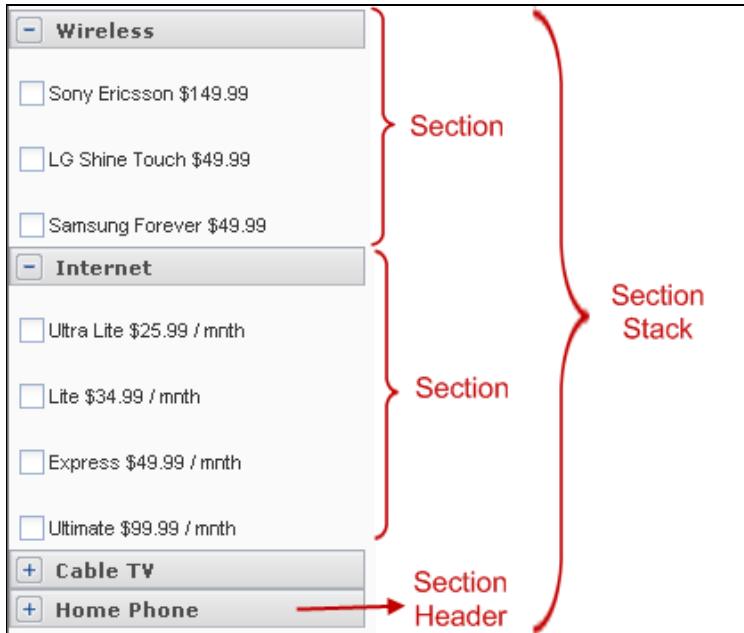
The grid layout displays two empty rows defined by the RowSpacer element.

## Preview Pane



## Section

Section is an element that is contained within a Section Stack. The user is able to divide groups of data/form elements into logical blocks that share the same "real estate" on a page. These Sections can be collapsed and expanded at runtime for readability and ease of use. Section Header is a child element of the Section element and it is used to facilitate the expand and collapse functionality of Sections. Below is an example of expanded and collapsed Sections within a Section Stack.



## Summary

references Variable	Yes
references Method	Yes
references Form	Yes
Allowable Child Elements	Button Chart Checkbox Date Field Dynamic Document Dynamic Table Form Frame Grid Layout HTML Content Header Horizontal Layout Hyperlink Image Label Large Text Large Text Area Layout Spacer Menu Item Password Field Radio Button Group Reference Field Section Header Section Stack Select Field Separator Table Tabset

Text Area
Text Field
Translation
Tree
Upload File
Variable
Vertical Layout

The following table describes the properties of section element:

Property	Mandatory/Optional	Comment
<b>Name</b>	Mandatory	Name of the Section.
<b>*Cell Alignment</b>	Optional	<p>The cell alignment property works in conjunction with the height/width properties of the parent Section Stack. The values in the height/width properties must be set for the cell alignment property to function.</p> <p>This property specifies the horizontal and vertical positioning of the children form elements of the Section. An undefined value results in a "top-left" alignment.</p>
<b>Dynamic Tooltip</b>	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Error Icon Gap</b>	Optional	<p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Expand</b>	Optional	This property dynamically expands or collapses a Section element. Choose from a permission method, or <b>TRUE / FALSE</b> option to assign the Section element to an expanded or collapsed state. The Default setting is True which means expanded state. This property does <b>not</b> need to work with a Section Header. If this property is set to False then the Section element and its children are hidden.
<b>Form</b>	Optional	<p>Choose from a list of available Forms which this Form Frame shall present. If the Form Frame is assigned to a Variable property, the choices are limited to the Forms of that Variable. If Variable property is not assigned, the choices are limited to Forms in the current User Interface. Default is not assigned, where no Form is displayed. Alternatively, a local string variable can be used to dynamically change the form to be used at runtime. This string variable should have ONLY the name of the form such as "Default". Setting this property to a string variable will automatically use the user interface of the Variable property to look for the form if it is provided, otherwise it will use the current user interface for the form lookup.</p> <p><b>Note:</b> If metadata is migrated from different version 5.x releases and property is set to the <i>Form.Default</i>, the migration process will reset the value to <i>Default</i>.</p> <p>Please refer to the <a href="#">Form Property</a> example below for more details.</p>
<b>Header</b>	Mandatory	Choose from a list of Finders or Finder View Names. Refer to <a href="#">Finders</a> for more details.
<b>Icon Height</b>	Optional	Height of the icon used or consumed within the element. The value is defined as the number of pixels (integer) or percentage. The default value is 0.
<b>Icon Tooltip</b>	Optional	Click the ellipsis button and specify a tooltip description within the <b>Value</b> field for the icon.
<b>Icon Width</b>	Optional	Width of the icon used or consumed within the element. The value is defined as the number of pixels (integer) or percentage. The default value is 0.
<b>On Enter</b>	Mandatory	The system calls any script defined in this property. This method is invoked when the Section is focused on the Web page and the user presses the ENTER key.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>State ChangeCallBack</b>	Mandatory	Choose from a list of methods that is called when the user expands or collapses a Section at runtime. This method is called when the user clicks the expand/collapse image of the Section Header. For example, Velocity Studio may choose to use this property to control the expand/collapse state of multiple sections.
		Please refer to the <a href="#">State Change Callback property</a> example below for more details.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the

		styling of the Section such as positioning and color. Default is NONE.
<b>Tooltip</b>	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers over the Section. Default is empty, which means that a tooltip has not been defined. During runtime, the tooltip may be difficult to find as it is not the only layout on the screen and may cover a large area.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the section. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Variable</b>	Optional	Choose from the list of Variables of type <i>User Interface</i> to bind to this Section. Once a Variable is chosen, you can choose a Form to present the Variable in the Form property. If Form property is not selected with a chosen Variable, the Form with name <i>Default</i> (that is, <Variable>.Form.Default) is assumed. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.

## Notes

Section can have many child elements (see list above). In the absence of a layout element, the child elements are displayed vertically within the section from top-to-bottom.

A section can work in a similar way to a Form Frame. The Variable and Form properties of the Section element can be set up to function like a Form Frame, where the Section's appearance may change at runtime based on Variables. Please refer to the [Form Frame](#) section for further details.

The Expand property of the Section element will function without a Section Header. The Section and its child elements will simply not be visible if the Expand property is set to False.

Width of the Section is dictated by the Section Stack's width or the width of the layout elements (Horizontal or Vertical).

There can only be one (1) Section Header within a Section.

\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## State Change Callback Example

The **State Change Callback** property is called when the user clicks the expand/collapse icon on the Section Header element. The method assigned to this property can be coded to dynamically expand/collapse other sections at runtime. For example, the developer creates two Section elements: *SectionA* and *SectionB* and codes the following methods and permissions such that *SectionB* is expanded when *SectionA* is expanded:

1. A variable *isExpandedB* is created and initialized as *False* (onInit method).
2. Two methods are created:
  - o a new script method is created called *ExpandB* that sets the variable *isExpandedB* to *True*
  - o permissions method *PermB* returns the variable ("return this.*isExpandedB*").

Permission Script for PermB		Script for Expand	
Script:	<pre>function PermB() { // Metadata type method. Can be called by scripts.     return this.isExpandedB; }</pre>	Script:	<pre>function Expand() { // Metadata type method. Can be called by scripts.     //this.default.Section.Expand = isExpanded;     this.isExpandedB = true; }</pre>
Return:	com.conceptwave.system.Boolean		
Privileges:	<input checked="" type="checkbox"/> Privileges and participant operations <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Add Shared Favorites</li> <li><input checked="" type="checkbox"/> Administration App</li> <li><input checked="" type="checkbox"/> ConceptWave API Access</li> <li><input checked="" type="checkbox"/> Delegate Task</li> <li><input checked="" type="checkbox"/> Delete Shared Favorites</li> <li><input checked="" type="checkbox"/> Everyone</li> <li><input checked="" type="checkbox"/> Get Available</li> <li><input checked="" type="checkbox"/> Group Manager</li> <li><input checked="" type="checkbox"/> Process Manager Administrator</li> <li><input checked="" type="checkbox"/> Return Task</li> <li><input checked="" type="checkbox"/> Runtime Administrator</li> <li><input checked="" type="checkbox"/> Set Process Priority</li> <li><input checked="" type="checkbox"/> Show Error Details</li> <li><input checked="" type="checkbox"/> Take on Task</li> <li><input checked="" type="checkbox"/> User Profile Administrator</li> <li><input checked="" type="checkbox"/> Workgroup Select</li> <li><input checked="" type="checkbox"/> Worklist Administrator</li> </ul>		

3. ExpandB is assigned to the State Change Callback property of SectionA and PermB is assigned to SectionB Expand property.

Section A Properties		Section B Properties																																																					
Default <ul style="list-style-type: none"> <li><input type="checkbox"/> SectionStack                     <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> SectionA                             <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> SectionA Header</li> <li><input checked="" type="checkbox"/> DateField1</li> </ul> </li> <li><input checked="" type="checkbox"/> SectionB                             <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> SectionB Header</li> <li><input checked="" type="checkbox"/> DateField</li> </ul> </li> </ul> </li> </ul>	<table border="1"> <thead> <tr> <th>Name</th> <th></th> </tr> </thead> <tbody> <tr> <td>Cell Alignment</td> <td></td> </tr> <tr> <td>Expand</td> <td></td> </tr> <tr> <td>Form</td> <td></td> </tr> <tr> <td>Header</td> <td></td> </tr> <tr> <td>Name</td> <td>SectionA</td> </tr> <tr> <td>On Enter</td> <td></td> </tr> <tr> <td>Show Label</td> <td>true</td> </tr> <tr> <td>State Change CallBack</td> <td>Expand</td> </tr> <tr> <td>Style</td> <td></td> </tr> <tr> <td>Tooltip</td> <td></td> </tr> <tr> <td>Variable</td> <td></td> </tr> <tr> <td>Visible</td> <td></td> </tr> </tbody> </table>	Name		Cell Alignment		Expand		Form		Header		Name	SectionA	On Enter		Show Label	true	State Change CallBack	Expand	Style		Tooltip		Variable		Visible		Default <ul style="list-style-type: none"> <li><input type="checkbox"/> SectionStack                     <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> SectionA                             <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> SectionA Header</li> <li><input checked="" type="checkbox"/> DateField1</li> </ul> </li> <li><input checked="" type="checkbox"/> SectionB                             <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> SectionB Header</li> <li><input checked="" type="checkbox"/> DateField</li> </ul> </li> </ul> </li> </ul>	<table border="1"> <thead> <tr> <th>Name</th> <th></th> </tr> </thead> <tbody> <tr> <td>Cell Alignment</td> <td></td> </tr> <tr> <td>Expand</td> <td>PermB</td> </tr> <tr> <td>Form</td> <td></td> </tr> <tr> <td>Header</td> <td></td> </tr> <tr> <td>Name</td> <td>SectionB</td> </tr> <tr> <td>On Enter</td> <td></td> </tr> <tr> <td>Show Label</td> <td>true</td> </tr> <tr> <td>State Change CallBack</td> <td></td> </tr> <tr> <td>Style</td> <td></td> </tr> <tr> <td>Tooltip</td> <td></td> </tr> <tr> <td>Variable</td> <td></td> </tr> <tr> <td>Visible</td> <td></td> </tr> </tbody> </table>	Name		Cell Alignment		Expand	PermB	Form		Header		Name	SectionB	On Enter		Show Label	true	State Change CallBack		Style		Tooltip		Variable		Visible	
Name																																																							
Cell Alignment																																																							
Expand																																																							
Form																																																							
Header																																																							
Name	SectionA																																																						
On Enter																																																							
Show Label	true																																																						
State Change CallBack	Expand																																																						
Style																																																							
Tooltip																																																							
Variable																																																							
Visible																																																							
Name																																																							
Cell Alignment																																																							
Expand	PermB																																																						
Form																																																							
Header																																																							
Name	SectionB																																																						
On Enter																																																							
Show Label	true																																																						
State Change CallBack																																																							
Style																																																							
Tooltip																																																							
Variable																																																							
Visible																																																							

### Form Property Example

The Form property can be assigned to a variable of the type *string* which represents a Form to be displayed at runtime. There are instances when the metadata developer may want the form that is displayed within the Section to change based on runtime conditions or actions (that is, a button being clicked changes the value of this variable allowing a different form to be displayed). The example below will show how the Section will display *FormA* via a string variable *STest*:

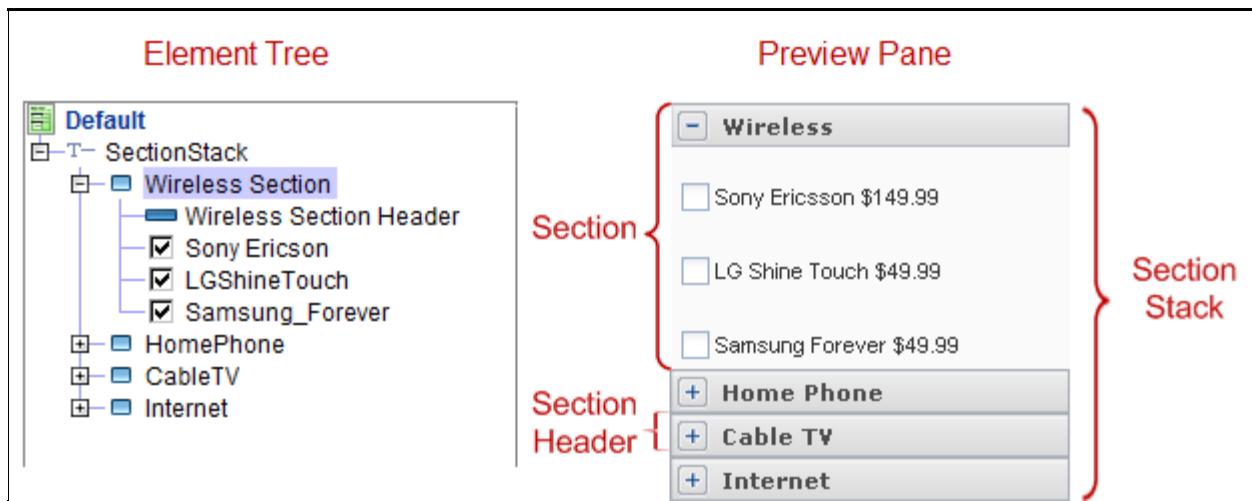
1. Create a variable called "STest" of the data type "string" (example a local variable under the UserInterface object where the Section is defined).
2. Assign *STest* to the Form property of the Section.
3. Initialize the *STest* variable in the onInit method (*this.STest = "FormA";*).
4. Create a new form called *FormA* (note: form name is the same as the string name - as the string refers to the form).

## Section Stack

Section Stack is a container that allows for the grouping of data or elements on a page in a logical fashion. When a page contains "too much stuff", the user is distracted. The Section Stack element will allow data or elements to appear in more digestible chunks. A Section Stack may comprise of individual Sections that share the same "real estate" on a page and which can be collapsed/expanded at runtime as shown in the example below.

You can have multiple Sections under a Section Stack, and have it display only one of those sections at a particular time, by setting Visibility Mode property of the Section Stack to *mutex*. Setting the Visibility Mode to *multiple* will allow multiple sections to be visible simultaneously during runtime.

The example below shows the relationship between the Section Stack, Section and Section Header elements.



## Summary

references Variable?	No
references Method?	No
references Form?	No
Allowable Child Elements	Section Iterator

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Section Stack.
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Section Stack in the page or parent Element. Default is undefined, which behaves as if it is top-left.
Header Style	Optional	This property allows you to select styles from your stylesheet for the section stack element. The style set for this property applies to the style of the entire section stack. To apply the header style to only the headers, set the style at the section header element level.
Height	Optional	Height used or consumed by the Section Stack. The value is defined as the number of pixels (integer), or as a percentage of page height or of the parent element.
Width	Optional	Width used or consumed by the Section Stack. The value is defined as the number of pixels (integer) or as a percentage of page width or of the parent element. For example, 30% of the page width or 150 pixels.
		The system calls any script defined in this property. The onEnter property is only met if

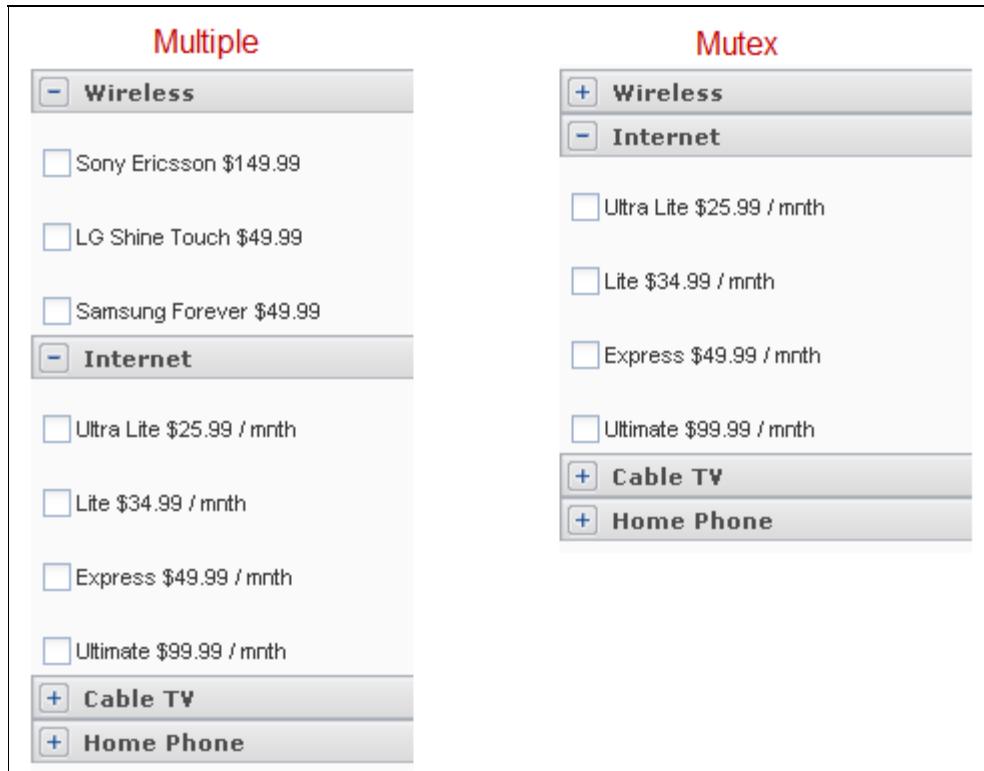
<b>On Enter</b>	Optional	there is an input field (for example, a textfield, selectfield, and so on) that has focus when the <b>Enter</b> key is pressed. This onEnter property is invoked only if the input field does not have an OnEnter property set. This feature can be used with multiple input fields, but only one onEnter method to fire for all fields. For Finders, the search input fields by default use the onEnter method of the sectionstack, which calls on the finder's search action.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Section Stack such as positioning and color. Default is NONE.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
<b>Visibility Mode</b>	Optional	<p>Determines how Sections in the Section Stack share the Session Stack area.</p> <ul style="list-style-type: none"> <li><b>mutex:</b> Only one Section within a Section Stack is expanded; expanding a Section collapses all other Sections in the Section Stack.</li> <li><b>multiple:</b> Multiple Sections in the Section Stack can be expanded without restrictions.</li> </ul> <p><b>Note:</b> This property is dependent on the <b>Expand</b> property of the Section element. The Section element must have the expand property set to True for the Sections to be collapsed or expanded at runtime.</p>

## Notes

The Section Stack works in conjunction with Sections and Section Headers to create sections of content that can be organized to share the same "real estate" on the same page. Use Section Stack to define a fixed area ("a pane") and add/organize content in Sections that may otherwise overwhelm the page.

Section Stack is also useful in hiding detailed content, by placing it in a Section that is collapsed by default.

Section Stack operates in one of two modes: in *mutex* mode, only one Section is visible at a time (similar to Microsoft Outlook's left-hand Navigation Pane), or in *multiple* mode, where Sections can be expanded/collapsed without restriction. The figure below displays the two options for the Viability Mode property.

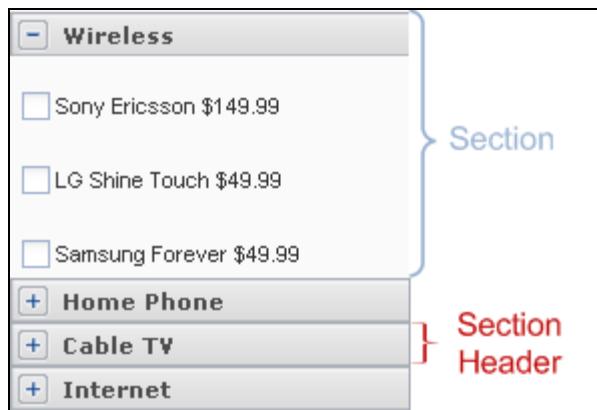


If a border is desired for the Section Stack, [create a style in the CSS](#) that declares a border-width, and assigns it to the Section Stack.

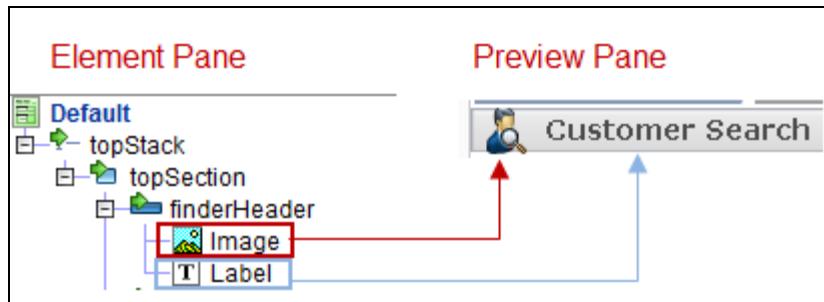
\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Section Header

A Section Header creates the header included in the user interface of a Section. It is used for design and contains limited or no functionality. The example below shows a Section Stack that contains an expanded "Wireless" Section and its associated Section Header (that is, Wireless) as well as collapsed Sections (Home Phone, Cable TV and Internet) and their associated Section Headers.



The Section Header is simply a banner that can contain child field elements such as an image and/or label as shown below. All form elements that are children of the Section Header will overlap the banner.



## Summary

references Variable	No
references Method	No
references Form	No
	Grid Layout Horizontal Layout Vertical Layout Layout Spacer Form Frame Button Checkbox Date Field Header Hyperlink Image Label
Allowable Child Elements	Password Field Radio Button Group Reference Field Select Field Separator Text Field Translation Upload File

Variable
HTML Content
Iterator
Menu Item
Dynamic Document

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Section Header.
Label	Mandatory	Visible label of the Section Header. Displayed over the grey banner in a left-justified alignment. Default is empty.
Icon	Optional	Select an icon graphic from the graphic resources that is displayed over the banner in a left-justified alignment. The system requires three image files <image name>.jpg, <image name>_opened.jpg and <image name>_closed.jpg that are called based on the expanded state of the section. Select the <image name>.jpg image for this property. The default image used is opener.png.
On Enter	Optional	The system calls any script defined in this property. This method is invoked when the Stack Header is focused on the Web page and the user presses the ENTER key.
Style	Optional	Choose from styles within the selected style sheet (located in the Velocity Studio toolbar), which defines the styling of the Section Header such as background color. Default is NONE.
Expand	Optional	The system determines whether the child field elements of the Section appear by default in a collapsed or expanded state at runtime.

## Notes

Child Elements of a Section Header are placed horizontally, and appear in a right-justified alignment moving left within the Header. Child Elements with a Section Header is a useful feature and allows the user the ability to add action fields for a Section - such as buttons - that can implement methods for the action.

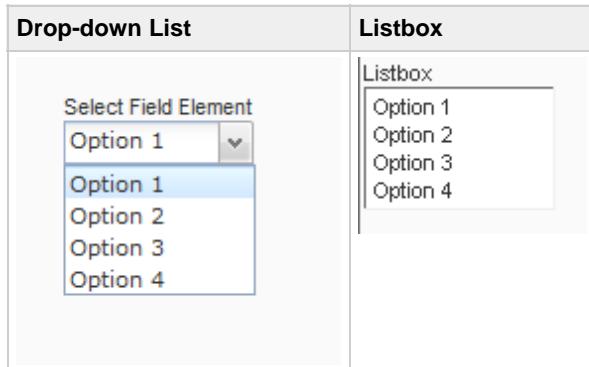
Header Width: The width of the header is based on the Width property of the parent Section Stack element or the layout directive (for example, Horizontal or Vertical elements). The Height and Width properties must be defined for the parent Section Stack element or layout directive elements for the header to be displayed.

The Section Header can be used to expand or collapse Section Stacks containing child field elements by clicking the Section Header. This is accomplished by setting the Expand property of the Section Header to TRUE. There is the ability to have multiple Sections defined within a Section Header and display these Sections simultaneously or one Section at a particular time. The ability to display multiple expanded Sections simultaneously or restrict the expansion of just one Section at a time is determined by the Visibility Mode parameter of the parent Section Stack element. By setting the Section Stack Visibility Mode parameter equal to *mutex*, then only one section will be displayed at a particular time and the other Sections collapse. Setting the Visibility Mode value equal to *multiple*, will allow for all selected Sections to be open simultaneously.

Icon: The graphical icon that is displayed in the header needs to contain three image files: <image name>\_opened.jpg, <image name>\_closed.jpg and <image name>.jpg. The expand/collapse state of the Section will dictate which image is displayed. For the Icon property, the <image name>.jpg must be selected. All three images must be placed in the <project metadata>/resources directory for selection and use. If an icon image is not selected in the Icon property then the system will use the opener.png image which is located in the skin directory (by default Enterprise).

## Select Field

This element can create a listbox or a drop-down list for a list of values defined as an enumeration variable. Select field can be used in top level user interface with its variable property set to the user interface's variable of type enumeration. The **Height** property determines the appearance of the box as either a listbox or a drop-down list. This element also allows for multiple selections to be made at runtime.



The following table describes the properties for select field:

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Select Field.
Label	Mandatory	Visible label of the Select Field. If empty, there is no Select Field label. Default is empty.
Variable	Optional	Choose from a list of variables of type string, integer, or boolean. A variable with enumeration values can be used in top level user interface with this variable property set to the user interface's variable of type enumeration (static only).
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Select Field in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Select Field occupies in the Grid Layout, to the right of the current cell. Default is empty, which means it occupies one column.
Cell Row Span	Optional	<p>This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
Double Click Method	Optional	The method assigned to this property is invoked at runtime whenever the user double clicks a value in the Select Field. The method can be a User Action Method or a Script. This method is only functional in Internet Explorer.
Dynamic Label	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
Dynamic	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the

<b>Label Style</b>		Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
<b>Dynamic Tooltip</b>	Optional	<p>Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i>. The static Tooltip property is considered as default until the action returns a value.</p> <p>When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.</p>
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Editable</b>	Optional	<p>This property determines whether the Select Field is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be enabled or disabled). A <i>False</i> still enables the user to view the list box and its values at runtime, but the user is unable to select one of the options from the list box.</p> <p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.</p>
<b>Empty Value Text</b>	Optional	This property defines what text is displayed inside the select list. By default, this property takes an empty value, which displays as a blank field. You can configure the text in this property to suit your needs.
<b>Enumeration Reference</b>	Optional	<p>This property provides a list of all data types within the metadata that have either an enumeration or a reference finder defined.</p> <p>At runtime, when the form is displayed, the value is still retrieved from the <b>Variable</b> property. The variable set on the fields can be primitive strictly for storing values. When you click the reference icon, the product uses the enumeration reference data type to find the finder user interface to display in a dialog.</p> <p>See the <a href="#">example</a> on how to use this property for details.</p>
<b>Error Icon Gap</b>	Optional	<p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Group By In Table</b>	Optional	This property allows the user to group the Finder results by the elements value.
<b>Height</b>	Optional	Height to be taken up by the Select Field area as defined as an integer value in pixels. This property value will determine if a listbox or a drop down list appears. This property defaults to a drop down list. Any value entered for this property will result in a listbox. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>Height in Lines</b>	Optional	Specify the height to be taken up by the Select Field area by lines. The default value for this field is 0.
<b>Icon</b>	Optional	Choose an image file that appears as an icon to the right of the Select Field. Default is

		NONE.
Include in Tab Order	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
Label Align	Optional	<p>Click the <b>Value</b> field and select from one of the following alignment options for your label:</p> <ul style="list-style-type: none"> <li><b>Left</b></li> <li><b>Centre</b></li> <li><b>Right</b></li> </ul>
Label Orientation	Optional	Specifies where the label of the Select Field appears in relation to the Select Field area; can be <i>left</i> , <i>right</i> or <i>top</i> of the Select Field area. The default orientation is <i>right</i> . However, in a Grid Layout, the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout..
Label Style	Optional	Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the Label such as font color. Default is NONE.
Multiple Selection	Optional	Only applicable when Select Field is a listbox. When true, allows the user to select multiple items in the listbox by holding the Ctrl key. Default to <i>False</i> , which does not allow multiple selections.
On Enter	Optional	The system calls any script defined in this property. This method is invoked when the Select Field is focused on the Web page and the user presses the ENTER key.
Picklist Style	Optional	<p>This property provides a means of setting the style of a dropdown list for a Select Field element, which includes the font style, colour, size, and style of the box.</p> <p><b>Note:</b> For the assigned style, the following are appended on changes of state:</p> <ul style="list-style-type: none"> <li><b>Dark</b></li> <li><b>Over</b></li> <li><b>OverDark</b></li> <li><b>Selected</b></li> <li><b>SelectedDark</b></li> <li><b>SelectedOver</b></li> <li><b>SelectedOverDark</b></li> </ul>
Remove Label Line	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
Run Trigger	Optional	<p>The method assigned to this property is invoked whenever the value of the Select Field changes. The method can be a User Action Method or a Script.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>You click a menu.</li> <li>You change your selection in a navigation tree.</li> <li>You click another tab of a tabset.</li> </ul> <p><b>Notes:</b></p>

		<ul style="list-style-type: none"> <li>Field triggers do not fire on save. However, document triggers do.</li> <li>As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
<b>Search Select</b>	Optional	Select the <b>Search Select</b> checkbox for the Select Field element. At runtime, typing in the select item (for example, <code>call</code> ) shows a list of values that currently match the characters that are entered. The list displayed for the drop-down is incrementally altered each time a new character is entered.
<b>Show Empty Values</b>	Optional	This property determines whether to display the first value in the drop-down as empty or only show the values which come from the enumeration or the reference data type values. Default is <code>True</code> which means that the first value displayed is empty.
<b>Show as Advanced</b>	Optional	When you select this property, the constructor used for the UI object becomes a custom widget. See <a href="#">Multiselect UI Element</a> for details.
<b>Show Label</b>	Optional	This property determines whether the label is displayed on the Select Field. Choose from <code>True</code> or <code>False</code> . Default is <code>True</code> which means that the label is displayed.
<b>Sorted</b>	Optional	Defaults to false. Boolean to determine whether to alphabetically sort the enumeration in the Select field, or remain in the same order as defined in <b>Enumerations</b> tab of the Variable's Data Type.
<b>Start Row</b>	Optional	Applicable only when parent Element is a Grid Layout. If true, the Select Field starts in a new row of the grid. If false, it is simply placed in the current cell as-is. Default is <code>false</code> , which does not start a new row.
<b>Style</b>	Optional	Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Select Field such as color and font. Default is <code>NONE</code> .
<b>Textbox Style</b>	Optional	<p>Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the listbox of the Select Field such as color. Default is <code>NONE</code>.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <code>Disabled</code> suffix appended to the assigned style. As example, when assigning the <code>MyStyle</code> style to the <b>Textbox Style</b> property, the stylesheet must have <code>MyStyleDisabled</code> to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <code>fieldDisabled</code> style to your stylesheet.</p> <p><b>Note:</b> The <code>fieldDisabled</code> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the <b>Select</b> field. Default is empty, which means no tooltip for the <b>Select</b> field.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the <b>Select</b> field. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Visible</b>	Optional	Determines whether the Select Field is visible or not. Choose <code>true</code> (visible) or <code>false</code> (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this field. Pre-defined system <a href="#">Permissions</a> such as <code>addPerm</code> , <code>deletePerm</code> , <code>editablePerm</code> , <code>attachPerm</code> , <code>executePerm</code> , and <code>visiblePerm</code> can be assigned as well as Object Permissions. Defaults to <code>NONE</code> or <code>True</code> , which means no permissions assigned and field is visible.
<b>Width</b>	Optional	Width to be taken up by the Select Field area; can be in integer which is in pixels, or as a percentage (for example, "30%") of the page width or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Notes:

- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- The Select Field can be presented as a regular listbox or a drop down listbox, using the *Height* property. In regular listbox, multiple selection may be possible (controlled by *Multiple Selection* property). It is not possible to have multiple selection in drop down listbox.
- The Height property determines the height of the Select Field and determines the functionality - either a drop-down list or a listbox. Below are two examples of values for this property.

Drop-down List	List box
No Height Specified 	Height Specified 
<i>Height</i> Property = has no value (default)	<i>Height</i> Property = 50 pix

- You should always assign a Variable with enumeration defined to a Select Field. Here are the setup steps:
  - A. Set up an enumeration data type for a Document:
    1. Create a new Data Type that Extends from String, Boolean or Integer: <namespace> > Data Dictionaries > Data Type > New Data Type (For example, Name = EnumString)
    2. Enter enumeration values in the Enumeration Tab for selection at runtime.
  - B. Create a Select Field element in a Document User Interface
    1. Create a User Interface Document: <namespace> > Data Dictionaries > Documents
    2. Add the Variable (EnumString) created above to the Document: Navigate to the Variables tab and click the  to add the variable.
    3. In the UserInterface, override the Default Form
    4. Add the Form Element *Select Field*
    5. Assign the enumerated variable added in step 2 to the **Variable** property.
- If Select Field is parented by a Grid Layout, the label orientation of the Select Field follows Grid Layout's **Label Orientation** property.
- For enumeration sorting, the **Sorted** property is set to false by default. This means the enumeration is displayed in the order as defined in the **Enumerations** tab of the Variable's Data Type. To automatically sort in runtime, set the **Sorted** property to true.

\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

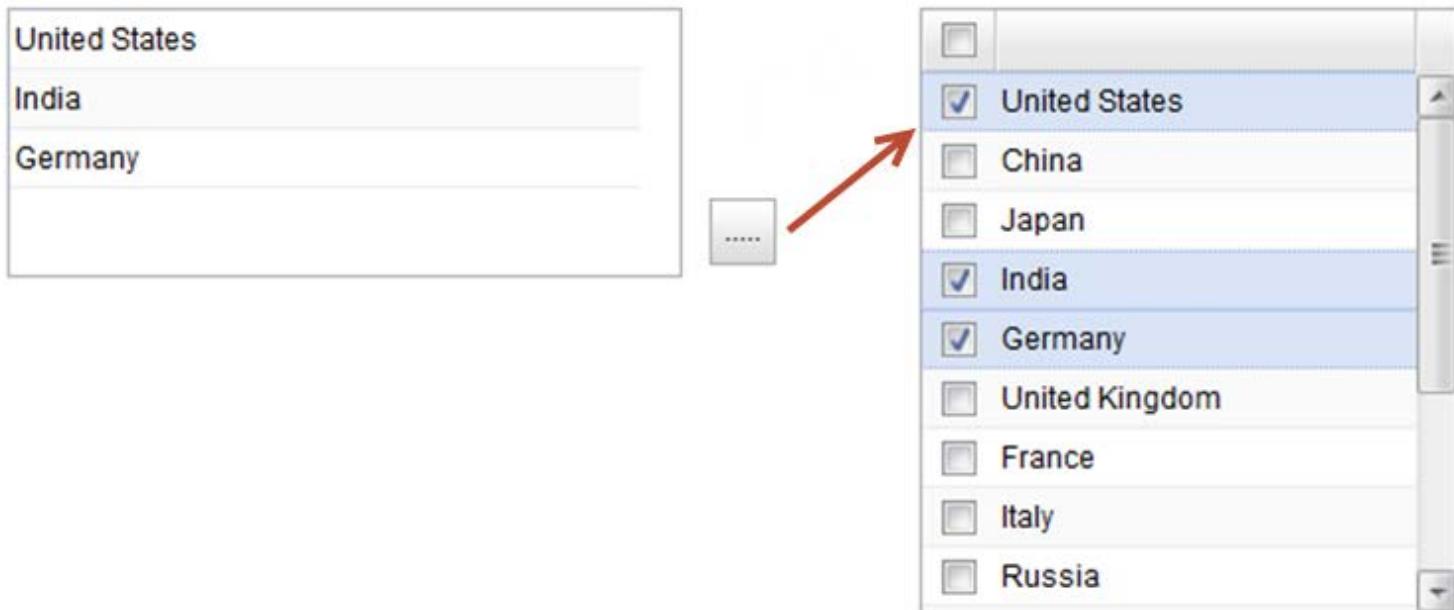
- Select Field display values are interpreted as HTML. If you use an ampersand (&) that requires translation, use & instead to avoid the & character being treated as an escape sequence, which may cause the remaining characters after the ampersand to disappear.

## Multiselect UI Element

The multiselect UI element uses the Select Field element's **Show as Advanced** property, which allows you to select a subset of a given list.

The multiselect UI element is a custom canvas item consisting of the following:

- One select item that shows the selected list
- An icon that, when clicked, shows another select item containing the full list



This custom element takes care in selecting values that are already selected. You can select or deselect values using the checkbox on each row. To deselect all values, click the master checkbox in the header.

Both the **Save** and **Cancel** buttons appear under the select item. When you click the **Save** button, the selected values are formed into a new value map and stored in the selected list. When you click the **Cancel** button, no changes occur.

The following are restrictions for the multiselect element:

- The element only accepts a data type with the **System DB Enum** property set.
- The element only accepts String, and if a validation error occurs if the comma-separated value returned to the server has a length smaller than the length specified on the data type.
- The element shifts any UI to its right when you click the expand button.

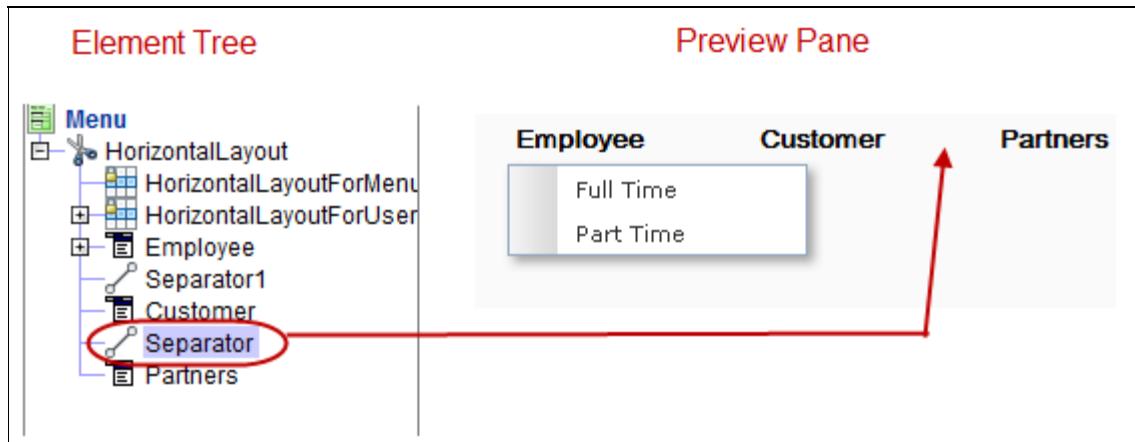
To use the multiselect UI element, do the following:

1. Create a dynamic code table data type and ensure the following:
  - There is a **System DB Enum** specified on the data type
  - The code table contains data
2. Use this data type as a document's variable type.
3. In the form, add a new Select element and bind it to the variable that you have created.
4. Set the Select element's **Show as Advanced** property to true by selecting its checkbox.

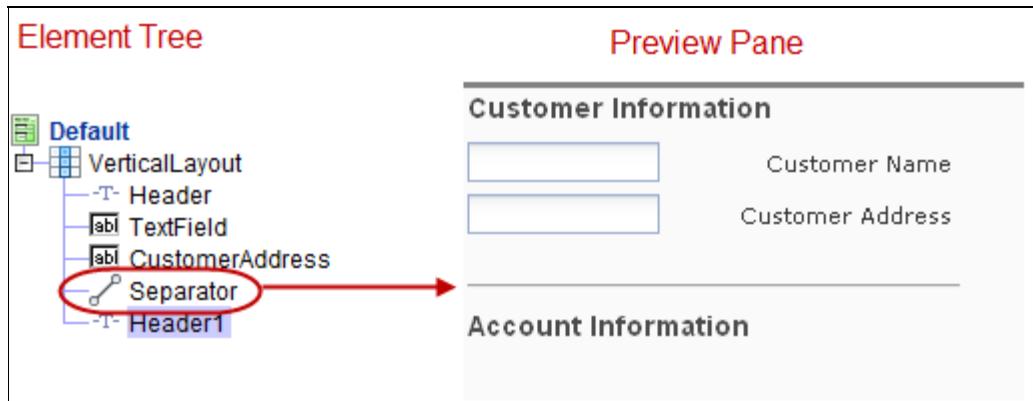
## Separator

Separator is a static item that appears as a horizontal separator, that does not respond to mouse events. It is used to organize user interfaces that may otherwise appear to be overwhelming to the users. The Separator is parented by layout directive elements (Horizontal, Vertical and Grid) as well as the Menu and Section elements. For the layout directive and the Section elements, the Separator places a thin horizontal line in the display to separate elements and is often more efficient use of space than using white space. Within the Menu Form of the Application User Interface, the system places a white space in the place of the Separator. Below are examples of the Separator element used within the Menu and Vertical layout directives.

### Separator with a Menu structure:



### Separator within a Vertical layout:



## Summary

references Variable?	No
references Method?	No
references Form?	No
Allowable Child Elements	None

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Separator
		Height to be taken up by the Separator; can be in integer which is in pixels, or in percentage (for example, "50%") of parent Menu. If this element is nested under a Table

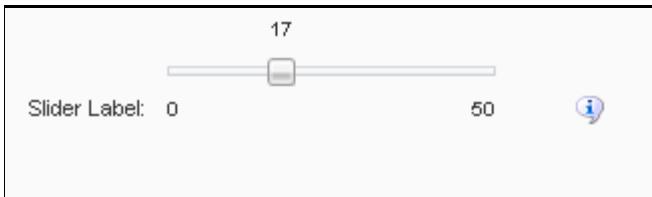
<b>Height</b>	Optional	element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>On Enter</b>	Optional	The system calls any script defined in this property. This method is invoked when the Separator is focused on the Web page and the user presses the ENTER key.
<b>Style</b>	Optional	Choose from styles within the selected style sheet (located in the Velocity Studio toolbar), which defines the styling of the Menu such as background color. Default is NONE.
<b>Label</b>	Optional	Visible label of the Separator. If empty, the label assumes the Name of the Separator. Default is empty.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
<b>Width</b>	Optional	Width to be taken up by the Separator; can be in integer which is in pixels, or in percentage (for example, "30%") of parent Menu. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Notes

- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- Create Separator between menu items to visually categorize them into different menu sections.

## Slider

The Slider element contains a GUI slider widget that allows you to select a numeric value from within a range by dragging a visual indicator up and down a track. You can change the value by clicking the track, or clicking and dragging the thumb.



To use the Slider element, you need to initialize the value of the integer variable to assign to the Slider element, and give it a **Maximum Value** and a **Minimum Value**.

### Summary

references Variable?	No
references Method?	Yes, Click Method and On Enter.
references Form?	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Slider element.
Label	Mandatory	Visible label of the Slider. If empty, the Slider has no label. Default is empty.
Visible	Mandatory	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Slider in the page or parent Element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Slider occupies in the Grid Layout, to the right of current cell. Default is empty, which occupy one column.
Cell Row Span	Optional	<p>This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>

<b>Dynamic Label</b>	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
<b>Dynamic Label Style</b>	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
<b>Dynamic Maximum Value</b>	Optional	Defines the maximum value of the slider that is set dynamically at runtime as a variable.
<b>Dynamic Minimum Value</b>	Optional	Defines the minimum value of the slider that is set dynamically at runtime as a variable.
<b>Dynamic Tooltip</b>	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Editable</b>	Optional	This property determines whether the element is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be enabled or disabled).
<b>Error Icon Gap</b>	Optional	<p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.</p> <p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Group By In Table</b>	Optional	This property allows the user to group the Finder results by the elements value. Default is <i>False</i> .
<b>Height</b>	Optional	This property allows you to decrease the slider element's height to 64 px. The value is defined as the number of pixels (integer) or percentage.
<b>Icon</b>	Optional	Choose an image file to appear as an icon to the right of the Slider. The default is none.
<b>Include in Tab Order</b>	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>• When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
		Specifies where the label appears in relation to the element area; can be <i>left</i> , <i>right</i> or <i>top</i>

<b>Label Orientation</b>	Optional	of the element area. The default orientation is . However, in a Grid Layout, the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout..
<b>Label Style</b>	Optional	Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Label such as font color. Default is NONE.
<b>Maximum Value</b>	Optional	Set this value to the maximum number that the user may choose. Default is zero (0).
<b>Minimum Value</b>	Optional	Set this value to the minimum number that the user may choose. Default is zero (0).
<b>Number of Available Values</b>	Optional	The number of values to show between the minimum and maximum. (default 1). This is the number of discrete values represented by the slider. If specified, the range of valid values (between minValue and maxValue) will be divided into this many steps. As the thumb is moved along the track it will only select these values and appear to jump between the steps.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Run Trigger</b>	Optional	<p>The method assigned to this property is invoked whenever the value of the Slider changes. The method can be a User Action Method or a Script.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>• You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>• You click a menu.</li> <li>• You change your selection in a navigation tree.</li> <li>• You click another tab of a tabset.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• Field triggers do not fire on save. However, document triggers do.</li> <li>• As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> <li>• The Run Trigger method only runs after the mouse button is released when changing the Slider element's value.</li> </ul>
<b>Show Label</b>	Optional	This property determines whether the label is displayed on the element. Choose from <i>True</i> or <i>False</i> . Default is <i>True</i> which means that the label is displayed.
<b>Start Row</b>	Optional	Applicable only when the parent element is a Grid Layout. If true, the element appears in a new row of the grid. If false, it is simply placed in the next available cell. Default is <i>False</i> , which means that the element does not appear on a new row.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Slider such as positioning and color. Default is none.
<b>Textbox Style</b>	Optional	<p>Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the text box within the Slider element such as color. Default is none.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the MyStyle style to the <b>Textbox Style</b> property, the stylesheet must have MyStyleDisabled to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
		Defines the tooltip text that is displayed in a hover box when the cursor hovers over the

<b>Tooltip</b>	Optional	element. Default is empty, which means that a tooltip has not been defined.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the element. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Variable</b>	Optional	Choose from the list of Variables available in the User Interface to bind to the Slider element. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
<b>Width</b>	Optional	Width to be taken up by the element; can be in integer which is in pixels, or in percentage (for example, 50%) of page width or parent element.

# Spinner

The Spinner element is a number spinner that allows the user to increase or decrease the number in the text box by clicking the up or down arrows. The element properties allow you to set a minimum and maximum number as well as the incremental value that the spinner steps when the user clicks the arrows.



## Summary

references Variable?	Yes, Dynamic Label, Variable
references Method?	Yes, Run Trigger, Dynamic Label Style, Visible.
references Form?	No
Allowable Child Elements	None

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Spinner element.
Label	Mandatory	Visible label of the Spinner. If empty, the Spinner has no label. Default is empty.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Spinner in the page or parent Element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Spinner occupies in the Grid Layout, to the right of current cell. Default is empty, which occupy one column.
Cell Row Span	Optional	<p>This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
Dynamic Label	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
Dynamic Label Style	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
		Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.

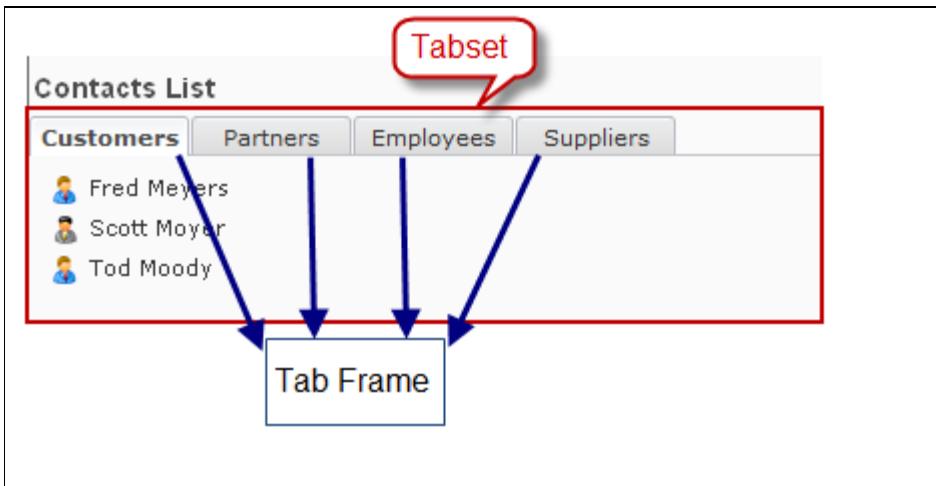
<b>Dynamic Tooltip</b>	Optional	When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
		This property determines whether the element is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be enabled or disabled).
<b>Editable</b>	Optional	<b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.
<b>Error Icon Gap</b>	Optional	Select this property's checkbox to make a gap for the validation icon. By default, this property is false.  <b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwf/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Group By In Table</b>	Optional	This property allows the user to group the Finder results by the elements value. Default is <i>False</i> .
<b>Icon</b>	Optional	Choose an image file to appear as an icon to the right of the text box. The default is none.
<b>Include in Tab Order</b>	Optional	This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values: <ul style="list-style-type: none"> <li>• When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.
<b>Label Orientation</b>	Optional	Specifies where the label appears in relation to the element area; can be <i>left</i> , <i>right</i> or <i>top</i> of the element area. The default orientation is <i>top</i> . However, in a Grid Layout, the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout..
<b>Label Style</b>	Optional	Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the Label such as font color. Default is NONE.
<b>Maximum Value</b>	Optional	Set this value to the maximum number that the user may choose. Default is zero (0).
<b>Minimum Value</b>	Optional	Set this value to the minimum number that the user may choose. Default is zero (0).
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field

		<p>in that row, this empty line is removed above the element.</p>
		<p>The method assigned to this property is invoked whenever the value of the Spinner changes. The method can be a User Action Method or a Script.</p>
		<p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>• You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>• You click a menu.</li> <li>• You change your selection in a navigation tree.</li> <li>• You click another tab of a tabset.</li> </ul>
<b>Run Trigger</b>	Optional	<p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• Field triggers do not fire on save. However, document triggers do.</li> <li>• As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
<b>Show Label</b>	Optional	<p>This property determines whether the label is displayed on the element. Choose from <i>True</i> or <i>False</i>. Default is <i>True</i> which means that the label is displayed.</p>
<b>Start Row</b>	Optional	<p>Applicable only when the parent element is a Grid Layout. If true, the element appears in a new row of the grid. If false, it is simply placed in the next available cell. Default is <i>False</i>, which means that the element does not appear on a new row.</p>
<b>Step</b>	Mandatory	<p>Set this value to the number that is incremented when the user clicks the arrow buttons on the spinner from the minimum value to the maximum value. Default is one (1).</p>
<b>Style</b>	Optional	<p>Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Spinner such as positioning and color. Default is none.</p>
<b>Textbox Style</b>	Optional	<p>Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the text box within the Spinner element such as color. Default is none.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the <i>MyStyle</i> style to the <b>Textbox Style</b> property, the stylesheet must have <i>MyStyleDisabled</i> to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	<p>Defines the tooltip text that is displayed in a hover box when the cursor hovers over the element. Default is empty, which means that a tooltip has not been defined.</p>
<b>Tooltip Style</b>	Optional	<p>Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the element. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.</p>
<b>Tooltip Width (px)</b>	Optional	<p>Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.</p>
<b>Variable</b>	Optional	<p>Choose from the list of Variables available in the User Interface to bind to the Spinner element. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.</p>
<b>Visible</b>	Optional	<p>Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.</p>
<b>Width</b>	Optional	<p>Width to be taken up by the element; can be in integer which is in pixels, or in percentage (for example, 50%) of page width or parent element.</p>



## Tabset

Tabset is a container that includes a set of Tab Frames that share the same space. Only the selected tab in the Tabset displays its contents within the frame, while the contents of the remaining tabs are hidden. The diagram below displays a Tabset consisting of 4 Tab Frames: Customers, Partners, Employees and Suppliers. The contents of the Customer tab (Tab Frame) is entirely displayed in the frame of the Tabset, while the remaining Tab Frames (Partners, Employees and Suppliers) are hidden?



## Summary

references Variable?	No
references Method?	No
references Form?	No
Allowable Child Elements	Tab Frame Iterator

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Tabset.
Height	Optional	Height to be taken up by the Tabset; can be in integer which is in pixels, or in percentage (for example, "50%") of page height or parent Element. Tab Frames in the Tabset are confined to this height.
Include in Tab Order	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"><li>• When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li><li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li><li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li></ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
On Enter	Optional	The system calls any script defined in this property. The onEnter property is only met if there is an input field (for example, a textfield, selectfield, and so on) that has focus when the <b>Enter</b> key is pressed. This onEnter property is invoked only if the input field does not

		have an OnEnter property set. This feature can be used with multiple input fields, but only one onEnter method to fire for all fields.
<b>Style</b>	Optional	Choose from styles within the selected style sheet (located in the Velocity Studio toolbar), which defines the styling of the Tabset such as background colour. Default is NONE.
<b>Tab Orientation</b>	Optional	Renders the tabs either at the <b>top</b> and <b>left</b> of the frame area. Defaults to top. The left orientation is limited to only displaying the Tab Frame icon and does not display the label.
<b>Tab Selected Method</b>	Optional	This method is invoked when the user selects (clicks) a Tab within the Tabset. It can be used in conjunction with the Variable property to dictate the content of the tab displayed.
<b>Tooltip</b>	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers over the Section. Default is empty, which means that a tooltip has not been defined. During runtime, the tooltip may be difficult to find as it is not the only layout on the screen and may cover a large area.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the section. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Variable</b>	Optional	This variable is of the type <i>string</i> ; and keeps track of the title of the tab which is displayed. Similar to the Image Variable, it can be initialized, set and used at any time before and after the tab is displayed by any method within the local user interface. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
<b>Width</b>	Optional	Width to be taken up by the Tabset; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element. Tab Frames in the Tabset are confined to this width.

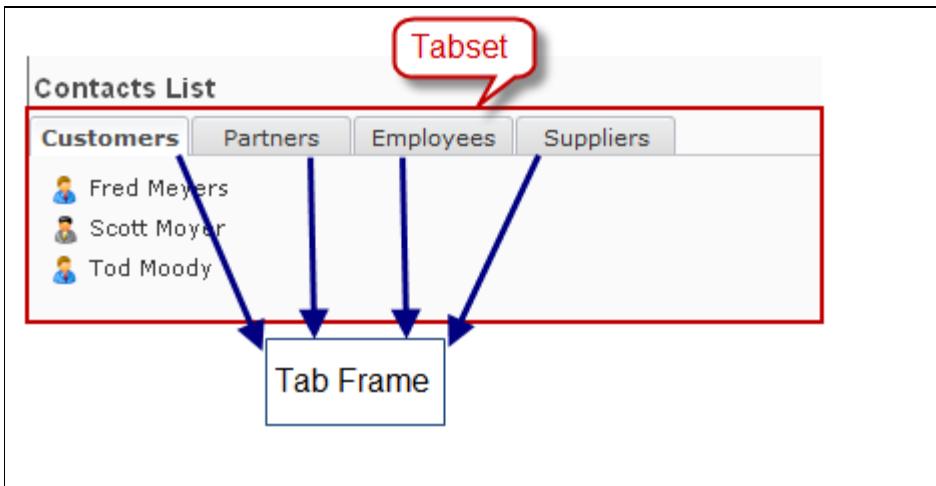
## Notes

With **Tab Orientation** property, the tabs can be placed at the top or at the left. A left orientation will not display the tab title, as the label cannot be drawn vertically.



## Tab Frame

A Tab Frame creates a tab in the Tabset. The example below displays four tabs or Tab Frames (Customers, Partners, Employees and Suppliers) within a Tabset. The Tab Frame can reference a Form via the Form or Variable property to display the contents of the Frame when the tab is selected. The Customer Tab Frame below references a form containing a listing of customer names and icons.



## Summary

references Variable?	Yes
references Method?	No
references Form?	Yes
Allowable Child Elements	<i>None</i>

The following table describes the properties of tab frame element:

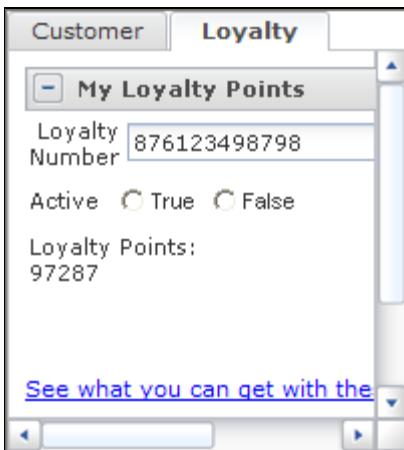
Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Tab Frame
Label	Mandatory	Visible label of the Tab Frame which appears in the tab bar. If this property value is empty, the label assumes the Name of the Tab Frame. Default is empty.
Dynamic Tooltip	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
Dynamic Tooltip Style	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (.css) name found in the stylesheet used by the application.
Error Icon Gap	Optional	Select this property's checkbox to make a gap for the validation icon. By default, this property is false.  <b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.
Error Icon Orientation	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
Form	Optional	Choose from a list of available Forms which this Form Frame shall present. If the Form Frame is assigned to a Variable property, the choices are limited to the Forms of that Variable. If Variable property is not assigned, the choices are limited to Forms in the

		<p>current User Interface. Default is not assigned, where no Form is displayed. Alternatively, a local string variable can be used to dynamically change the form to be used at runtime. This string variable should have ONLY the name of the form such as "Default". Setting this property to a string variable will automatically use the user interface of the Variable property to look for the form if it is provided, otherwise it will use the current user interface for the form lookup.</p> <p><b>Note:</b> If metadata is migrated from different version 5.x releases and property is set to the <i>Form.Default</i>, the migration process will reset the value to <i>Default</i>.</p>
<b>Icon</b>	Optional	If an icon is selected, the icon appears in the tab bar to the left of the label. Default is NONE.
<b>Icon Height</b>	Optional	Height of the icon used or consumed within the element. The value is defined as the number of pixels (integer) or percentage. The default value is 0.
<b>Icon Tooltip</b>	Optional	Click the ellipsis button and specify a tooltip description within the <b>Value</b> field for the icon.
<b>Icon Width</b>	Optional	Width of the icon used or consumed within the element. The value is defined as the number of pixels (integer) or percentage. The default value is 0.
<b>Image Variable</b>	Optional	Choose from a list of available <i>string</i> variables. This value of this variable determines the image that appears on the tab bar and its value is a URL or path of the image location. This variable can be initialized or dynamically set in any method within its local User Interface. The method should return a path to the image, for example "/cwf/Tree.gif"(return "/cwf/Tree.gif");. The image will be invoked whenever the tab is visible or when a page is being refreshed (on any user action). This property is dependent on the Form property and is only available if the Form property is given a value.
<b>Label Style</b>	Optional	Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Label within the Tab Frame such as color and font. Default is NONE.
<b>Label Variable</b>	Optional	Choose from a list of available <i>string</i> variables. This variable can be used to dynamically to set, change and use the title that is displayed on the tab bar. Similar to the Image Variable it can be initialized or dynamically changed by any method within the local user interface. This property is dependent on the Form property and is only available if the Form property value is given a value.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Show Label</b>	Optional	Dictates whether the label or title of the Tab is displayed. The default is "True" - which means the label is displayed.
<b>Style</b>	Optional	Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the Tab Frame such as color and font. Default is NONE.
<b>Tab Width</b>	Optional	Width to be taken up by the Tab; can be in integer which is in pixels, or in percentage (for example, "30%") of Tabset. This width is the width of the Tab (that is, Label) and not of the Tab Frame.
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Tab Frame. Default is empty, which means no tooltip for the Tab Frame.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the tab frame. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Validation Icon</b>	Optional	The Validation Icon property functions in conjunction with the <a href="#">Validation functionality</a> . There are four options for this property: Default (no validation icon), Static, State or Overlay. This property determines how icons or validation images (as a result of performing user interface validation on the TabFrame) appears on the Tab. A brief description of this property's values is described below and more details are described

		<p>in the <a href="#">Tab Errors</a> section of the <a href="#">User Interface Validation</a>.</p> <p><b>Default</b> - no images or icons appear on the Tab as a result of errors on User Interface validation.</p> <p><b>Static</b> - a predefined error (\resources\cwfv\error.png), warning (\resources\cwfv\warn.png) or information (\resources\cwfv\info.png) image will appear on the tab (to the right on the tab icon).</p> <p><b>State</b> - user defined images or icons replace the Icon when a validation error exists on the User Interface. The image files are an extension of the image defined in the Icon property. For example, if the Icon property is assigned the image, Icon.png, then the associated error, warning and information images are Icon_error.png, Icon_warn.png and Icon_info.png. These image files must be placed in the Resource folder of the metadata.</p> <p><b>Overlay</b> - the predefined error, warning and information icons will overlap the image defined in the Icon property when a validation error has been found on the TabFrame user interface.</p>
<b>Variable</b>	Optional	Choose from the list of Variables available in the User Interface to bind to this Tab Frame. Once a Variable is chosen, you can choose a Form to present the Variable in the Form property. If Form property is not selected with a chosen Variable, the Form with name <i>default</i> (that is, <Variable's data type>.UserInterface.Form.default) is assumed. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Determines whether the Tab Frame is visible or not. Defaults to NONE, which means visible.

## Notes

The height and width of the Tab Frame does not define the display area; it is the Tabset's height and width that fixates the dimension of the display frame. If the Tab Frame's height and width is larger than those defined in Tabset, scroll bars appear in the frame to scroll through Tab Frame's content.



If the Tab Frame references a Variable, the Variable's value is unknown at preview time. By default, the Preview Pane shows an empty Form of the Variable. At runtime, the presence of the Variable may change the appearance of the Form (for example, different size). Furthermore, in the event that the referenced Variable's data type is extended by another data type where its Form is also extended from the referenced (base) Form, the extended Form shall appear at runtime if the Variable assumes value of the extended data type.

Images or Icons can be added to the tab bar either statically via the Icon property or dynamically through the Image Variable property and a method.



\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

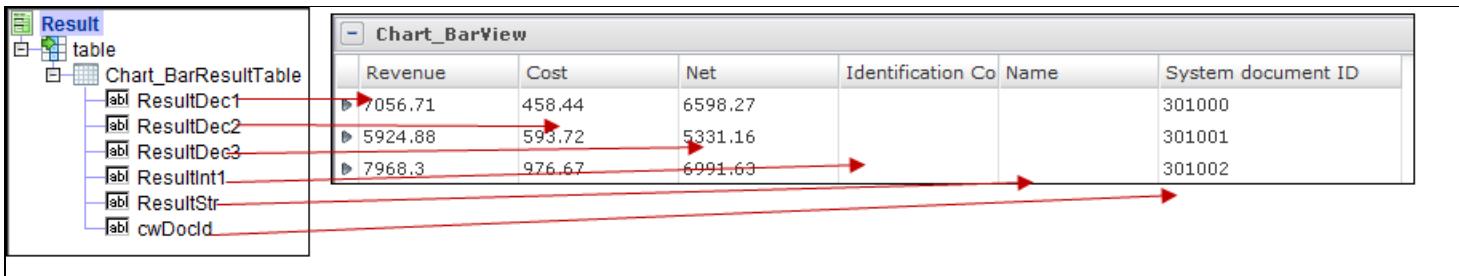
## Table

The table element is used to display [Finder Results Form](#) metadata in table format. By default, the Result Form display data that is configured using an array of documents in the Finder. Variables created for a document are assigned a type (that is, string, decimal, integer).

**Note:** Velocity Studio features require a separate license key that is activated in Velocity Studio.

General	Variables	Methods	Mapping								
			Name	Type			Methods	Vis	Opt	Edit	
ResultDec1				Decimal, 10.2			<input type="checkbox"/>				
ResultDec2				Decimal, 10.2			<input type="checkbox"/>				
ResultDec3				Decimal, 10.2			<input checked="" type="checkbox"/>				
ResultInt1				Integer, 12			<input type="checkbox"/>				
ResultStr				String, 20			<input type="checkbox"/>				
cwDocId				String, 16			<input type="checkbox"/>				

The system uses these variables and generates a corresponding Form element under the Table element in the Finder Result Form. The number of child Form elements that are displayed beneath the Table element (they must be set to visible) determines the number of columns displayed in the table at runtime.



At design time, to locate the Table element, from the **Select Form Element Type** dialog box, expand the Structures node. Form elements from the **Input Elements** and **Misc Controls** categories can be added beneath the table element.

At runtime, if table element is used with user interface that has any validation errors, the errors are displayed at the same time the table list is shown.

The table element is also important to configure the following options:

- Hover Forms
- Editing Table Data
- Cell Alignment

## Summary

references Variable	Yes
references Method	Yes
references Form	Yes
Allowable Child Elements	Checkbox Date Field Date Time Hyperlink Large Text Large Text Area Reference Field Select Field Translation Upload File Variable Image Label

The following table describes the properties of table element:

Property	Mandatory/Optional	Comment
----------	--------------------	---------

<b>Name</b>	Mandatory	Name of the Table element.
<b>Variable</b>	Mandatory	Choose from the list of Variables available in the User Interface to bind to this Table. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
<b>Auto Fit Data</b>	Optional	Set to <i>Vertical</i> , which allows the table to automatically resize to fit the incoming data. The <b>Height</b> property must be set to a numeric value and defines the minimum height for the table. The <b>Auto Fit Max Height</b> property defines the maximum height for the table. The default value for this property is DEFAULT.
<b>Auto Fit Max Height</b>	Optional	Set to a numeric value to automatically resize the table to fit the incoming data up to the defined height. Once this height is reached, a scroll bar appears to enable the user to view all data.
<b>Autosave on Data Change</b>	Optional	By default, this property is set to true, meaning that documents are automatically saved each time a record is edited. The default setting makes this feature fully backward-compatible. When this property is unchecked, editing rows in editable table are not saved automatically when exiting edit mode. Instead, table records must be saved by explicitly invoking the save method.
<b>Can Accept Drop</b>	Optional	This property is used to specify whether the table can accept a value dropped into it from another table on the application user interface. Possible values are TRUE, FALSE, or a permission method.
<b>Can Drag Out</b>	Optional	This property is used to specify whether you can drag a value out of the table to another table on the application user interface. Possible values are TRUE, FALSE, or a permission method.
<b>Can Re-order</b>	Optional	<p>This property allows you to specify the sorting order of enumerations by dragging and dropping row elements in a table. You can set this property to TRUE, as indicated in this example:</p> <ol style="list-style-type: none"> <li>1. Create a new finder. In the result form set, the Table element's <b>Can Re-order</b> property to TRUE.</li> <li>2. Add a save menu that is bound to a click method with a script using <code>this.result.getCurrentSequence()</code>.</li> <li>3. Ensure that you have data in the database column for this finder.</li> <li>4. Start runtime, load the finder, drag and drop some records from one row to another, and then click <b>Save</b>.</li> </ol>
<b>Cell Height</b>	Optional	The default height of each row in pixels.
<b>Cell Width</b>	Optional	The default width of each row in pixels.
<b>Double Click Method</b>	Optional	Executed when a row on a table is double-clicked.
<b>Editable</b>	Optional	<p>Specifies whether the Table is editable, <i>True</i>, or read-only <i>False</i>. The property provides a list of permission-based methods that set the permissions of the table. Setting this property to true enables editing for Large Text, Reference, Translation and Checkbox elements when they appear within a Table.</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• For each checkbox Form element in the Result Form, set the Toggle Edit property to <i>True</i>. If you have overridden the <i>OnSelChanged</i> method (from the Methods tab of the User Interface), ensure that the super implementation is called or the toggle edit functionality will not function properly. Default is NONE, which is editable.</li> <li>• If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's <i>editablePerm</i> permission takes over. If this method is not overwritten, the model's <i>editablePerm</i> permission is used. User interfaces displayed under another user interface with <i>editablePerm</i> are also be inherited if the embedded user interfaces do not have their own <i>editablePerm</i> permission.</li> </ul>
<b>Fast Cell Update</b>	Optional	<p>This advanced property improves performance for dynamic styling of grid-renderer cells in Internet Explorer, at the expense of slightly slower initial rendering and some limitations on supported styling options.</p> <p>For finders that have slow scrolling speed, select the <b>Fast Cell Update</b> property, clear the <b>Wrap Cells</b> property, and then change the image type for all the images to <b>center</b>.</p> <p><b>Note:</b> This property is only available when using Internet Explorer.</p>
<b>Filter Full Data</b>	Optional	When this property is turned on, filtering, regardless of whether the data is shown per paging, occurs on the entire dataobjectlist. Filter criteria remains the same when you move from one page to another. Filter criteria are only cleared if you clear the data.
<b>Get Drag Data Method</b>	Optional	If no method is specified for this property, the system calls <code>cwGetDragData</code> as the default method at runtime. The <code>cwGetDragData</code> method returns an object. For Finder User Interface, the default implementation of <code>cwGetDragData</code> method will just return the object passed in. The <b>Get Drag Data Method</b> property is only available if the <b>Can Drag Out</b> property is set to true or a permission method.
<b>Get Drag</b>	Optional	If no method is specified for this property, the system calls <code>cwGetDragTypes</code> as the default method at

<b>Types</b>		runtime. The cwGetDragTypes method returns a comma-separated string of metadata type names. For Finder User Interface, the default implementation of cwGetDragTypes method will gather a list of metadata type names that correspond to the output object's metadata type, and any objects in the metadata which extends this output type. The <b>Get Drag Types</b> property is only available if the <b>Can Drag Out</b> property is set to true or a permission method.
<b>Get Drop Data Method</b>	Optional	If no method is specified for this property, the system calls cwGetDropData as the default method at runtime. The cwGetDropData method returns an object. The default implementation will just return the object passed in. This method is called on the target controller near the beginning of cwOnDrop method. The <b>Get Drop Data Method</b> property is only available if the <b>Can Accept Drop</b> property is set to true or a permission method.
<b>Get Drop Types</b>	Optional	If no method is specified for this property, the system calls cwGetDropTypes as the default method at runtime. The cwGetDropTypes method returns an array of metadata type names. The implementation of cwGetDropTypes is the same as that of cwGetDragTypes. The <b>Get Drop Types</b> property is only available if the <b>Can Accept Drop</b> property is set to true or a permission method.
<b>Group Start Open</b>	Optional	This property describes the default state of table groups when the table is grouped by values for a given field or fields. The property can take the following values: <ul style="list-style-type: none"> <li>• <b>all</b> Open all groups.</li> <li>• <b>none</b> Start with all groups closed.</li> </ul> The default value is <b>all</b> .
<b>Head Height</b>	Optional	The height of this table's header, in pixels.
<b>Header Style</b>	Optional	The CSS style applied to the day-of-week headers. By default this applies to all days of the week.
<b>Height</b>	Optional	This property represents the height used or consumed by the table. The value is defined as the number of pixels (integer), or as a percentage of page height or of the parent element (for example, 30% of the page width or 150 pixels). <p>When the table height is set to 100%, it takes the parent's height. As an example, if you place a table into a form frame or a layout, it takes the height of the parent element. If the height of the parent element is not set, or is set in % and the parent's height is not set, it does not know what the actual height should be.</p> <p>The table does not have a minimum height. As a result, if the parent's height is not set properly, the form frame collapses its height to 0.</p>
<b>Hide Header</b>	Optional	Header item for a collapsible section in a Dynamic Form. Each SectionItem is associated with a number of other items in the form, which is shown or hidden as a group when the section is expanded or collapsed. Clicking a SectionItem expands or collapse the section. To make a form where only one section is expanded at a time, set DynamicForm.sectionVisibilityMode to <i>mutex</i> .
<b>Hover Form</b>	Optional	The form to show on hover. This form can be local to the finder user interface, or a form belonging to the hover variable's user interface. Additionally, a local string variable can be used to dynamically change the form to be used at runtime. This string variable should have ONLY the name of the form such as "Default". Setting this property to a string variable will automatically use the hover variable's user interface to look for the form if it is provided, otherwise it will use the finder user interface for the form lookup.
<b>Hover Method</b>	Optional	Specifies the method used to display the contents of the cell over which the user is hovering. The method must contain a script that sets the cwShowHover variable to <i>True</i> .
<b>Hover Style</b>	Optional	Style to apply to hovers shown over this table.
<b>Hover Variable</b>	Optional	Specifies a variable, for any user interface (that is, Document, Order), that displays a form when the user is hovering over a cell. The variable must be instantiated with a value.
<b>Include in Tab Order</b>	Optional	This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values: <ul style="list-style-type: none"> <li>• When set to &lt;Default&gt;, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order.</p>

		To ensure that an element is in the tabbing order, select this property.
<b>Is Tile Grid</b>	Optional	By default, this property is set to false. Select this property's checkbox if you want to use the tile grid layout.
<b>Load Data Message</b>	Optional	Defines the message that displays in the table body at runtime when data is being received from the application server. The default message is <i>Loading data</i> .
<b>No Items Message</b>	Optional	Defines the message that displays in the table body when the table is empty. The property is used to suppress the default message <i>No items to show</i> .
<b>On Drag Method</b>	Optional	This property can be used to specify a method that the system runs at runtime when a user drags an element from a tree or table in the application user interface. If no method is set for this property, the system uses cwOnDragOut as the default method at runtime. This property is only accessible if the <b>Can Drag Out</b> property is set to true or a permission method.
<b>On Drop Method</b>	Optional	This property can be used to specify a method that the system runs at runtime when a user drops an element (i.e., by releasing the mouse button) into a tree or table in the application user interface. If no method is set for this property, the system uses cwOnDrop as the default method at runtime. The cwOnDrop method will either copy or move the data being dropped. To copy, hold down the CTRL-key while you drag and drop. This property is only accessible if the <b>Can Accept Drop</b> property is set to true or a permission method.
<b>On Enter</b>	Optional	The system will call any script defined in this property (including <code>onInit</code> , <code>onClick</code> and <code>toString</code> ). This method is invoked when the form element is focused on the Web page and the user presses the ENTER key.
<b>On Filter Callback</b>	Optional	During filtering, the table selection gets updated based on the filtered list. This property allows you to specify a callback method. During the filtering, the selection list on the dataobject list is updated so that the only selection that is tracked is currently selected in the filtered list. Once filtering has been performed and the results are returned to the browser, the method specified in the property is then called.
<b>Scrolling on Update</b>	Optional	This property controls the scrolling behaviour after a table update and takes the following values: <ul style="list-style-type: none"> <li>• <b>Default</b> The table scrolls to the first selected record. If no record is selected, the table scrolls to the top of the records.</li> <li>• <b>Restore position</b> The table scrolls to the position before the update.</li> <li>• <b>No scrolling</b> The table always scrolls to the top after an update.</li> </ul>
<b>Select on View</b>	Optional	There are instances where you want your application to allow the user to double-click a row to see its details and then come back to the table with previous selections still intact. To achieve this functionality, select this property's checkbox. See the example that follows on how to use this property.
<b>Selection Appearance</b>	Optional	This property contains two options: <ul style="list-style-type: none"> <li>• simple</li> <li>• checkbox</li> </ul> When <b>checkbox</b> is selected, the first column of table contains a checkbox for each row. This checkbox allows you to select a row in the table. Clicking the row itself neither selects the row, nor fires the <b>Selection Changed Method</b> . The <b>Double Click Method</b> is still applicable when this property is set to <b>checkbox</b> . See the example that follows on how to use this property.
<b>Selection Changed Method</b>	Optional	This method is invoked when a selection is made in the Finder table.
<b>Selection Type</b>	Optional	This property allows the table element's selection type to be configured at a form level. Click this property's drop-down field and select from one of the following options: <ul style="list-style-type: none"> <li>• <b>DEFAULT</b> Other properties determine this selection type. See the note that follows.</li> <li>• <b>single</b> Select only one item at a time.</li> <li>• <b>multiple</b> Select one or more items by holding the <i>Ctrl</i> or <i>Shift</i> key.</li> </ul> <p><b>Note:</b> The following rules determine the table selection type:</p> <ol style="list-style-type: none"> <li>1. If the UserInterface-level <code>cwTableSelection</code> variable is set, use it to ensure backward compatibility.</li> <li>2. Otherwise, if the <b>Selection Type</b> form-level property is set, use it.</li> <li>3. If nothing has been set in the previous steps, the table element's <b>Selection Appearance</b> property determines the selection type.</li> </ol>

		<p>4. If the <b>Selection Appearance</b> property is <b>DEFAULT</b> or <b>simple</b>, the selection type is <b>multiple</b>.</p> <p>5. If the <b>Selection Appearance</b> property is <b>checkbox</b>, the selection type is to select one or more items as a toggle, so you do not need to hold down control keys to select more than one object.</p>
<b>Show Detail Column</b>	Optional	Specifies whether the Table's detail columns display ( <i>True</i> ) or hide ( <i>False</i> ). Default is true. At runtime, the user must click the icon to display or hide the column's data.
<b>Show Filter</b>	Optional	Specifies whether the Table can be filtered ( <i>True</i> ) or no filtered ( <i>False</i> ). Default is False.
<b>Status Icon Left Coordinate</b>	Optional	This property allows you to control the left coordinate of the status icon for tile table layouts (see the <b>Is Tile Grid</b> property). This property's default value is 5.
<b>Status Icon Top Coordinate</b>	Optional	This property allows you to control the top coordinate of the status icon for tile table layouts (see the <b>Is Tile Grid</b> property). This property's default value is 70.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Table such as positioning and color. Default is <b>NONE</b> .
<b>Summary Row Style</b>	Optional	Specifies the style of your summary row. Click the drop-down menu under <b>Values</b> and select from the styles available.
<b>Tile Label Align</b>	Optional	This property indicates the alignment of each tile label. Click the drop-down menu and select from one of the following options: <ul style="list-style-type: none"> <li>• <b>Left</b></li> <li>• <b>Right</b></li> </ul>
<b>Tile Count</b>	Optional	This property allows you to specify how many tiles to display per row. By default, this property is set to 0.
<b>Tile Size</b>	Optional	This property determines both the width and height in pixels.
<b>Tile Style</b>	Optional	This property determines the style of each tile.
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Label. Default is empty, which means no tooltip for the Label.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system Permissions such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to <b>NONE</b> , which means no permissions assigned. Setting the property to <b>TRUE</b> shows the table at runtime; otherwise, its value set to <b>FALSE</b> hides the table at runtime.
<b>Width</b>	Optional	Width used or consumed by the Table. The value is defined as the number of pixels (integer) or as a percentage of page width or of the parent element. For example, 30% of the page width or 150 pixels.
<b>Wrap Cells</b>	Optional	If <i>True</i> , column contents can wrap. If <i>False</i> , all the content appears on a single line.  <b>Note:</b> The text contained in the column must contain spaces or the system will not wrap the text.

#### Notes:

- For information about table functionality at runtime, see [Finders at Runtime](#).
- Radio button Form elements are not allowed in the Table. Instead add a Select element to the table.
- You can resize the columns of your finder table based on the percentage for input fields. If the width is not specified, the product uses a default **Width** of 100 px.
- The column height cannot be configured separately in tables. Instead, use the Table element's **Cell Height** property and set the **Wrap Cells** set to true, which changes the height of every column. Scroll bars for cells in read-only mode are not supported.
- Adding a table in a hover form is not supported.

The following Finder User Interface methods are invoked for a table element. For more information see [Finder User Interface Methods](#).

Method	Description
<b>onSelChanged</b>	If the Finder's table is editable ( <i>True</i> ), this method checks if the current column that the user has clicked has specified the toggle functionality to be turned on.
<b>copyAction</b>	Finder's with editable tables calls the <i>searchAction()</i> method in cases when this Finder is currently no displaying any data. A table element is notified to open the new row in edit mode.
<b>showDetailAction</b>	When the Finder's table has the <i>Show Detail Column</i> property set to <i>True</i> , this method is invoke when the detail column is clicked.

#### Editing Table Data

You can edit data in the table when the Table element Editable property is set to *True*. Setting the Editable property to *True* enables you to edit the fields

of the table at runtime. To edit table data at runtime, double-click the row that contains the data that you want to edit. For more information, see [Finders at Runtime](#).

## Drag and Drop Table Elements

On the application user interface, you can drag and drop a table element from one table to another. To enable drag and drop, the **Can Drag Out** and **Can Accept Drop** properties must be set to true or a permission method. By default, you can only drag and drop table elements if both tables have the same object type.

To create a copy of the element being dragged and dropped (instead of removing it from the original location), hold down the CTRL-key while you drag and drop.

To allow drag and drop between tables of different data types, write a new method or override the cwGetDropTypes method, which returns a comma-separated list of metadata full names to accept. Also, you will need to implement the cwGetDropData method to support other data types. As Data Object Lists and Data Structures only accept one type per list, cwGetDropData should provide a way to convert this type to the one actually accepted by the data source. The following shows an example of the implementation for the situation where Finder 1 accepts doc1, doc2.

Implementation for cwGetDropData:

```
var newObjects = new Array(object.length);
for(var i = 0; i < object.length; i++){
    if(object[i].metadataTypeName == "DragAndDrop.doc2"){
        var doc = new DragAndDrop.doc1();
        doc.name = object[i].name;
        newObjects[i] = doc;
    }
}
return newObjects;
```

Implementation for cwGetDropTypes:

```
return "DragAndDrop.doc2,"+ this.cw$super_cwGetDropTypes();
```

To provide a different list of data to give to the target element, implement cwGetDragData.

## Create a Table Cell Alignment Property

When you add new Form elements under an existing Table element, you can configure the Table element to display with a cell alignment property. The cell alignment property allows you to format how the data displays in the cell of the table at runtime.

To configure the cell alignment property, do the following:

1. Navigate to the Result Form of a Finder.
2. From the **Select Form Element Type** dialog, select an element from the list.
3. Click the **Next** button.
4. In the **Select Variable** dialog, select a variable for this element.
5. Click **Finish**.
6. Click the Form element that you added to the Table element.
7. Click the **Cell Alignment** property and select from the chart the type of alignment that you want to display at runtime.

\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Hover Form

By configuring the hover properties of the table, you can display a pop-up form as a result of a hover on a Finder result. There are four properties used to configure the hover option: Form, Method, Style and Variable. At runtime, when you hover a row, the form displays beside the cursor. See the [Properties](#) section. The cwShowHover variable in the Finder User Interface is of type Boolean. It must be set to *True* when the Hover Method is called. The value of this variable, once sent back to the browser, is set to null.

There are several ways in which to configure a hover form:

To configure a hover form using an local form, follow these steps:

1. In a **Finder User Interface**, right-click the User Interface node and select **New Form**.

2. In the **Element pane**, customize the new Form by adding Form elements and configuring their properties.
3. In the User Interface **Method** tab, create a method (New User Action) and create a script.
4. Set the `cwShowHover` variable to true.
5. Run the metadata.
6. In the Web browser, navigate to the Finder.
7. Place your cursor in the row of the data; the Form appears as a hover.

The following example shows how to implement the finder hover:

1. Create a string variable on the **Finder User Interface** (for example, `ToolTipText`).
2. Create a form in the Finder User Interface and call it `ToolTipForm`.
  - a. On this form, include a **Label** element.
  - b. Bind the **Label** element to the `ToolTipText` string variable.
3. Create a UI method and call it on `ToolTip`.

```
this.ToolTipText = null;
this.cwShowHover = false;

var list = this.model.list; // Note: this.model.list.hover finds the hovered row

if (list != null) {
    var overRow = list.hover; // an instance of resultDocument
    if (overRow != null) {
        this.ToolTipText = overRow.toolTipText;
        this.cwShowHover = true; // setting value to true results in tooltip form being displayed
    }
}
```

4. On the **Table** element, set the **Hover Form** and **Hover Method** properties.

To configure a hover form with a variable, complete these steps:

1. In a **Finder User Interface**, create a variable for the document that you want to display as a hover form.
2. Right-click the **Default** Form and select **Override**.
3. To the Default Form add a **Form Frame** Form element.
4. In the Form Frame **Variable** property, select the document user interface variable previously created (step 1).
5. Set the Form property using the Variable that refers to the document that you want displayed as the hover form.
6. Customize the Form by adding elements and configuring their properties.
7. Run the metadata.
8. In the Web browser, navigate to the Finder.
9. Place your cursor in the row of the data; the Form appears as a hover.

### Add Dynamic Editable Property in Finder

To support a dynamic editable property on a result form field (that is, the field is editable only if specific conditions are met), do the following:

1. Create a finder and set the table element's **Editable** property to **TRUE**.
2. Add bound fields under the table.
3. Create a new permission script method under the finder user interface. Proceed to add a parameter of type `com.conceptwave.system.Document` called `doc`.
4. Write a script that returns either true or false depending on `doc`'s variable value, such as the following samples:

```
return doc.boo_CB;
```

or

```
if(doc.str_TF == null)
    return true;
else
    return false;
```

5. Set this permission method as the **Editable** property of one of the table fields.
6. Run the framework, load the finder, and conduct a search.
7. Proceed to click a row. If the permission is to return true, the table field (column) is in editable mode. Otherwise, the table field (column)'s appearance does not change.

**Note:** There is a short delay (2 seconds) between the time the cursor is set on a row and the display of the form. This value may be overridden by adding a variable called **Hover Delay** into your design time configuration. Acceptable values for this variable are between 100 and 5000 milliseconds. Any

numbers above or below this range are ignored and the default 2000 (2 seconds) is used.

## Display a Hyperlink in a Finder with a URL Dependent on the Result Document

To support a dynamic URL in a finder, do the following:

1. Create a script method that returns a string. The following is an example of the script:

```
return "<a href='http://www.google.ca'>TEST</a>" ;
```

2. In your table, add a **Label** element and bind it to the script that you have created.
3. Set the Label element's **HTML Escape** property to false.
4. Ensure that the **Editable** permission on the link column is set to false.
5. Save your metadata and run.

## Use Visible Permissions with Table Columns Under a Top-level User Interface

In previous releases, table columns relied heavily on the presence of finder user interfaces. You can set visible permissions to work with table columns that are under a top-level user interface. The following is an example.

1. Create a new **User Interface** object.
2. In your user interface, create a Boolean variable and call it **hideCol**.
3. In the **Methods** tab, right-click your user interface and select **New Permission**. Create a new permission called **hideColPermission**.
4. In your **hideColPermission** script, add the following code:

```
return this.hideCol;
```

5. Create a table containing two columns:
  - Employee
  - Department
6. For the Department column, set its **Visible** property to **hideColPermission**.
7. Create a button called **disableColumnButton**.
8. Create a script called **disableColumn** that contains the following code:

```
this.hideCol = false;
```

9. For the **disableColumnButton**'s **On Click** property, call the **disableColumn** script.
10. For the user interface's **OnInit** method, add the following code:

```
this.hideCol = true;
```

11. Save your metadata and then run your application. The table appears, which shows both columns.
12. Click the **disableColumnButton** button. The Employee column appears and the Department column does not display.

## Use Selection Appearance and Select on View properties in a Finder

The following example shows how to use the **Selection Appearance** and **Select on View** properties:

1. Create a Finder user interface with an output document. In the result form, set the following table element properties:
  - **Selection Appearance** set to **checkbox**
  - **Select on View** set to **true**
2. Write a method that calls `this.result.getObjectToView()`. Proceed to set this method to the table's **Double Click Method** property.
3. At runtime, select few rows using the selection checkbox. Once you have selected these rows, double-click a row. Ensure that `this.result.getObjectToView()` is the row clicked and that `this.row.selected` contains only the rows that are selected.

## Multi-level paths for Variable field

When defining a result row in a finder, if you have a number of nested data structures, you can specify a multi-level path in the **Variable** field, such as `result.party.primaryPartyName.firstName`. For a multi-level path, you must first create a column field under the table, and then set the **Variable** field.

## Total Rows in Tables

You have the ability to show automatic summaries (totals) of data sets when viewed using a finder. The application provides a row specifically for displaying all or certain columns that are shown in a finder table.

To use the summary row functionality, you need the following Table field element properties for each input element under a Table:

Property	Dependency
Show Data Summary	The element must exist under a Table element. The Table element must have the <b>Show Data Summary</b> checkbox selected.
Show Group Summary	The element must exist under a Table element. At least one table field element must exist with the <b>Group by In Table</b> checkbox selected. Additionally, the table element must have the <b>Show Group Summary</b> checkbox selected, as well as the column, to see the summary in the grouping.
Summary Label	The element must exist under a table element. The <b>Summary Type</b> property must be set to <b>count</b> .
Summary Type	The element must exist under a Table element. The <b>Show Data Summary</b> checkbox must be checked. See the <a href="#">Summary Types</a> section for a list of available summary types.

See the Table element's [Summary Row Style](#) property that you can use to summarize total rows in tables.

## Summary Types

The following built-in **Summary Type** values are available:

Summary Type	Description
Sum	Iterates through the set of records, selecting and summing all numeric values for the specified field. Returns null to indicate an invalid summary value if any non-numeric field values are encountered.
Average	Iterates through the set of records, selecting all numeric values for the specified field and determining the mean value. Returns null to indicate an invalid summary value if any non-numeric field values are encountered.
Max	Iterates through the set of records, selecting all values for the specified field and finding the maximum value. Handles numeric fields and date type fields only. Returns null to indicate an invalid summary value if any non-numeric or date field values are encountered.
Min	Iterates through the set of records, selecting all values for the specified field and finding the minimum value. Handles numeric fields and date type fields only. Returns null to indicate an invalid summary value if any non-numeric or date field values are encountered.
Multiplier	Iterates through the set of records, selecting all numeric values for the specified field and multiplying them together. Returns null to indicate an invalid summary value if any non-numeric field values are encountered.
Count	Returns a numeric count of the total number of records passed in.

## Limitations

The following are limitations to keep in mind when using total rows in tables:

- All summary types, except **Count**, are only relevant to date and numeric fields.
- Only finders with all results visible in UserInterface have the ability to show the summary. Finders, such as the Event Log, are not capable of displaying the summary row unless the entire data is sent. Sending the full data is already done with the custom view loaded, which has sorting.

## Hover Detail Viewer for Tables

---

The hover detail viewer provides you with a clean and quick way to display additional information for finder table hovers.

To use the hover detail viewer, you need the following Table field element property for each input element under a Table:

Property	Description
<b>Show Data in Hover</b>	This property must be set for the column's data to appear as part of the detail viewer. The viewer shows a maximum of five data columns.  If no columns are selected and the table element's <b>Show Hover Detail</b> property is selected, all data columns marked as <b>Visible = false</b> are shown in the detail viewer.

**Note:** The hoverDelay configuration variable may be set to either increase or decrease the response time of the hover. By default, this value is set to 300 ms.

See the Table element's [\*\*Show Hover Detail\*\*](#) property that you can use to summarize total rows in tables.

## Row Filter in a Table

You can use the row filter in your browser to filter information in a table. This functionality allows you to perform the following tasks:

- Specify the **Can filter** property for each column
- Specify the filter editor type for columns:
  - Checkbox (checkbox or drop-down)
  - Select (drop-down or text)
- Filter DateTime columns using the Date Range cell editor, which is the same editor for the Date column
- Perform filtering on the server side only, allowing you to use more complex filtering criteria, such as the following with comparison operations:
  - The LIKE operation using \* for text and select columns
  - Operations such as <, <=, >=, >, =, != for numeric columns
  - Operators such as *in* (...) and *not in* (...) for both text and numeric columns

Filtering in a table consists of using a one-row filter editor that is displayed above the table column headers:

				Europe					G8	
Code	Country	Capital	Governmer	Continent	Nationhood	Area (km²)	Population	GDP (\$M)	G8	
MN	Monaco	Monaco	constitution	Europe	01/01/1419	1.9	31719	788	<input type="checkbox"/>	
NO	Norway	Oslo	constitution	Europe	10/26/1905	324220	4383807	106200	<input type="checkbox"/>	
LU	Luxembourg	Luxembourg	constitution	Europe		2586	415870	10000	<input type="checkbox"/>	
SZ	Switzerland	Bern	federal rep.	Europe		41290	7207060	158500	<input type="checkbox"/>	
DA	Denmark	Copenhagen	constitution	Europe		43070	5249632	112800	<input type="checkbox"/>	
LS	Liechtenstein	Vaduz	hereditary	c	01/23/1719	160	31122	630	<input type="checkbox"/>	
FR	France	Paris	republic	Europe		547030	58317450	1173000	<input type="checkbox"/>	
SW	Sweden	Stockholm	constitution	Europe		449964	8900954	177300	<input type="checkbox"/>	
UK	United Kingdom	London	constitution	Europe	01/01/1801	244820	58489975	1138400	<input type="checkbox"/>	
NL	Netherlands	Amsterdam	constitution	Europe	01/01/1579	37330	15568034	301900	<input type="checkbox"/>	
BE	Belgium	Brussels	constitution	Europe	10/04/1830	30510	10170241	197000	<input type="checkbox"/>	
IT	Italy	Rome	republic	Europe	02/07/961	301220	57480074	1099600	<input type="checkbox"/>	

You can view the row filter editor after you select the table element's **Show Filter** property.

To apply the filter criteria to the table, do one of the following:

- Press the **Enter** key in the filter editor
- Click the **Filter** image button located in the top right corner of the table

## Enable Filtering for Table

To enable filtering of your table, the following properties and options are needed:

- To show the filter row, select the Table element's **Show Filter** property.
- The **Filter Editor Height** property allows you to set the filter row height (the default value of this property is 20). Optionally, you can use the **Can Filter** property of the column field elements to show or hide the filter cell editor for each column.
- The Checkbox and Select columns also have **Filter Editor Type** option.

## Filter Editor Types

The following table shows the filter editor types that are available and their descriptions:

Filter Editor Type	Description
Text, Label, TextArea	<p>Uses the text field filter.</p> <p>For string data types, you can use the following:</p> <ul style="list-style-type: none"><li>• * and in(...) operators</li><li>• not in (...) operators</li></ul> <p>For numeric data types, you can use the following:</p> <ul style="list-style-type: none"><li>• Number comparison operators</li><li>• Simple list of values separated with either commas (,) or semicolons (;)</li><li>• in (...) and not in (...) operators</li></ul>
Checkbox	<p>The Checkbox column filter can be one of the following:</p> <ul style="list-style-type: none"><li>• <b>Checkbox</b> type (default) that displays a checkbox with three states:<ul style="list-style-type: none"><li>◦ Selected</li><li>◦ Unselected</li><li>◦ None</li></ul></li><li>• <b>Select</b> type with the following options<ul style="list-style-type: none"><li>◦ None</li><li>◦ Yes</li><li>◦ No</li></ul></li></ul> <p><b>Note:</b> The states Unselected and None may not be intuitively recognizable.</p>
Select	<p>The Select column filter can be one of the following:</p> <ul style="list-style-type: none"><li>• <b>Select</b> type that displays a drop-down menu with column values</li><li>• <b>Text</b> type to filter the value by visual key using either * or in (...) operators</li></ul> <p><b>Note:</b> The <b>Select</b> field with the <b>Select</b> filter type is not supported for dynamic enumerations and reference types. It is recommended that you use the <b>Text</b> filter type instead.</p>
Date, DateTime	Date range cell editor containing a read-only field with a button that, when clicked, shows a Date Range dialog.
Spinner	Text filter editor with numeric operators.
Hyperlink	Not supported.
Image	Not supported.
Calculated columns	The filter editor type depends on both the column type and the <b>Return</b> data type property specified in the Variable script method.

## Server-side Filtering

Filtering is performed completely on the server side with the internal Java code. The main benefit is that the finder does not have to be used to either re-select or filter the result manually by metadata or by running SQL statements.

Filtering occurs *on top* of the finder's SELECT operation, meaning that the Search action still has to be invoked to get the initial "result" list. The **Auto Search** property can be used to run search automatically. Invoking the finder's **Search** action (for example, by clicking the **Search** menu item) clears the current table filtering so that the table displays the original unfiltered result.

To apply the filter criteria, the table element sends a special data source request to the server. The server-side listener gets the table Variable list (for example, the finder "result"), and filters the records according to the specified criteria. Filtering does make any changes in the data source and the "result" list remains the same.

**Note:** Filtering can be used with any table that uses DataObjectList as a data source, and not necessarily the finder result.

## Filter Criteria and Notation

When using filter criteria, the following notations are available:

## String comparison

Keep the following points in mind when doing a string comparison:

- A string comparison is case-insensitive (for example, the **abc** filter matches **abc**, **ABC**, **Abc**, ... values).
- The wildcard (\*) character is not required, as it is automatic (for example, the **abc** filter matches **abctest**, **testabc**, **testabctest**, and so on). In other words, all values that contain the string are returned.
- The **in(...)** operation requires quotation marks, either single ('') or double ("") to separate values, which enable you to specify \* inside the filter value token.

The following are examples of using the **in(...)** notation:

```
// Use ' or ", or " inside '...'  
in ('ab', "*def", "xyz", ' Quotes inside the token: "quote" '')  
  
// 'in' and 'not in' operators in uppercase  
NOT IN ('abc', 'def')  
  
// Empty string finds records with either null or an empty value  
In ('')
```

## Number comparison

The number filter accepts the following:

- A simple sequence of values that are separated by either commas (,) or semicolons (;).
- Numeric comparison operators, such as <, <=, >=, >, =, and !=; which allow you to use multiple operators that are separated by either commas (,) or semicolons(;).
- The **in(...)** and **not in(...)** operators.

The number filter also allows you to use a set of strict equal comparisons combined with an OR operation, and others comparisons such as AND. The following is an example:

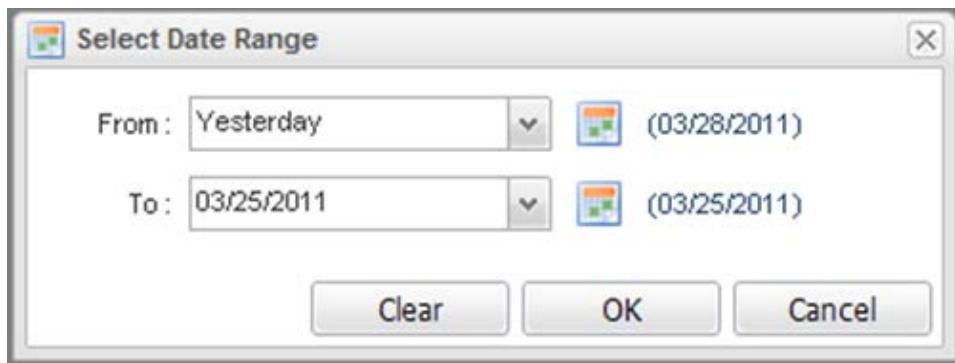
```
// "10, 15, >= 20, < 30, != 25" is parsed as  
// (x == 10) or (x == 15) or (x >= 20 and x < 30 and x != 25)
```

The following are examples of using number filter notations:

```
// Criterion is parsed as (x == 1 or x == 7.00 or x == 12)  
1, 7.00, 12  
  
// same as  
in (1, 7.00, 12)  
in ('1', '7.00', '12')  
  
// parsed as (x >= 1 and x < 7 and x != 25)  
>=1, <7, != 25  
  
// in(..) operator allows using other operators inside:  
in (1, 7, '> 12')  
  
// will return all non-empty values  
not in ('')
```

## Date Range Comparison

The filter editor for both the Date and DateTime columns displays the Select Date Range dialog when you click the button within the editor:



This filter returns all dates in specified range *inclusively*. The **From** and **To** fields can have empty values.

**Note:** For DateTime types, when comparing the **To** date, the time is truncated to include all timestamps of that date.

## Code Tables in an Editable Table

---

You can display unique code table lists based on the current result row, which ensures that value maps coming from dynamic code tables and reference finders are unique.

The following is an example of using this feature:

1. Create a finder user interface with an editable table. Have this editable show the following data types using select fields:
  - o Reference data type (either nRef or sRef)  
Ensure that the drop-down values for the reference data type are unique per result row (for example, by using conversion maps).
  - o Standard database enumeration  
For example, this data type can be a country database enumeration.
  - o Static enumeration (either string or integer)
  - o Dynamic code table
2. Save your metadata and then run it.
3. At runtime, the following actions occur:
  - o The column that displays a select field using the reference data type shows a different list for each row when you enter edit mode.
  - o Standard database enumerations do not change within a session. As a result, the drop-down list for this select field remains the same for each row.
  - o Static enumerations are always the same for each row.
  - o As with the reference drop-down column, the column that uses the dynamic code table has a unique drop-down list at runtime.

## Tile Grid Images

You can add images at the bottom of a tile in a tile grid (table element).

The following is an example of using this feature:

1. In Velocity Studio, add a table element to your form.
2. Select the table element's **Is Tile Grid** property checkbox.
3. Set the table element's **Variable** property.
4. Add data-specific columns (that is, input fields).
5. Proceed to add a horizontal layout and set the following properties:

Property	Description
<b>Height</b>	Set the horizontal layout's height to be shown at the tile's button. Ensure that the tile size is large enough to accommodate for this size.
<b>Members Margin</b>	Set the amount of space to use between each image under the layout.
<b>Style</b>	Select the style to use for the layout.
<b>Visible</b>	Select the permission to be called for each tile in the tile grid. It is recommended that you add this permission's second parameter to represent the data object being passed in.
<b>Width</b>	By default, the width has a value of <b>100%</b> .

6. Under the horizontal layout, you can add images. These images have a limited set of properties that can be used:

Property	Description
<b>Click Method</b>	This property denotes the click method for this image. To get the current tile, use <code>this.result.selected[0]</code> .
<b>Disabled</b>	Set this permission method to be called for each tile in the tile grid. It is recommended that you add this permission's second parameter to represent the data object being passed in.
<b>Height</b>	This property contains the image's height.
<b>Image Type</b>	Specify the image's type in case the height or width in the properties does not match the actual size.
<b>Image URL</b>	This property represents the static image to use for this tile.
<b>Visible</b>	Select the permission to be called for each tile in the tile grid. It is recommended that you add this permission's second parameter to represent the data object being passed in.
<b>Width</b>	This property denotes the image's width.

7. Save your metadata.

To use this functionality in UI Common, do the following:

1. Create a finder user interface and set the **Extends** field to `ui_common.baseFinder`.
2. Right-click **ResultTileGrid** and select **Override**.
3. Right-click the table and select **Extend**. Proceed to add the data needed to be shown in the tile.
4. To add new images in the horizontal layout, right-click the layout and select **Add**.

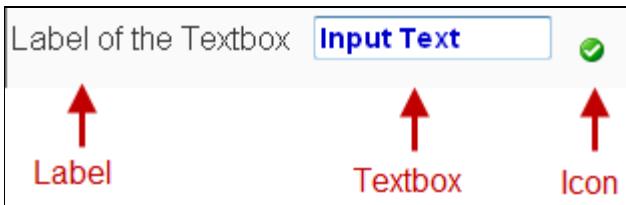
By default, the ResultTileGrid form includes the following icons:

Icon Name	Visibility	Disable	Functionality
tileActionIcons (layout with the icons)	isShowTileActions – returns <code>isTileActionButtonsVisible</code>	None	None
editIcon	isShowTileEdit – returns <code>isTileEditVisible</code>	isDisableTileEdit, which returns <code>isTileEditDisabled</code>	Defaults to call <code>onShowDetail</code> method
deleteIcon	isShowTileDelete – returns <code>isTileDeleteVisible</code>	isDisableTileDelete, which returns <code>isTileDeleteDisabled</code>	Defaults to call <code>onDelete</code>
copyIcon	isShowTileCopy – return <code>isTileCopyVisible</code> and is set to false by default	isDisableTileCopy, which returns <code>isTileCopyDisabled</code>	Defaults to call <code>onCopyAction</code> – this method is empty. Override it to provide the functionality.
folderIcon	isShowTileFolder - returns <code>isTileFolderVisible</code> and is set to false by default	isDisableTileFolder, which returns <code>isTileFolderDisabled</code>	Defaults to call <code>onFolderAction</code> – this method is empty. Override it to provide the functionality
favelIcon	isShowTileFavourite – returns <code>isTileFavouriteVisible</code> and is set to false by default	onFavouriteAction, which returns <code>isTileFavouriteDisabled</code>	Defaults to call <code>onFavouriteAction</code> – this method is empty. Override to provide the functionality

**Note:** Image clicks do not function if the tile grid contains a Selection Change method. Instead, the double-click method has been added to accommodate this restriction.

## Text Field

The Text Field element enables you to create a textbox with a label and icon. Various methods are associated with this element that enable you to invoke user actions (such as Run Trigger and On Enter). There are also various properties available to allow you to format the style and size of the Text Field element. The example below displays a Text Field with a label, an icon and a Text Box Style with a *blue* font style.



### Summary

references Variable	Yes
references Method	Yes, Run Trigger, On Enter
references Form	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Textfield.
Label	Mandatory	Visible label of the Text Field. If empty, there is no Text Field label. Default is empty.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
Auto-focus to next field	Optional	<p>This property indicates whether the focus moves to the next input field when the permitted length of the data is reached. By default, this checkbox property is selected. The next field is considered to be any input field (for example, text, select field, etc.) that is visible and editable (that is, not disabled).</p> <p>The behaviour applies to the following:</p> <ul style="list-style-type: none"><li>Masked text field, where the <b>Display Format</b> defines the permitted length</li><li>Regular text field, where the <b>Data Length</b> indicates the permitted length</li></ul>
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Text Field in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Text Field occupies in the Grid Layout, to the right of the current cell. Default is empty, which means it occupies one column.
Cell Row Span	Optional	This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout. <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row</p>

		<p>span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
<b>Data Length</b>	Optional	Specifies the maximum number of characters that can be entered into the select textbox. Default is 0, which means no character restrictions.
<b>Display Format</b>	Optional	<p>Specifies the accepted format that the text field will enforce and display at runtime. This property uses <i>regular expression</i> syntax to define the field length and allowable values. For example, a Display Format property of "[\d]{3}" limits the user to enter a number of fixed length 3 (3 digit number) at runtime. The <i>regular expression</i> entered here must be of a fixed length containing a number (\d), or word character (\w). Default is empty which means there are no data restrictions. Another example of this property value would be "[A-Za-z0-9]{9}" which would result in any alphanumeric character that is 9 characters long. Note that the <i>square brackets</i> ([ ]) has to proceed the fixed length format or <i>curly brackets</i> ({}).</p> <p>When a Decimal Data Type is assigned to the Variable, the \d, 0 or [0-9] formats are also acceptable. A \d\d\d\d.\d\d display format would result in a maximum of 2 digits displayed after the decimal and 4 digits to the left of the decimal.</p>
<b>Dynamic Label</b>	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
<b>Dynamic Label Style</b>	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
<b>Dynamic Tooltip</b>	Optional	<p>Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i>. The static Tooltip property is considered as default until the action returns a value.</p> <p>When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.</p>
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
		This property determines whether the Text Field is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be enabled or disabled).
<b>Editable</b>	Optional	<p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.</p>
<b>Error Icon Gap</b>	Optional	<p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Font Properties</b>	Optional	Specifies the font properties of the content entered in the textbox at runtime: font-family, font-color, font-size, bold, italic and underline. This overrides the styling assigned, if any,

		to the textbox.
<b>Group By In Table</b>	Optional	This property allows the user to group the Finder results by the elements value. Default is <i>False</i> .
<b>Height</b>	Optional	Height to be taken up by the Text Field area as defined as an integer value in pixels. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>HTML Escape</b>	Optional	<p>This property specifies if the HTML code is used to configure the text field. This property is only valid when parented by a Tree or Table element. Default is true.</p> <p><b>Note:</b> You can uncheck this property in a table to display and update the correct value in edit mode.</p>
<b>Icon</b>	Optional	Choose an image file that appears as an icon to the right of the Text Field. Default is NONE.
<b>Include in Tab Order</b>	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>• When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>Label Align</b>	Optional	Click the <b>Value</b> field and select from one of the following alignment options for your label: <ul style="list-style-type: none"> <li>• <b>Left</b></li> <li>• <b>Centre</b></li> <li>• <b>Right</b></li> </ul>
<b>Label Orientation</b>	Optional	Specifies where the label of the Text Field appears in relation to the Text Field area; can be <i>left</i> , <i>right</i> or <i>top</i> of the Text Field area. The default orientation is <i>right</i> . However, in a Grid Layout, the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout..
<b>Label Style</b>	Optional	Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the Label such as font color. Default is NONE.
<b>On Enter</b>	Optional	The system calls any script defined in this property. This method is invoked when the Text Field is focused on the Web page and the user presses the ENTER key.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Run Trigger</b>	Optional	<p>The method assigned to this property is invoked whenever the value of the Text Field changes. The method can be a User Action Method or a Script.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>• You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>• You click a menu.</li> <li>• You change your selection in a navigation tree.</li> <li>• You click another tab of a tabset.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• Field triggers do not fire on save. However, document triggers do.</li> </ul>

		<ul style="list-style-type: none"> <li>As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
<b>Show Label</b>	Optional	This property determines whether the label is displayed in the Text Field. Choose from <i>True</i> or <i>False</i> . Default is <i>True</i> which means that the label is displayed.
<b>Start Row</b>	Optional	Applicable only when parent Element is a Grid Layout. If <i>True</i> , the Text Field starts in a new row of the grid. If <i>False</i> , it is simply placed in the current cell as-is. Default is <i>False</i> , which does not start a new row.
<b>Style</b>	Optional	Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Text Field such as color and font. Refer to the information on <a href="#">Style Sheets</a> for details on configuration. Default is <b>NONE</b> .
<b>Text Align</b>	Optional	Specifies the alignment of the text in Text Field; it can be <i>left</i> , <i>center</i> , or <i>right</i> . Default is empty, which is left-aligned.
<b>Textbox Style</b>	Optional	<p>Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the textbox of the Text Field such as color and font. Default is <b>NONE</b>.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the <b>MyStyle</b> style to the <b>Textbox Style</b> property, the stylesheet must have <b>MyStyleDisabled</b> to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Text Field. Default is empty, which means no tooltip for the Text Field.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the text field. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Variable</b>	Optional	Choose from the list of Variables available in the User Interface to bind to this Text Field. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Determines whether the Text Field is visible or not. Choose true (visible) or false (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this field. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to <b>NONE</b> or <b>True</b> , which means no permissions assigned and field is visible.
<b>Width</b>	Optional	Width to be taken up by the Text Field; can be in integer which is in pixels, or as a percentage (for example, "30%") of the page width or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Notes

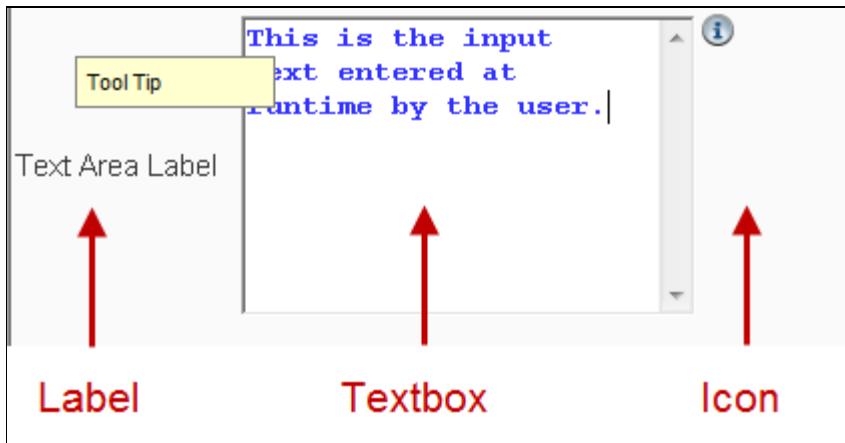
- The system automatically inserts commas or decimals to Variables of the type *Decimal* based on the data type's **Scale** and **Length** properties. Refer to the section on data types for more information.
- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- If Textfield is parented by a Grid Layout, the label orientation of the Textfield follows Grid Layout's **Label Orientation** property.
- Separate styles can be assigned to different parts of the Text Field: field area, textbox and label. The following is an example of a Text Field with *Style=blue* background, *Label Style=red* font colour, *Textbox Style=grey* background:



- If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment.
- If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Text Area

The Text Area element enables you to create a large textbox with a label and icon within the User Interface. Various methods are associated with this element that enable you to invoke user actions (such as Run Trigger and On Enter). There are also various properties available to allow you to format the style and size of the Text Area element. The example below displays a Text Area with a label, an icon, tooltip and a Text Box Style with a *blue bold* font style.



### Summary

references Variable	Yes
references Method	Yes, Run Trigger, On Enter.
references Form	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Text Area.
Auto Fit Text	Optional	Select this checkbox to have the text within the <b>Text Area</b> field automatically fit. This checkbox is not selected by default.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Text Area in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Text Area occupies in the Grid Layout, to the right of the current cell. Default is empty, which means it occupies one column.
Cell Row Span	Optional	This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.  By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions.

		<p>With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
<b>Data Length</b>	Optional	Specifies the maximum number of characters that can be entered into the select textbox. Default is 0, which means no character restrictions.
<b>Display Format</b>	Optional	Specifies the accepted format that the Text Area will enforce and display at runtime. This property uses <i>regular expression</i> syntax to define the field length and allowable values. For example, a Display Format property of "[\d]{3}" limits the user to enter a number of fixed length 3 (3 digit number) at runtime. The <i>regular expression</i> entered here must be of a fixed length containing a number (\d) or word character (\w). Default is empty which means there are no data restrictions. Another example of this property value would be "[A-Za-z0-9]{9}" which would result in any alphanumeric character that is 9 characters long. Note that the <i>square brackets</i> and parameter ({ }) has to proceed the fixed length format and <i>curly brackets</i> ({}).
<b>Dynamic Label</b>	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
<b>Dynamic Label Style</b>	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
<b>Dynamic Tooltip</b>	Optional	<p>Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i>. The static Tooltip property is considered as default until the action returns a value.</p> <p>When this element is used under the Table element, the <b>Dynamic Tooltip</b> property accepts a script, which is called for each result document value bound to a field with a dynamic tooltip set. This script returns the string desired to be shown on the hover. Additionally, the property accepts a variable value belonging to the result document. The value for this variable is then used to display the tooltip.</p>
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Editable</b>	Optional	<p>This property determines whether the Text Area is enabled or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be enabled or disabled).</p> <p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.</p>
<b>Error Icon Gap</b>	Optional	<p>Select this property's checkbox to make a gap for the validation icon. By default, this property is false.</p> <p><b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.</p>
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Font Properties</b>	Optional	Specifies the font properties of the content entered in the textbox at runtime: font-family, font-color, font-size, bold, italic and underline. This overrides the styling assigned, if any, to the textbox.
<b>Group By In Table</b>	Optional	This property allows the user to group the Finder results by the elements value. Default is <i>False</i> .
		Height to be taken up by the Text Area area as defined as an integer value in pixels. If

<b>Height</b>	Optional	this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>HTML Escape</b>	Optional	<p>This property specifies if HTML code is used to configure the text area. This property is only valid when parented by a Tree or Table element. Default is true.</p> <p><b>Note:</b> You can uncheck this property in a table to display and update the correct value in edit mode.</p>
<b>Icon</b>	Optional	Choose an image file that appears as an icon to the right of the Text Area. Default is NONE.
<b>Include Icon in Tab Order</b>	Optional	The property allows you to optionally exclude the icon button from the tab order.
<b>Include in Tab Order</b>	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>• When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>Label</b>	Optional	Visible label of the Text Area. If empty, there is no Text Area label. Default is empty.
<b>Label Align</b>	Optional	<p>Click the <b>Value</b> field and select from one of the following alignment options for your label:</p> <ul style="list-style-type: none"> <li>• <b>Left</b></li> <li>• <b>Centre</b></li> <li>• <b>Right</b></li> </ul>
<b>Label Orientation</b>	Optional	Specifies where the label of the Text Area appears in relation to the Text Area area; can be <i>left</i> , <i>right</i> or <i>top</i> of the Text Area area. The default orientation is <i>right</i> . However, in a Grid Layout, the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout..
<b>Label Style</b>	Optional	Choose from styles in the selected stylesheet at the toolbar, which defines the styling of the Label such as font color. Default is NONE.
<b>Label Vertical Align</b>	Optional	<p>This property defaults to <b>center</b> to denote that the label is centered vertically. It also takes values of <b>top</b> and <b>bottom</b>.</p> <p><b>Note:</b> This property only applies when the <b>Label Orientation</b> property is set to either <b>left</b> or <b>right</b>.</p>
<b>On Enter</b>	Optional	The system calls any script defined in this property. This method is invoked when the Text Area is focused on the Web page and the user presses the ENTER key.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Run</b>	Optional	<p>The method assigned to this property is invoked whenever the value of the Text Area changes. The method can be a User Action Method or a Script.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>• You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>• You click a menu.</li> </ul>

<b>Trigger</b>		<p>You change your selection in a navigation tree.</p> <ul style="list-style-type: none"> <li>• You click another tab of a tabset.</li> </ul>
		<p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• Field triggers do not fire on save. However, document triggers do.</li> <li>• As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
<b>Show Label</b>	Optional	This property determines whether the label is displayed in the Text Area. Choose from <i>True</i> or <i>False</i> . Default is <i>True</i> which means that the label is displayed.
<b>Start Row</b>	Optional	Applicable only when parent Element is a Grid Layout. If <i>True</i> , the Text Area starts in a new row of the grid. If <i>False</i> , it is simply placed in the current cell as-is. Default is <i>False</i> , which does not start a new row.
<b>Style</b>	Optional	Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Text Area such as color and font. Refer to the information on <a href="#">Style Sheets</a> for details on configuration. Default is <b>NONE</b> .
<b>Text Align</b>	Optional	Specifies the alignment of the text in Text Area; it can be <i>left</i> , <i>center</i> , or <i>right</i> . Default is empty, which is left-aligned.
<b>Textbox Style</b>	Optional	<p>Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the textbox of the Text Area such as color and font. Default is <b>NONE</b>.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the <b>MyStyle</b> style to the <b>Textbox Style</b> property, the stylesheet must have <b>MyStyleDisabled</b> to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Text Area. Default is empty, which means no tooltip for the Text Area.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the text area. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Variable</b>	Optional	Choose from the list of Variables available in the User Interface to bind to this Text Area. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Determines whether the Text Area is visible or not. Choose true (visible) or false (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this field. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to <b>NONE</b> or <b>True</b> , which means no permissions assigned and field is visible.
<b>Width</b>	Optional	Width to be taken up by the Text Area; can be in integer which is in pixels, or as a percentage (for example, "30%") of the page width or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Notes

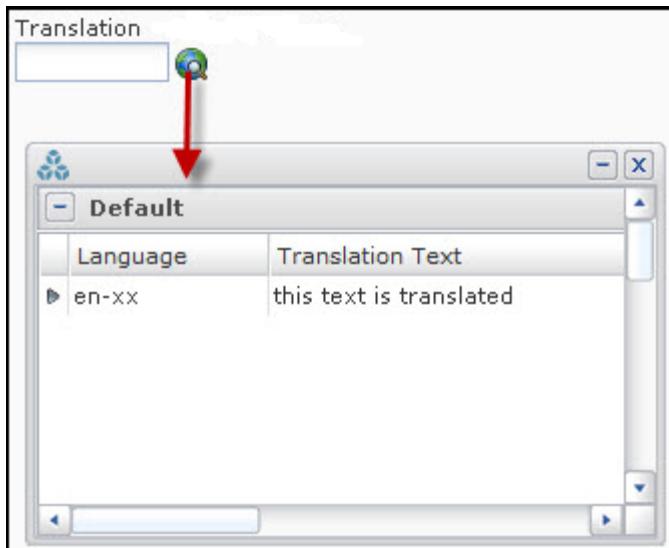
- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- If Text Area is parented by a Grid Layout, the label orientation of the Text Area follows Grid Layout's **Label Orientation** property.

- If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment.
- If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Translation

The Translation Element is a reference element that creates a Finder that displays translated text in a dialog box. This element type is bound to a translation variable using the Translation Data Type. Translation data is persisted to the Translation Data Type. The data contains both dynamic (the translated text) and static variables (the language type) which are mapped to rows as name-value pairs. The Translation Element type only works under Documents types, it will not work under topical user interface types.

After implementing the Translation Element, you can type text into the finder that you want translated. Clicking the icon of the Translation field displays a dialog box that contains translated text.



In addition, the Translation element can be structured inside of a table (see the following properties, Can Sort, Group by In Table).

### Summary

references Variable	Yes
references Method	Yes
references Form	No
Allowable Child Elements	None

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Label.
Can Sort	Optional	Specifies whether the Translation Field can be sorted or unsorted. The default is set to true. This option is available when the Translation Field appears in a table.
*Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Label in the page or parent Element. Default is undefined, which behaves as if it is top-left
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Label shall occupy in the Grid Layout, to the right of current cell. Default is empty, which occupy one column.
Cell Row Span	Optional	Only applicable when in a Grid Layout. Specifies the number of rows that the Label shall occupy in the Grid Layout, to the bottom of current cell. Default is empty, which occupy one row.
		This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when

<b>Cell Row Span</b>	Optional	<p>form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.</p> <p>By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.</p> <p>To use this property, see the <a href="#">example</a> for details.</p>
<b>Data Length</b>	Optional	Specifies the maximum number of characters that can be entered into the textbox. Default is 0, which means no character restrictions.
<b>Dynamic Tooltip</b>	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
<b>Dynamic Tooltip Style</b>	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
<b>Editable</b>	Optional	<p>Specifies whether the Translation Field is editable, <i>True</i> or read-only, <i>False</i>. Default is NONE, which is editable. Setting this property to <i>True</i> enables editing of this element when it appears within a Table.</p> <p><b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.</p>
<b>Group by In Table</b>	Optional	Displays the Translation Field data included in a table in Finder results. Each results is grouped in the table based upon its size (from largest to smallest). Default is <i>False</i> .
<b>Height</b>	Optional	The height that the Label uses. It can be in integer which is in pixels, or in percentage (for example, 50%) of page height or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>Icon</b>	Optional	If an icon is selected, the icon appears in the Translation, besides the label. Default is /cwfv/earth_view.png.
<b>Icon Action</b>	Optional	Only applicable when an Icon is specified. Choose from the list of Methods available in the User Interface to bind to the Icon. The method is invoked when a user clicks the Icon. Default is <i>onRefClick</i> .
<b>Label</b>	Optional	Visible label of the Translation. If empty, the label assumes the Name of the Translation. Default is empty.
<b>Label Align</b>	Optional	<p>Click the <b>Value</b> field and select from one of the following alignment options for your label:</p> <ul style="list-style-type: none"> <li>• <b>Left</b></li> <li>• <b>Centre</b></li> <li>• <b>Right</b></li> </ul>
<b>Label Orientation</b>	Optional	Optional. Choose the location of the Label. Options include top, left or right.
<b>Label Style</b>	Optional	Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Label such as font color. Default is NONE.
<b>On Enter</b>	Optional	When the cursor is in the text field and the user presses ENTER, the action of the selected method is invoked.
<b>Optional</b>	Optional	Validation error occurs when no data is entered in the large text box.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.

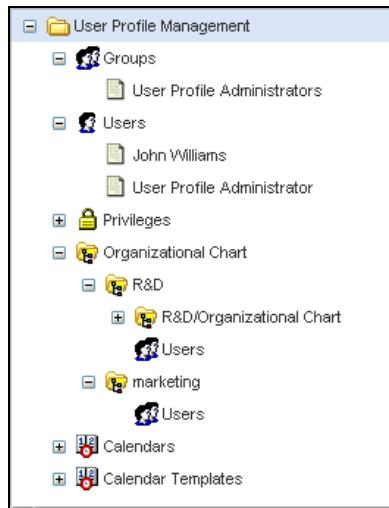
		<p>When data is changed in the Translation Field, and the dialog box becomes out-of-focus, the action of the selected Method is invoked.</p> <p>The trigger of a field fires only if the field has first gained focus, and then lost focus under these conditions:</p> <ul style="list-style-type: none"> <li>• You focus on another field through either clicking your mouse or pressing the Tab key.</li> <li>• You click a menu.</li> <li>• You change your selection in a navigation tree.</li> <li>• You click another tab of a tabset.</li> </ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• Field triggers do not fire on save. However, document triggers do.</li> <li>• As long as the cursor is still inside the field, the trigger does not fire. Clicking outside the field into the layout will not trigger the field as layouts are not focusable by default.</li> </ul>
<b>Start Row</b>	Optional	Applicable only when parent Element is a Grid Layout. If true, the Label starts in a new row of the grid. If false, it is placed in the current cell as-is. Default is /default, which does not start a new row.
<b>Style</b>	Optional	Defines the styling of the Label such as color and font. Default is NONE.
<b>Text Align</b>	Optional	Specifies the alignment of the text; it can be <i>left</i> , <i>center</i> , or <i>right</i> . Default is empty, which is left-aligned.
<b>Textbox Style</b>	Optional	<p>Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the textbox of the Textfield such as color and font. Default is NONE.</p> <p>To change the default disabled style for text fields, set this property with your own style, and make sure your stylesheet contains a style with the <i>Disabled</i> suffix appended to the assigned style. As example, when assigning the MyStyle style to the <b>Textbox Style</b> property, the stylesheet must have MyStyleDisabled to represent the disabled style.</p> <p>To change the disabled style for text fields globally, add the <i>fieldDisabled</i> style to your stylesheet.</p> <p><b>Note:</b> The <i>fieldDisabled</i> style is also applied to any element that has a text field when disabled (for example, the date element or the password element).</p>
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Label. Default is empty, which means no tooltip for the Label.
<b>Variable</b>	Optional	Choose from the list of Variables of type User Interface to bind to this Translation Field. Default is not assigned. Use the property's <a href="#">lookup button</a> to view property details.
<b>Visible</b>	Optional	Determines whether Translation is visible or not. Choose true (visible) or false (not visible) or from the list of Methods available in the User Interface to bind to the Field.
<b>Width</b>	Optional	The width that the Label uses. It can be in integer which is in pixels, or in percentage (for example, 30%) of page width or parent Element. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Notes

- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Tree

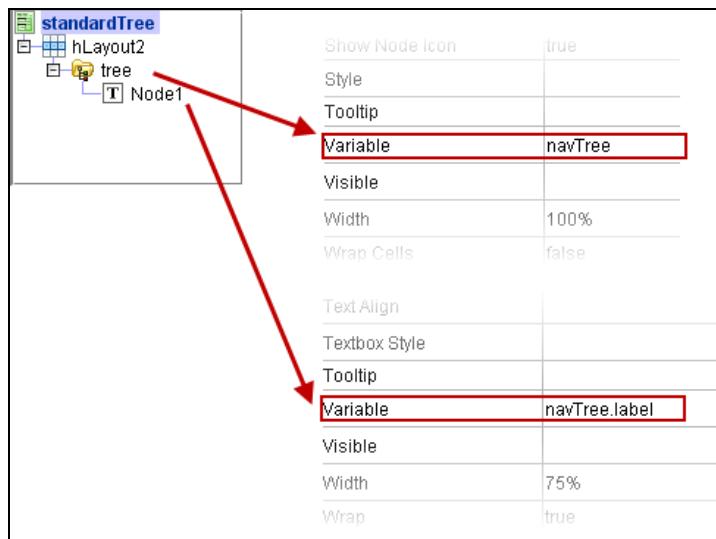
The Tree Form Element is a widget that presents hierarchical data set in a navigational tree. The nodes in the Tree Form Element may be expandable/collapsible by the user to show/hide its children nodes.



The Tree Form Element is the key Form Element to present [Navigation Tree](#) metadata objects, which is a hierarchical, derived User Interface object with a [Default Form at the Root Node](#) that renders the Tree at the left pane and displays corresponding node details at the details pane at the right when a node is clicked. The use within Navigation Tree metadata object is also the most common use for Tree Form Element. This is because the Tree Form Element, alone, does not readily show node details when a node is clicked.

The data set of the Tree, binded by **Variable** property of the Tree Form Element, should be a Document array. Furthermore, the Document type should be extended from `com.conceptwave.system.TreeDocument`, which has [Tree Document Variables](#) such as `parentid`, `label` and `image` to manage the structure and rendering of tree nodes.

At design time, to locate the Tree Form Element, from the **Select Form Element Type** dialog box, expand the **Structures** node. Once added, assign the array data set to the Tree's **Variable** property. Child Form elements from some of the **Input Elements** and **Misc Controls** categories can be added beneath the Tree. To create a standard-looking tree, add a single child Form Element with [Label](#), and assign `.label` Variable of the Tree's data set to the Label Form Element's **Variable** property.



See description of [Table Tree](#) that follows on having multiple child Form Elements beneath the Tree.

## Summary

<b>references Variable</b>	Yes
<b>references Method</b>	Yes
<b>references Form</b>	No
<b>Allowable Child Elements</b>	Checkbox Date Field Date Time Hyperlink Large Text Large Text Area Reference Field Select Field Text Area Text Field Translation Upload File Variable Image Label

## Properties

Property	Mandatory/Optional	Comment												
<b>Name</b>	Mandatory	Name of the Tree element.												
<b>Label</b>	Mandatory	Visual label of the Tree element.												
<b>Variable</b>	Mandatory	Assign the data set that this Tree is representing. This should be a Document array of type extended from system base Document <code>com.conceptwave.system.TreeDocument</code> . Use the property's <a href="#">lookup button</a> to view property details.												
<b>Auto Fit Data</b>	Optional	<p>This property defines how the tree resizes the body height to fit the retrieved data. Options are as follows:</p> <table border="1"> <thead> <tr> <th>Option</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><b>Default</b></td> <td>When the <b>Wrap Cells</b> field is false, the following actions occur at runtime:           <ul style="list-style-type: none"> <li>There is no horizontal data autofit. The column widths truncate the cell values.</li> <li>If column widths are not set, or at set in % to fill 100% of the tree width, no horizontal scrollbar appears.</li> <li>If column widths are set in px, cell values are truncated.</li> <li>The vertical scrollbar appears, if needed.</li> </ul>           When the <b>Wrap Cells</b> field is true, the cell text wraps and takes up multiple lines, if needed. The column widths remain the same.         </td></tr> <tr> <td><b>Vertical</b></td> <td>Expand vertically up to <b>Auto Fit Max Height</b> to fit all rows and accommodate nodes. No horizontal autofit is available.   <b>Note:</b> This option is available if the tree's <b>Height</b> property is not set or is set in px. This option is not available if the <b>Height</b> property is set in %.         </td></tr> <tr> <td><b>Horizontal</b></td> <td>Expand horizontally to accommodate nodes, with the initial width set to the <b>Width</b> property. The Tree element expands the width to fit all cell values up to <b>Auto Fix Max Width</b>, and then the scrollbar appears. The scrollbar is vertical by default.   <b>Note:</b> This option is available if the tree's <b>Width</b> property is not set or is set in px. This option is not available if the <b>Width</b> property is set in %.         </td></tr> <tr> <td><b>Horizontal Scroll</b></td> <td>Expand column widths to fit cell values, but without expanding the element width. The horizontal scrollbar appears, if needed. The vertical scrollbar appears by default.         </td></tr> <tr> <td><b>Both</b></td> <td>Expand both vertically up to <b>Auto Fit Max Height</b> and horizontally up to <b>Auto Fit Max Width</b> to fit all cell values and rows, and accommodate nodes.   <b>Note:</b> This option is available if both the <b>Width</b> and <b>Height</b> properties are either not set or are set in px.         </td></tr> </tbody> </table> <p>The default value for this property is <b>Default</b>.</p>	Option	Description	<b>Default</b>	When the <b>Wrap Cells</b> field is false, the following actions occur at runtime: <ul style="list-style-type: none"> <li>There is no horizontal data autofit. The column widths truncate the cell values.</li> <li>If column widths are not set, or at set in % to fill 100% of the tree width, no horizontal scrollbar appears.</li> <li>If column widths are set in px, cell values are truncated.</li> <li>The vertical scrollbar appears, if needed.</li> </ul> When the <b>Wrap Cells</b> field is true, the cell text wraps and takes up multiple lines, if needed. The column widths remain the same.	<b>Vertical</b>	Expand vertically up to <b>Auto Fit Max Height</b> to fit all rows and accommodate nodes. No horizontal autofit is available.  <b>Note:</b> This option is available if the tree's <b>Height</b> property is not set or is set in px. This option is not available if the <b>Height</b> property is set in %.	<b>Horizontal</b>	Expand horizontally to accommodate nodes, with the initial width set to the <b>Width</b> property. The Tree element expands the width to fit all cell values up to <b>Auto Fix Max Width</b> , and then the scrollbar appears. The scrollbar is vertical by default.  <b>Note:</b> This option is available if the tree's <b>Width</b> property is not set or is set in px. This option is not available if the <b>Width</b> property is set in %.	<b>Horizontal Scroll</b>	Expand column widths to fit cell values, but without expanding the element width. The horizontal scrollbar appears, if needed. The vertical scrollbar appears by default.	<b>Both</b>	Expand both vertically up to <b>Auto Fit Max Height</b> and horizontally up to <b>Auto Fit Max Width</b> to fit all cell values and rows, and accommodate nodes.  <b>Note:</b> This option is available if both the <b>Width</b> and <b>Height</b> properties are either not set or are set in px.
Option	Description													
<b>Default</b>	When the <b>Wrap Cells</b> field is false, the following actions occur at runtime: <ul style="list-style-type: none"> <li>There is no horizontal data autofit. The column widths truncate the cell values.</li> <li>If column widths are not set, or at set in % to fill 100% of the tree width, no horizontal scrollbar appears.</li> <li>If column widths are set in px, cell values are truncated.</li> <li>The vertical scrollbar appears, if needed.</li> </ul> When the <b>Wrap Cells</b> field is true, the cell text wraps and takes up multiple lines, if needed. The column widths remain the same.													
<b>Vertical</b>	Expand vertically up to <b>Auto Fit Max Height</b> to fit all rows and accommodate nodes. No horizontal autofit is available.  <b>Note:</b> This option is available if the tree's <b>Height</b> property is not set or is set in px. This option is not available if the <b>Height</b> property is set in %.													
<b>Horizontal</b>	Expand horizontally to accommodate nodes, with the initial width set to the <b>Width</b> property. The Tree element expands the width to fit all cell values up to <b>Auto Fix Max Width</b> , and then the scrollbar appears. The scrollbar is vertical by default.  <b>Note:</b> This option is available if the tree's <b>Width</b> property is not set or is set in px. This option is not available if the <b>Width</b> property is set in %.													
<b>Horizontal Scroll</b>	Expand column widths to fit cell values, but without expanding the element width. The horizontal scrollbar appears, if needed. The vertical scrollbar appears by default.													
<b>Both</b>	Expand both vertically up to <b>Auto Fit Max Height</b> and horizontally up to <b>Auto Fit Max Width</b> to fit all cell values and rows, and accommodate nodes.  <b>Note:</b> This option is available if both the <b>Width</b> and <b>Height</b> properties are either not set or are set in px.													
<b>Auto Fit Max Height</b>	Optional	Set to a numeric value to automatically resize the tree to fit the incoming data up to the defined height. Once this height is reached, a scroll bar appears that allows you to view all data.												
<b>Auto Fit Max Width</b>	Optional	Specifies the width in pixels that is the upper limit of the expansion due to AutoFit, if <b>Auto Fit Data</b> is <i>Horizontal</i> or <i>Both</i> .												
<b>Can Accept Drop</b>	Optional	This property is used to specify whether the tree can accept a node dropped into it from another tree on the application user interface. Possible values are TRUE, FALSE, or a permission method.												
<b>Can Drag Out</b>	Optional	This property is used to specify whether you can drag a node out of the tree to another tree on the application user interface. Possible values are TRUE, FALSE, or a permission method.												
<b>Can Re-order</b>	Optional	This property allows you to specify the sorting order of enumerations by dragging and dropping row elements.												

<b>Cell Editable</b>	Optional	This property lets you specify whether an individual cell can be edited, rather than at a record level (row) or a field level (column). This property accepts any method (on a tree controller, model, record document, and so on) that returns a boolean value. See <a href="#">Cell-level Properties</a> for details.  <b>Note:</b> If the tree cell is editable, the <b>Double click Method</b> bind to this cell does not trigger. Double-clicking this cell only makes the cell enter edit mode.
<b>Cell Height</b>	Optional	The default height of each node/row in the Tree in pixels.
<b>Cell Style</b>	Optional	This property lets you specify the style property for an individual cell, rather than at a record level (row) or a field level (column). This property accepts any method (on a tree controller, model, record document, and so on) that returns a boolean value. See <a href="#">Cell-level Properties</a> for details.
<b>Cell Visible</b>	Optional	This property lets you specify whether an individual cell is visible, rather than at a record level (row) or a field level (column). This property accepts any method (on a tree controller, model, record document, and so on) that returns a boolean value. See <a href="#">Cell-level Properties</a> for details.
<b>Cell Width</b>	Optional	The default width of each node-column in the Tree in pixels.
<b>Double Click Method</b>	Optional	Executed when a node on the Tree is double-clicked.
<b>Editable</b>	Optional	Specifies whether the Tree is editable, <i>True</i> , or read-only <i>False</i> . The property provides a list of permission-based methods that set the permissions of the Tree. Setting this property to true enables editing for LargeText, Reference, Translation and Checkbox elements when they appear within a Tree.  <b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.
<b>Generate DoubleClick on Enter</b>	Optional	If set to true, user pressing ENTER key shall generate the equivalent of double-click event at the selected node.
<b>Get Drag Data Method</b>	Optional	If no method is specified for this property, the system calls cwGetDragData as the default method at runtime. The cwGetDragData method returns an object. For Finder User Interface, the default implementation of cwGetDragData method will just return the object passed in. The <b>Get Drag Data Method</b> property is only available if the <b>Can Drag Out</b> property is set to true or a permission method.
<b>Get Drag Types</b>	Optional	If no method is specified for this property, the system calls cwGetDragTypes as the default method at runtime. The cwGetDragTypes method returns a comma-separated string of metadata type names. For Finder User Interface, the default implementation of cwGetDragTypes method will gather a list of metadata type names that correspond to the output object's metadata type, and any objects in the metadata which extends this output type. The <b>Get Drag Types</b> property is only available if the <b>Can Drag Out</b> property is set to true or a permission method.
<b>Get Drop Data Method</b>	Optional	If no method is specified for this property, the system calls cwGetDropData as the default method at runtime. The cwGetDropData method returns an object. The default implementation will just return the object passed in. This method is called on the target controller near the beginning of cwOnDrop method. The <b>Get Drop Data Method</b> property is only available if the <b>Can Accept Drop</b> property is set to true or a permission method.
<b>Get Drop Types</b>	Optional	If no method is specified for this property, the system calls cwGetDropTypes as the default method at runtime. The cwGetDropTypes method returns an array of metadata type names. The implementation of cwGetDropTypes is the same as that of cwGetDragTypes. The <b>Get Drop Types</b> property is only available if the <b>Can Accept Drop</b> property is set to true or a permission method.
<b>Header Height</b>	Optional	The height of this Tree's header, in pixels. Applicable only when <b>Hide Header</b> is false.
<b>Header Style</b>	Optional	The CSS style applied to the header.
<b>Height</b>	Optional	Height used or consumed by the Tree. The value is defined as the number of pixels (integer), or as a percentage of page height or of the parent element. For example, 30% of the page width or 150 pixels.
<b>Hide Header</b>	Optional	If false, displays the header for the column(s) at the top of the Tree, similar to header in a Table. Hides the header if true.
<b>Hover Form</b>	Optional	The form to show on hover. This form can be local to the finder user interface, or a form belonging to the hover variable's user interface. Additionally, a local string variable can be used to dynamically change the form to be used at runtime. This string variable should have ONLY the name of the form such as "Default". Setting this property to a string variable will automatically use the hover variable's user interface to look for the form if it is provided, otherwise it will use the finder user interface for the form lookup.
<b>Hover Method</b>	Optional	Specifies the method used to display the contents of the cell over which the user is hovering. The method must contain a script that sets the cwShowHover variable to <i>True</i> .
<b>Hover Style</b>	Optional	Style to apply to hovers shown over this Tree.
<b>Hover Variable</b>	Optional	Specifies a variable of User Interface type, that displays a Form when the user is hovering over a cell. Note: The UI Variable must have been instantiated at runtime.
<b>Icon Size</b>	Optional	Specifies the size of the icon images at the nodes of the Tree.
<b>Include in Tab Order</b>	Optional	This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values: <ul style="list-style-type: none"><li>When set to &lt;Default&gt;, the element is included in tab order according to global rules and its focusable state.</li><li>If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li><li>If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li></ul> <b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.
<b>Indent Size</b>	Optional	Specifies the amount of relative indent to incur when a child node is displayed beneath its parent node.
<b>Load Data Message</b>	Optional	Defines the message that displays in the Tree body at runtime when data is being retrieved from the application server. The default message is <i>Loading data....</i>
<b>Miller Columns</b>	Optional	If true, the Tree rendering becomes the Miller Columns widget instead of within a single fixed list table. See Note below.
<b>No Items Message</b>	Optional	Defines the message that displays in the Tree body when the table is empty. The property is used to suppress the default message <i>No items to show</i> .

<b>On Drag Method</b>	Optional	This property can be used to specify a method that the system runs at runtime when a user drags an element from a tree in the application user interface. If no method is set for this property, the system uses cwOnDragOut as the default method at runtime. This property is only accessible if the <b>Can Drag Out</b> property is set to true or a permission method.
<b>On Drop Method</b>	Optional	This property can be used to specify a method that the system runs at runtime when a user drops an element (i.e., by releasing the mouse button) into a tree in the application user interface. If no method is set for this property, the system uses cwOnDrop as the default method at runtime. The cwOnDrop method will either copy or move the data being dropped. To copy, hold down the CTRL-key while you drag and drop. This property is only accessible if the <b>Can Accept Drop</b> property is set to true or a permission method.
<b>On Enter</b>	Optional	The system will invoke the script defined in this property when the Form Element is focused on the web page and the user presses the ENTER key.
<b>On Expand/Collapse Method</b>	Optional	At runtime, when expanding or collapsing a tree node, the method specified for this property is invoked and the expanding node document is passed as a parameter.
<b>Opener Icon</b>	Optional	
<b>Overflow</b>	Optional	<p>Defines how overflow child form elements are displayed based on the height/width of the tree.</p> <p><b>Auto</b> - The child elements that do not fit within the height of the Tree defined are shown via a scroll bar. In other words, the Tree is scrollable if the child members exceeds its specified size.</p> <p><b>Hidden</b> - The child elements that do not fit within the Tree are NOT displayed at runtime.</p> <p><b>Visible</b> - If the child field element members size exceeds the specified size, the Tree will grow to accommodate the child element members.</p>
<b>Selection Changed Method</b>	Optional	This method is invoked when the selected node is changed in the Tree.
<b>Show Alternating Style</b>	Optional	If enabled, shows alternating rows (nodes) in alternating styles, appearing as a "ledger" effect for easier reading.  <b>Note:</b> If this property is enabled, the cell style for alternating rows has <i>Dark</i> appended to the original cell styles. You must ensure that the Dark group appended styles are in your stylesheet. For example, if the original style is <i>cell</i> , have styles such as <i>cellDark</i> , <i>cellOverDark</i> , <i>cellSelectedDark</i> , and so on in your stylesheet.
<b>Show Connectors</b>	Optional	Boolean that specifies whether to show connector lines between parent and child nodes.
<b>Show Node Icon</b>	Optional	Boolean that specifies whether to show icon of nodes in the Tree.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Tree such as positioning and color. Default is NONE.
<b>Tooltip</b>	Optional	Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Tree. Default is empty, which means no tooltip for the Label.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the tree. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Visible</b>	Optional	Defines the user permissions assigned to this Tree, which governs whether the Tree is visible in the Form or not.
<b>Width</b>	Optional	Width used or consumed by the Tree. The value is defined as the number of pixels (integer) or as a percentage of page width or of the parent element. For example, 30% of the page width or 150 pixels.
<b>Wrap Cells</b>	Optional	<p>If this property is true, column contents can wrap. Otherwise, if this property is not selected, all the content appears on a single line.</p> <p><b>Note:</b> When this property is selected, the fixed, default cell height is not used. As a result, some gaps may appear between tree connectors. To adjust the gap, set the <b>Cell Height</b> property (for example, 12 px), which acts as the tree cell's minimum height while <b>Wrap Cells</b> is true.</p>

## Notes

The Tree Form Element is commonly used in conjunction, and within the Form of, Navigation Tree. See [Navigation Trees at Runtime](#) for examples on how the Tree Form Element works within Navigation Tree.

## Table Tree

The Tree Form Element is capable of having multiple child Form Elements to display multiple columns for the hierarchical data set, similar to columns in a [Table](#). Such usage is referred to as the *Table Tree*. This feature enables showcasing multiple Variables in the Document array at the Tree's **Variable** property. Furthermore, some of these columns may even be made editable at the Tree at runtime, controlled by the child Form Element's **Editable** property.

When creating a Table Tree, you should set **Hide Header** property to false to show the labels of the column headers.

Name	Title	Salary
Charles Madigen	Chief Operating Officer	\$26,200,000
Ralph Brogan	Mgr Software Client Supp	\$13,700,000
Bhushan Sam	Line Worker	\$8,300,000
Betsy Rosent	Met Read/Coll	\$7,450,000
Francine Berg	Env Asst	\$5,350,000
Amanda Jone	Line Wrker A	\$7,050,000
Fa Bai	Sr Engineer	\$8,100,000
Chase Godwi	Dsl Sys Rep	\$8,900,000
Carol Finley	Mgr Fin Rpts Budgets	\$311,100,000
Gene Porter	Mgr Tech Plng IntIS T	\$18,100,000
Rogine Leger	Mgr Syst P P	\$10,500,000
Abigail Lippman	Mgr Proj Del	\$17,500,000
Christian Jean	Line Wrker A	\$8,500,000
Jeremy Malor	Th Pl Opr	\$6,550,000
Danielle Maye	Equip Engineer	\$7,600,000
Jack Chartran	Driver Clerk	\$7,050,000
Jake Epting	Gen Maint B	\$6,950,000
Kate Giusti	Elec Maint A	\$5,250,000
Ai Ma	Dir Labour Rel	\$7,750,000

Table Tree may also be created in a Navigation Tree. See [Navigation Trees at Runtime](#) for details on how to synchronize the data set of the Tree with the model objects that the tree nodes are representing.

#### Miller Columns

Miller Columns is an alternative presentation of hierarchical data set. Unlike the regular Tree appearance where children nodes are placed below the parent node at an indented fashion when expanded using the expand/collapse icon, Miller Columns show an extra column at the right to display the children nodes when the parent node is clicked, and so on.

Charles Madigen	Charles Madigen (10)	John Garrison (13)	Galina Ugarova
Ralph Brogan	Ralph Brogan	Anna Galetti	No items to show.
Carol Finley	Carol Finley	Glenda Stockton	
Gene Porter	Gene Porter	Rong Xu	
Rogine Leger	Rogine Leger	Galina Ugarova	
Abigail Lippman	Abigail Lippman	Brenda Cox	
John Garrison	John Garrison	Alexia Badger	
Rui Shu	Rui Shu	Pat Freele	
Kirill Amirov	Kirill Amirov	Cameron Miller	
Joan Little	Joan Little	Lori Newton	

Simply check the **Miller Columns** boolean property in the Tree to enable this presentation. When using a Miller Column, you may set **Hide Header** property to false to show the labels of the column headers, which display the label of the parent node.

Miller Columns may also be used in a Navigation Tree. See [Navigation Trees at Runtime](#) for details.

#### Hover Form

By configuring the hover properties of the Tree you can display a pop-up form as a users mouse-hover on a Tree node. There are four properties used to configure the hover option: **Hover Form**, **Hover Method**, **Hover Style** and **Hover Variable**, as described in the Properties table above. The *cwShowHover* variable in the Tree's User Interface (typically, the root node of Navigation Tree) is of type Boolean. It must be set to *True* when the Hover Method is called. The value of this variable, once sent back to the browser, is set to null.

The Hover Form may be a local Form (of the Tree's User Interface metadata object), or a foreign Form. Typically, Hover Forms can be configured by creating a User Action Method at the Tree's User Interface that instantiates the model object/User Interface/Form for the Hover, with Variable *cwShowHover* of the Tree's User Interface set to true at the method script, and then associate this User Action Method at the Tree's **Hover Method** property.

#### Cell-level Properties

The methods for the **Cell Editable**, **Cell Visible**, and **Cell Style** properties can take two parameters that specify the current cell:

- *param1*: Document (denotes the current record document)
- *param2*: String (current field name)

Your script can return the property value based on these parameters.

At runtime, cell-level properties override both field-based and tree-based properties. If a cell-level property has not been specified, the property is inherited from the field. For a cell to be editable, the entire tree must be editable. Similarly, for a cell to be visible, the entire tree must be visible. However, the cell style overrides the tree style.

**Note:** The cell-level visibility property for a checkbox does not work. A checkbox is a control and it can neither be hidden, nor displayed at a cell level. It can only be performed at a field level.

### Drag and Drop Tree Nodes

On the application user interface, you can drag and drop a node from one tree to another. To enable drag and drop, the **Can Drag Out** and **Can Accept Drop** properties must be set to true or a permission method. By default, you can only drag and drop tree nodes if both trees have the same object type.

To create a copy of the element being dragged and dropped (instead of removing it from the original location), hold down the CTRL-key while you drag and drop.

To allow drag and drop between trees of different data types, write a new method or override the cwGetDropTypes method, which returns a comma-separated list of metadata full names to accept. Also, you will need to implement the cwGetDropData method to support other data types. As Data Object Lists and Data Structures only accept one type per list, cwGetDropData should provide a way to convert this type to the one actually accepted by the data source. The following shows an example of the implementation for the situation where Finder 1 accepts doc1, doc2.

Implementation for cwGetDropData:

```
var newObjects = new Array(object.length);
for(var i = 0; i < object.length; i++){
    if(object[i].metadataType == "DragAndDrop.doc2"){
        var doc = new DragAndDrop.doc1();
        doc.name = object[i].name;
        newObjects[i] = doc;
    }
}
return newObjects;
```

Implementation for cwGetDropTypes:

```
return "DragAndDrop.doc2," + this.cw$super_cwGetDropTypes();
```

**NOTE:** For navigation trees, cwGetDropData and cwGetDropTypes need to be overwritten *per* tree node.

To provide a different list of data to give to the target element, implement cwGetDragData.

To only allow drag and drop on certain tree nodes, specify a permission method on **Can Accept Drop** or **Can Drag Out** for the default behavior. Under the **Methods** tab of each tree node you want to modify, provide a permission method with the same name and return true/false.

### Display Multiple Rows Within a Single Tree Cell

To display multiple rows within a single tree cell, do the following:

1. Set your Tree element's **Wrap cells** property to **true**.
2. The column that displays the multiline text should be **Label**, and not text field or text area. Set the **Label** property, with the **HTML Escape** property set to **false**.
3. Instead of using **\n**, use **<br>** to set line breaks between rows.

### Include a Hyperlink in a Tree Cell

Adding a hyperlink in a tree element is unsupported. However, to open an external link in a dialog from a hyperlink can be done as follows:

1. For the Label element under the tree, set the **HTML Escape** property to **false**.
2. Override the **onNodeVisualKey** method from the Methods tab to return the HTML for the hyperlink. For example, to open the link in a new tab, in the **Script** field, return `<a href='http://www.ericsson.com' target='_blank'>Ericsson</a>`. To open the link in a new dialog, return `<a href='http://www.ericsson.com' onclick='window.open(this.href, \"Ericsson\", \"width=1200,height=600,scrollbars=yes,resizable=yes\"); return false;'>Ericsson</a>`.
3. When you run your metadata, the table behaves as if it were a hyperlink.

## Upload File

The Upload File element allows the user at runtime to browse files and upload the file to the database. The Browse button allows the user to select the file to be uploaded and the Upload button saves the file to the database. The file contents are uploaded to the database and associated with the document specified in the Upload Owner property. Only a single file can be uploaded successfully to each instance of an Upload File element.

A callback method is invoked once the upload has completed in either in error or successfully. The status of the callback is a string that either indicates "success" or an error code that indicates the reason for upload failure. The callback method can be used to handle the errors.

The following is an example of the Upload File element user interface at runtime.

The screenshot shows a user interface for the Upload File element. At the top, there is a tab labeled 'General' which is currently selected. Below the tabs, there is a section titled 'Header' containing the text 'Select your file for upload:'. Underneath this text is a file input field with the label 'File:' and a 'Browse...' button to its right. At the bottom of this section is a large, prominent 'Upload...' button.

The Upload File element is a browser-driven control, meaning that its look and feel differs depending on the Web browser that you are using to view it. For example, Google Chrome does not use the text field that Firefox uses to show the file that is currently selected. Instead, Google Chrome uses a label beside the **Upload File**. The **File** input field is for display only, meaning that you cannot enter the file path manually.

### Summary

references Variable?	Yes
references Method?	Yes, Icon Action, On Enter and Run Trigger.
references Form?	No
Allowable Child Elements	<i>None</i>

### Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Upload File.
Label	Mandatory	Visual label of the Upload File element. This property is translatable.
Button Label	Optional	Label name of the <b>Upload</b> button, which defaults to <b>Upload....</b> This property is translatable.
Button Style	Optional	Select from various button styles. This property is only visible when the form exists under an UploadPopup User Interface type.
Callback Method	Optional	This method is invoked once the upload has been completed either in error or successfully. The status of the upload completions (success or error) is passed to the uploadCallback property. This Callback method use the uploadCall back parameter.
Can Sort	Optional	This property is used when parented by a Table Element and refers to a Finder table column during runtime. This will allow the user to sort the search results data by ascending or descending order at runtime. Setting this property to <i>False</i> will disable the sorting functionality. By default this functionality is set to <i>True</i> .
Cell Alignment	Optional	Controls the horizontal and vertical positioning of the Upload File in the page or parent element. Default is undefined, which behaves as if it is top-left.
Cell Column Span	Optional	Only applicable when in a Grid Layout. Specifies the number of columns that the Upload File occupies in the Grid Layout, to the right of current cell. Default is empty, which means it occupies one column.
Cell Row	Optional	This property provides more flexibility when building a grid-based user interface, instead of using a combination of horizontal and vertical layouts. This property appears when form elements appear under a Grid Layout element and the form element has a parent element. This property specifies the number of rows that the element occupies in the grid layout.

<b>Span</b>		By default, this field is set to 1. All nine positions are made available for the <b>Cell Alignment</b> property if the form element is under a Grid Layout element and has a row span greater than 1, so that the form element can be aligned in nine different positions. With rowspan as 1, only the three centre positions are available.  To use this property, see the <a href="#">example</a> for details.
<b>Dynamic Label Style</b>	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
<b>Editable</b>	Optional	This property determines whether the Upload File is enable or disabled at runtime. The values for this property are either <i>True</i> (enabled), <i>False</i> (disabled) or a <i>Permissions</i> method (for example, depending on the user's permissions, this field will be enabled or disabled). If the value is set to <i>True</i> then the user will be able to enter text in the Dialog Text Editor. If the value is set to <i>False</i> , then the Dialog Text Editor appears but the user is unable to enter text in this dialog box.  <b>Note:</b> If a field contains an editable permission that is either coming from the form field itself, or inherited from the variable or data type, it takes precedence. Otherwise, the user interface's editablePerm permission takes over. If this method is not overwritten, the model's editablePerm permission is used. User interfaces displayed under another user interface with editablePerm are also be inherited if the embedded user interfaces do not have their own editablePerm permission.
<b>Error Icon Gap</b>	Optional	Select this property's checkbox to make a gap for the validation icon. By default, this property is false.  <b>Note:</b> This property is available only for left (or default) error orientation. The gap is calculated according to the /cwfv/error.png icon size. When the image is substituted with a smaller one, the gap is also smaller.
<b>Error Icon Orientation</b>	Optional	This property allows you to configure how an icon are displayed. This property takes the values left (default) and right.
<b>Height</b>	Optional	Height to be taken up by the Text Area area as defined as an integer value in pixels. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
<b>Include in Tab Order</b>	Optional	This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values: <ul style="list-style-type: none"> <li>When set to &lt;Default&gt;, the element is included in tab order according to global rules and its focusable state.</li> <li>If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.
<b>Label Orientation</b>	Optional	Specifies where the label of the Upload File appears in relation to the Upload File area; can be <i>left</i> , <i>right</i> or <i>top</i> of the Upload File area. The default orientation is <i>right</i> . However, in a Grid Layout, the default label orientation is defined by the Grid Layout element. But if a value of this Label Orientation property is specified, than it overrides the Label Orientation property provided by the Grid Layout.
<b>Label Style</b>	Optional	Choose from styles in the selected stylesheet on the toolbar, which defines the styling of the Label such as font color. Default is NONE.
<b>On Enter</b>	Optional	The system calls any script defined in this property. This method is invoked when the Upload File is focused on the Web page and the user presses the ENTER key.
<b>Remove Label Line</b>	Optional	Select this property's checkbox when the element is in a grid layout and its column span is equal to the grid layout's number of columns. In other words, if the field is the only field in that row, this empty line is removed above the element.
<b>Show Label</b>	Optional	This property determines whether the label is displayed on the Upload File. Choose from <i>True</i> or <i>False</i> . Default is True which means that the label is displayed.
<b>Show Upload Button</b>	Optional	This property determines whether the Upload button is displayed on the form. Choose from <i>True</i> or <i>False</i> . Default is True which means that the button is displayed. It is possible to set this property to False and add a Button element. Associate the Button element with the Upload file by using the Upload Control Name property on the Button element. These properties are only visible when the form exists under an UploadPopup User Interface type.
<b>Start Row</b>	Optional	Applicable only when the parent element is a Grid Layout. If true, the Upload File appears in a new row of the grid. If <i>False</i> , it is simply placed in the next available cell. Default is <i>False</i> , which means that the Upload File element does not appear on a new row.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the button such as positioning and color. Refer to the information on <a href="#">Style Sheets</a> for details on configuration. Default is none.
		Defines the tooltip text, which is displayed in a hover box when the cursor hovers over the Upload File. Default

<b>Tooltip</b>	Optional	is empty, which means no tooltip for the Upload File.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the element. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Upload Owner</b>	Mandatory	This is a local variable of type Document or a Method within a Document User Interface. Refer to the Notes section below for a more detailed explanation.
<b>Visible</b>	Optional	Determines whether the Upload File is visible or not. Choose true (visible) or false (not visible) or from the list of Permission Methods available in the User Interface to bind to the Field. This property defines the user permissions assigned to this field. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE or True, which means no permissions assigned and field is visible. Refer to the notes below for a Visible example.
<b>Width</b>	Optional	Width to be taken up by the Upload File; can be in integer which is in pixels, or in percentage (for example, "30%") of page width or parent Element. If this field is empty, the Width of the field should be taken up by that defined in the Column Width property of the Grid Layout. If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.

## Notes

- If this element is nested under a Table element, the width and height properties can be set using either the percentage (%) or pixel (px) measurement. If this element is not nested under a Table element, the width and height properties are only available using the pixel setting.
- Any runtime instance of a Document can only have one attachment associated with it, which means that only one file can be associated with each instance of this element. Any attempt to associate another file will fail. It is recommended that the upload file element is hidden once a file has been uploaded - This is controlled by the Visible property via permissions.
- The File text field as well as the Browser button are not customizable as they are browser controls.
- An attempt to upload a "null" file will fail.
- Uploading a file containing Byte Order Mark (BOM) characters is supported.

## Visible Property

The example below will check that a Document has been created (`this.model.isStored`) but that the user has not uploaded a file (`this.model.mimetype = null`) to that document. If the user has uploaded a file to the document, then the system will not display the Upload element that has been assigned this Visible property - `showUploader`.

Name:	showUploader	<input type="checkbox"/> Is pr								
Description:										
<input type="checkbox"/> Reference										
Parameters:	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>\$psCondition</td> <td>com.conceptwave.system.Boolean</td> </tr> <tr> <td></td> <td></td> </tr> <tr> <td></td> <td></td> </tr> </tbody> </table>		Name	Type	\$psCondition	com.conceptwave.system.Boolean				
Name	Type									
\$psCondition	com.conceptwave.system.Boolean									
Script:	<pre>function showUploader(\$psCondition) { // Runs every time the permission is evaluated     var permissionObject = this;     return this.model.MIMETYPE == null &amp;&amp; this.model.isStored; }</pre>									
Return:	<input type="checkbox"/> com.conceptwave.system.Boolean <div style="float: right;">▼</div>									
Privileges:	<input checked="" type="checkbox"/> Privileges and participant operations <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Add Shared Favorites</li> <li><input checked="" type="checkbox"/> Administration App</li> <li><input checked="" type="checkbox"/> ConceptWave API Access</li> <li><input checked="" type="checkbox"/> Delegate Task</li> <li><input checked="" type="checkbox"/> Delete Shared Favorites</li> <li><input checked="" type="checkbox"/> Everyone</li> <li><input checked="" type="checkbox"/> Get Available</li> <li><input checked="" type="checkbox"/> Group Manager</li> <li><input checked="" type="checkbox"/> Process Manager Administrator</li> <li><input checked="" type="checkbox"/> Return Task</li> <li><input checked="" type="checkbox"/> Runtime Administrator</li> <li><input checked="" type="checkbox"/> Set Process Priority</li> <li><input checked="" type="checkbox"/> Show Error Details</li> <li><input checked="" type="checkbox"/> Take on Task</li> <li><input checked="" type="checkbox"/> User Profile Administrator</li> <li><input checked="" type="checkbox"/> Workgroup Select</li> <li><input checked="" type="checkbox"/> Worklist Administrator</li> </ul>									

### Upload Owner Property

This property value determines where the uploaded file is stored. The values of this property is either a document associated to a database table or a method which is called when the file is uploaded. Both are restricted to the local parent User Interface. The contents of the attachment are stored in the CWFBLOBATTACHMENT table. The CWFBLOBATTACHEMENT table contains two columns or leafs (DocID and Blob). The Blob column stores the actual content of the attachment and the DocID references the attachment.

#### Document

- Variable of type *Document*: The downloaded file will be associated to the document via the DocID. The Document must contain a column with MIMETYPE which stores the file type. Without a MIMETYPE column/leaf the attachment will be stored as a text field (this is particularly important when storing an image type file). ORIGINALFILESPEC is another column or leaf in the table that is required. It stores the file name.
- Creating a Document of type cwf\_oe.CWDOCATTACHMENT will point to the database table CWDOCATTACHMENT which already contains the columns or leafs DOCID, MIMETYPE and ORIGINALFILESPEC, which would allow for any content type to be uploaded. Careful when using this system table as there are several mandatory fields that need to be updated when the file is uploaded (DocID, DocMetaDataType and VisualKey).

Name	Type	Methods	Vis	Opt	Edit
confirmObject	Object	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
model	Document	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CWDATTACHMENT	Document	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
uploadOwnerDoc	Document	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

General		Methods	
Name:	CWDATTACHMENT	<input type="checkbox"/> Array	<input type="checkbox"/> Constant
Data type:	cwf_oe.CWDATTACHMENT (AttachmentDoc)	<input type="checkbox"/> Audit	
Type error code:		Mandatory error code:	
Element properties:	<Inherited>		
	Name	Value	

- Only 1 (one) file can be uploaded per Document or Upload element. Once a file has been uploaded successfully another element can not be uploaded to the same Document variable. The original file must be deleted before a new file can be uploaded.
- cwf\_oe.CWDATTACHMENT** in the system library can serve as a good example of this setup required for this element and its properties.

#### Method

- The script method assigned to this property will be called when a file is uploaded. The method should be assigned with an Object type parameter `com.conceptwave.system.Object`. This object will call an instance of the `FileReader` scriptable object and may use the APIs available under the [FileReader](#) for interaction with the uploaded data. This script will contain instructions for uploading the file including the database table for uploading the attachment.
- The example below shows `file` parameter as an instance of the File Reader API (not included).

Name:	uploadOwnerMethod	<input type="checkbox"/> Is private				
Description:						
Parameters:	<table border="1"> <tr> <td>Name</td> <td>Type</td> </tr> <tr> <td>file</td> <td>com.conceptwave.system.Object</td> </tr> </table>	Name	Type	file	com.conceptwave.system.Object	<input type="checkbox"/> +
Name	Type					
file	com.conceptwave.system.Object					
Script:	<pre>function uploadOwnerMethod(file) { // Metadata type method. Can be called by scripts. }</pre>					

- To programmatically access the filename data returned when a user browses for a filename using the Upload File element, do the following:
  - Create a custom method (for example, `uploadOwnerMethod ()`, as shown in the previous example) and associate it to the `UploadOwner` property in the UI.
  - The `file` parameter is of type `FileReader`. Proceed to use `file.getFileName()` or another function of the `FileReader` object.

**Note:** The `getFileName()` method only returns the filename and not the entire filename path.

#### Call Back Property

- This property is invoked after the upload completes. During the file upload process, the system can pass the status of the file upload to a "status" parameter. It is necessary to define this parameter as a `string` in the callback method as shown below to capture "success" or failure indicated by a system generated string that conveys what went wrong during the upload process.

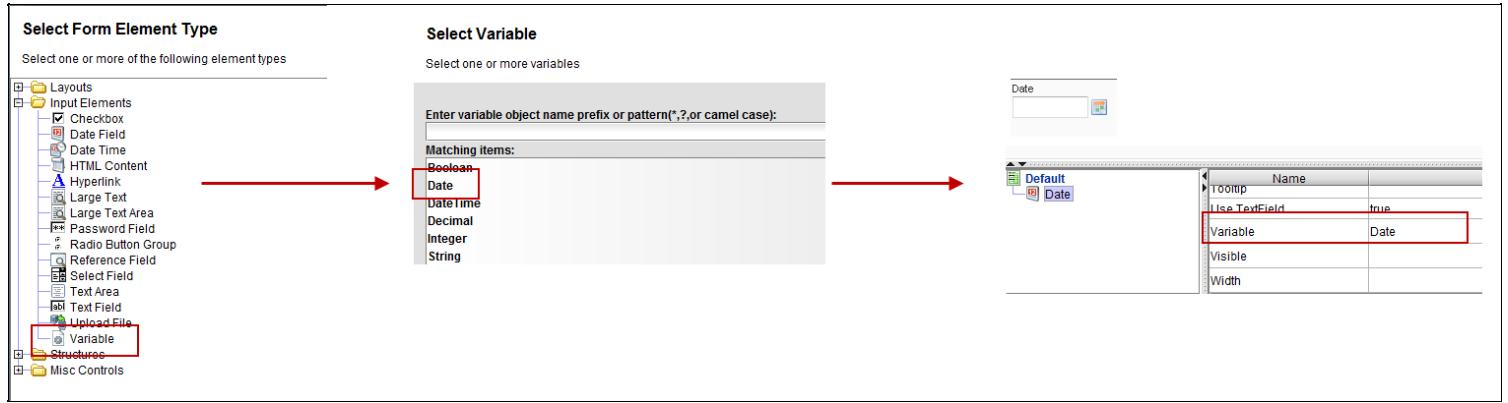
Name:	uploadCallBack				
Description:					
Parameters:	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>status</td> <td>com.conceptwave.system.String</td> </tr> </tbody> </table>	Name	Type	status	com.conceptwave.system.String
Name	Type				
status	com.conceptwave.system.String				
Script:	<pre>function uploadCallBack(status) { // Metadata type method. Can be called by scripts.     if(status != "success") {         Global.showUserMessage("Upload file failed due to: " + status);     }     this.status = "The status of the upload attachment is:"+status;</pre>				

- Without a Call Back method defined, the client UI will not update itself to convey any changes until another action forces an update. This means that if the Upload Owner method changes some value that should be reflected in the client UI, the changes will not be seen or could be lost if a Call Back method is not specified.
- Below are example the supported CallBack Errors:

Error Number	Error Message	Comments
DE0259	Error when processing attachment of object	
DE6038	File uploaded exceeds the maximum file size	The default file size limit is 1MB.
DE6039	No File was uploaded	Empty File or the user clicked the Upload button without selecting a file.
DE6040	Data uploaded is not multipart data	Potentially a corrupted file
DE6044	Unknown error occurred during data upload.	

## Variable

The Variable Element is a shortcut for creating an element. When selecting the Variable element, the system will determine the appropriate field element to create based on the *variable* type selected. The variable will bind to the Variable property of the element created. It is not a real Element, but a user interface development shortcut. The example below displays the selection of a Variable element of type *date* and the associated creation of the Date Field element with the Variable property bound to the Date variable.



The type of field Element that is created depends on the Variable's data type, as listed in this table:

Variable's Data Type	Resulting Field Element
Boolean	Checkbox
Date	Date Field
DateTime	Date Field
Decimal	Textfield
Integer	Textfield
String	Textfield

## Vertical Layout

---

There are three layout directives for user interface design:

- Horizontal
- Vertical
- Grid

Vertical Layout presents child elements in a vertical stack. The example below displays Credit Card information page with all credit card field elements displayed in a vertical layout. The user is able to specify the alignment of the elements relative to the page (for example, Right, Left, or Center).

**Credit Card Information**

Credit Card Type:  Visa  MC

Name on Credit Card:

Credit Card Number:

Expiry Date (mm/yy):

### Summary

<b>references Variable?</b>	No
<b>references Method?</b>	Yes
<b>references Form?</b>	No
<b>Allowable Child Elements</b>	Grid Layout Horizontal Layout Vertical Layout Layout Spacer Form Frame Button Checkbox Date Field Header Hyperlink Image Label Large Text Password Field Radio Button Group Reference Field Rich Text Editor Section Stack Select Field Separator Text Area Text Field Translation Upload File Variable HTML Content Iterator Menu Item

Tabset
Table
Dynamic Table
Dynamic Document

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Vertical Layout.
Can Accept Drop	Optional	<p>This property allows you to drop objects to layouts from a table, tree, or tile grid. The following example shows how to use this property.</p> <ol style="list-style-type: none"> <li>1. Create a layout and set the <b>Can Accept Drop</b> property to either <b>TRUE</b> or a permission method.</li> <li>2. Set On Drop method to a method that you previously created (that is, in the Methods tab, right-click the method node and select <b>Create Drop Method</b> from the menu, which provides the required parameter list).</li> <li>3. Implement this method to suit your needs.</li> </ol> <p><b>Note:</b> There is no default implementation for dropping objects into a layout.</p>
*Cell Alignment	Optional	<p>The cell alignment property functions in conjunction with the height/width properties. The values in the height/width properties must be set for the cell alignment property to function.</p> <p>This value specifies the horizontal and vertical positioning of its children within the Vertical Layout. An undefined value results in a "top-left" alignment.</p>
Dynamic Tooltip	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
Dynamic Tooltip Style	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
Edge	Optional	Select this checkbox to show an edge around a layout element. See the other Edge properties in this table to customize your edged layout.
Edge: Content Offset	Optional	This property denotes the amount the contained elements are offset to. The property defaults to <b>Edge: Size</b> if the value of the <b>Edge: Content Offset</b> is 0. Set this property to less than <b>Edge: Size</b> to allow the contained elements to overlap the edge and corner media.
Edge: Custom Sides	Optional	<p>Use this property to list custom sides for the border, such as <code>left</code>, <code>top</code>, <code>bottom</code>, and <code>right</code>. Some examples include the following:</p> <ul style="list-style-type: none"> <li>• <code>left, top, bottom</code></li> <li>• <code>top, bottom</code></li> </ul>
Edge: Dynamic Image	Optional	<p>This property contains either a method or variable that sets the edge image dynamically (starts with a <code>/</code>). If the dynamic image is null, a static image is used for the edge instead.</p> <p><b>Note:</b> If neither a dynamic, nor static image is specified, the default <code>[SKIN]/edge.gif</code> image is used.</p>
Edge: Fill Center	Optional	Specify whether to show the image background (with the <code>_center</code> extension) in the centre section (that is, behind the decorated layout).
Edge: Size	Optional	Use this property to define the size in pixels for corners and edges.
		<p>This property contains the base name of images for edges. Extensions for each corner or edge piece are added to the image URL before its file extension. For example, a default base name of <code>edge.gif</code>, the top-left corner images is <code>edge_TL.gif</code>.</p> <p>The list of valid extensions is as follows:</p>

<b>Edge: Static Image</b>	Optional	<ul style="list-style-type: none"> <li>• _TL (top-left)</li> <li>• _TR (top-right)</li> <li>• _BL (bottom-left)</li> <li>• _BR (bottom-right)</li> <li>• _T (top)</li> <li>• _L (left)</li> <li>• _B (bottom)</li> <li>• _R (right)</li> <li>• _center (centre)</li> </ul>
<b>Fill Space</b>	Optional	<p>Set this property to true if the parent layout is needed to fill empty spaces with its children. For example, if the parent layout is 100% height and 100% width, and its children combined either use less than or more than 100%, the parent resizes its children to use 100%.</p> <p>This property should not be set to true if the parent layout requires a centered alignment. In general, when a parent has one child and the parent's alignment is set to center, the child should only be moved to the centre position. However, if the parent is instructed to <i>fill space</i>, its child is resized to fill the rest of the space that is unoccupied.</p>
<b>Height</b>	Optional	Height used or consumed by the Vertical Layout. The value is defined as the number of pixels (integer) or as a percentage of page height or of the parent element. For example, 50% of the page height or 200 pixels.
<b>Include in Tab Order</b>	Optional	<p>This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values:</p> <ul style="list-style-type: none"> <li>• When set to <b>&lt;Default&gt;</b>, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul> <p><b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.</p>
<b>Members Margin</b>	Optional	Specifies the number of spaces allowed between grid layouts in the layout stack.
<b>Mouse Over Method</b>	Optional	Click this property's <b>Value</b> field and select a method to invoke when users hover their mouse over a specified element.
<b>On Enter</b>	Optional	The system calls any script defined in this property. The onEnter property is only met if there is an input field (for example, a textfield, selectfield, and so on) that has focus when the <b>Enter</b> key is pressed. This onEnter property is invoked only if the input field does not have an OnEnter property set. This feature can be used with multiple input fields, but only one onEnter method to fire for all fields.
<b>Overflow</b>	Optional	<p>Defines how overflow child form elements are displayed based on the height of the layout. The default is Auto.</p> <p><b>Auto</b> - The child elements that do not fit within the height of the layout defined are shown via a scroll bar. The user can scroll across to see the children at runtime. In other words, the layout will scroll if the child members exceeds its specified size.</p> <p><b>Hidden</b> - The child elements that do not fit within the height of layout are NOT displayed at runtime.</p> <p><b>Visible</b> - If the child field element members size exceeds the specified layout size, the layout will grow to accommodate the child element members.</p>
<b>Redraw</b>		By default, this property is unchecked. When selected, this property works when form layouts have either a width or height set at 100%. When you resize the browser window, the layouts in the current form are redrawn so that it fits the current window's size.

<b>on resize</b>	Optional	<b>Note:</b> For complex layouts, it is not recommended that you use this property, as the entire page is redrawn. As a result, redrawing the entire page may take a while. Depending on the complexity of the page and the slowness of your computer, you may see a delay in seeing the page being redrawn.
<b>Show Resize Bar</b>	Optional	This feature only works when the Vertical Layout is parented by a Horizontal Layout or another Vertical Layout. If true, a resize bar is created for Vertical Layout, that allows the user to resize the Vertical Layout by dragging it, and show/hide layout by clicking it. The orientation of the Resize bar depends on the parent - Vertical Layout parent yields a resize bar left-and-right, Horizontal Layout yields a resize bar up-and-down. The length of the resize bar is also dictated by the parent. Specifically it is the height of the Vertical Layout parent, or the width of the Horizontal Layout parent, that determines the resize bar's length; an explicit width/height value must be set to have the resize bar appear. Default is <i>default</i> , which is not showing the Resize Bar.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Vertical Layout such as positioning and colour. Default is NONE.
<b>Tooltip</b>	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers over the Vertical Layout. Default is empty, which means that a tooltip has not been defined. During runtime, the tooltip may be difficult to find as it is not the only layout on the screen and may cover a large area.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the Vertical Layout. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as addPerm, deletePerm, editablePerm, attachPerm, executePerm, and visiblePerm can be assigned as well as Object Permissions. Defaults to NONE, which means no permissions assigned.
<b>Width</b>	Optional	Width used or consumed by the Vertical Layout. The value is defined as the number of pixels (integer) or as a percentage of page width or of the parent element. For example, 30% of the page width or 150 pixels.

\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout, for example vlayout or hlayout with the width/height set, this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width/height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

## Notes:

- If you require drawing a border or box around a label or single form item element, wrap the element with Grid Layout instead of Horizontal Layout, Vertical Layout, Horizontal Stack Layout, or Vertical Stack Layout. The following are examples of drawing a border around a label:
  - It is recommended that you use a Grid Layout with the **Edge** property selected and a Label element under the layout:



- It is not recommended that you use a Vertical Layout with the **Edge** property selected and a Label element under the layout:



The reason for using the Grid Layout is that at runtime, the label and any form element that is not under Grid Layout) is already wrapped with Grid Layout.

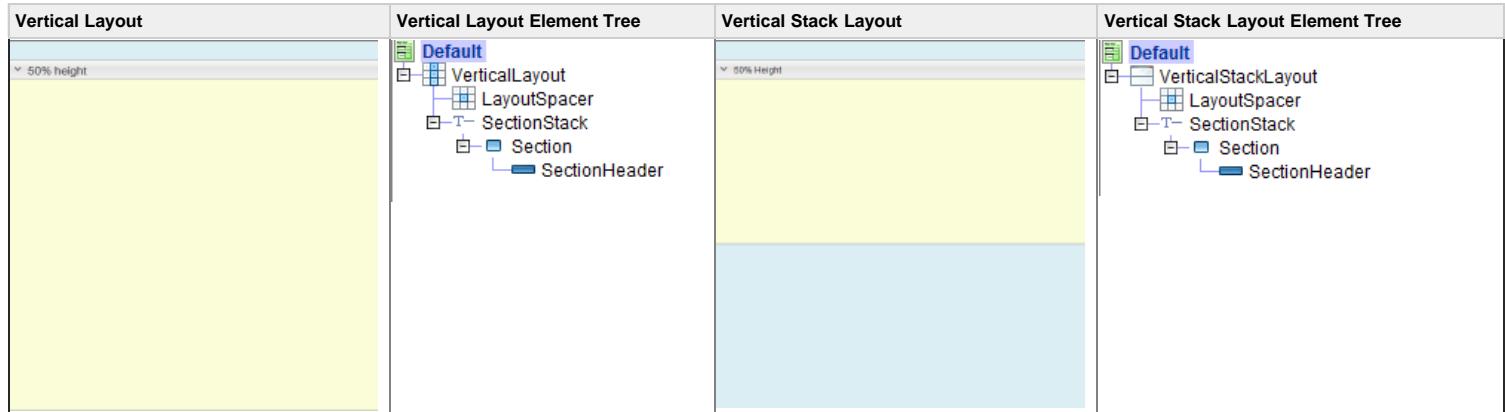
- By default, vertical layouts have a height of 1 px. If your dynamic forms, such as ones displayed through a form frame, are not appearing runtime, ensure that there is a height specified on at least the parent layout of that form frame, or on the form frame itself.
- You can [display your Web browser's scroll bar](#) when your browser's window is too slow to view the entire application page. These steps involve setting properties to your main layout and other child layouts.

## Vertical Stack Layout

**Note:** The Vertical Stack layout is deprecated. Use the [Vertical](#) or [Horizontal](#) layout.

Vertical Layout and Vertical Stack Layout present child elements in a vertical directive. Each element is presented adjacent to one another vertically. However, for the two vertical layout directives, the height percentage property of their respective child elements results in a different presentation behavior. The child form elements of the Vertical Layout takes on the percentage of the available empty space. Whereas the Vertical Stack Layout's child elements will present the child elements as a percentage of the parent element.

The example below shows a Vertical Layout and a Vertical Stack Layout with a Layout Spacer and Section Stack child elements. The vertical layouts are each shown by the blue areas and the Section Stack is represented by the yellow area. Both the layout properties and the child element properties have exactly the same values. However, the child elements are displayed differently in the two layout directives. The difference is the handling of the child sizing percentage. The Vertical Layout will adjust its children to fill the available space, whereas the stack layouts will always respect the width and height values of the children.



The table below shows the property values for the parent layout and the child elements:

Parent Layout Properties		Child Layout Spacer Properties		Child Section Stack Properties	
Name	Value	Name	Value	Name	Value
Default		Default		Default	
VerticalStackLayout		VerticalStackLayout		VerticalStackLayout	
LayoutSpacer		LayoutSpacer		LayoutSpacer	
SectionStack		SectionStack		SectionStack	
Section		Section		Section	
SectionHeader		SectionHeader		SectionHeader	
Cell Alignment		Height	25px	Header Style	
Context Menu		Name	LayoutSpacer	Height	50%
Dynamic Style		Style		Name	SectionStack
Height	500px	Visible		Visibility Mode	mutex
Name	VerticalStackLayout	Width		Visible	
On Enter				Width	100%
Overflow					
Show Resize Bar	<input type="checkbox"/>				
Style					
Tooltip					
Visible					
Width	500px				

For the example above, the Section Stack element has a width percentage property value of 50% and a LayoutSpacer height value of 25px in both layout directives. For the Vertical Layout, the Section Stack element (area displayed in yellow) takes on the height of the remaining space of its parent (Vertical Layout height - the LayoutSpacer height). Whereas for the Vertical Stack Layout, the child Section Stack height is 50% of the Vertical Stack Layout height regardless of the LayoutSpacer's height.

### Summary

references Variable?	No
references Method?	Yes
references Form?	No
Allowable Child Elements	Button Checkbox Date Field Date Time Dynamic Table Dynamic Document Form Frame Grid Layout Header Horizontal Layout Horizontal Stack Layout HTML Content Hyperlink Image Iterator Label Layout Spacer Menu Item

Password Field
Radio Button Group
Reference Field
Section Stack
Select Field
Separator
Slider
Spinner
Tabset
Text Area
Text Field
Tree
Upload File
Variable
Vertical Layout
Vertical Stack Layout

## Properties

Property	Mandatory/Optional	Comment
Name	Mandatory	Name of the Vertical Stack Layout.
*Cell Alignment	Optional	The cell alignment property functions in conjunction with the height/width properties. The values in the height/width properties must be set for the cell alignment property to function. This value specifies the horizontal and vertical positioning of its children within the Vertical Stack Layout. An undefined value results in a "top-left" alignment.
Dynamic Style	Optional	The value of this property is a method with a Return type <i>String</i> . This method will dynamically change the element's style at runtime based on the instructions returned by the method. The style returned by the method must exist within the current cascading style sheet (css) file that is being used by the application.
Dynamic Label	Optional	Defines a label value that is set dynamically at runtime. Set property to a <i>String</i> variable or a script that returns a <i>String</i> . The static Label property is considered as default until the action returns a value.
Dynamic Label Style	Optional	The value of this property is a method with a Return type <i>String</i> . In conjunction with the Variable property, this method will dynamically change the style of the label at runtime based on the instructions returned by the method. The label style returned by the method must exist within the current css file that is being used by the application.
Dynamic Tooltip	Optional	Defines a dynamic tooltip text that is displayed in a hover box when the cursor hovers the element. Set property to a <i>String</i> variable or a script with a Return type <i>String</i> . The static Tooltip property is considered as default until the action returns a value.
Dynamic Tooltip Style	Optional	This property dynamically changes the style of the Dynamic Tooltip. Set property to a script with a Return type <i>String</i> that is a cascading style sheet (css) name found in the stylesheet used by the application.
Edge	Optional	Select this checkbox to show an edge around a layout element. See the other Edge properties in this table to customize your edged layout.
Edge: Content Offset	Optional	This property denotes the amount the contained elements are offset to. The property defaults to <b>Edge: Size</b> if the value of the <b>Edge: Content Offset</b> is 0. Set this property to less than <b>Edge: Size</b> to allow the contained elements to overlap the edge and corner media.
Edge: Custom Sides	Optional	Use this property to list custom sides for the border, such as <code>left</code> , <code>top</code> , <code>bottom</code> , and <code>right</code> . Some examples include the following: <ul style="list-style-type: none"> <li>• <code>left</code>, <code>top</code>, <code>bottom</code></li> <li>• <code>top</code>, <code>bottom</code></li> </ul>
Edge: Dynamic Image	Optional	This property contains either a method or variable that sets the edge image dynamically (starts with a <code>/</code> ). If the dynamic image is null, a static image is used for the edge instead. <b>Note:</b> If neither a dynamic, nor static image is specified, the default <code>[SKIN]/edge.gif</code> image is used.
Edge: Fill Center	Optional	Specify whether to show the image background (with the <code>_center</code> extension) in the centre section (that is, behind the decorated layout).
Edge: Size	Optional	Use this property to define the size in pixels for corners and edges.
Edge: Static Image	Optional	This property contains the base name of images for edges. Extensions for each corner or edge piece are added to the image URL before its file extension. For example, a default base name of <code>edge.gif</code> , the top-left corner images is <code>edge_TL.gif</code> . The list of valid extensions is as follows: <ul style="list-style-type: none"> <li>• <code>_TL</code> (top-left)</li> <li>• <code>_TR</code> (top-right)</li> <li>• <code>_BL</code> (bottom-left)</li> <li>• <code>_BR</code> (bottom-right)</li> <li>• <code>_T</code> (top)</li> <li>• <code>_L</code> (left)</li> <li>• <code>_B</code> (bottom)</li> <li>• <code>_R</code> (right)</li> <li>• <code>_center</code> (centre)</li> </ul>
Height	Optional	Height used or consumed by the Vertical Stack Layout. The value is defined as the number of pixels (integer) or as a percentage of page height or of the parent element. For example, 50% of the page height or 200 pixels.
Include in Tab Order	Optional	This property allows you to optionally suppress tabbing to any or all form elements. The <b>Include in Tab Order</b> property overrides the <a href="#">suppress tabbing</a> configuration setting and can take the following values: <ul style="list-style-type: none"> <li>• When set to <code>&lt;Default&gt;</code>, the element is included in tab order according to global rules and its focusable state.</li> <li>• If the property is set to <b>Exclude</b>, the element is excluded from tabbing even if the element is focusable.</li> <li>• If the property is set to <b>Include</b>, the element is included <i>only if it is focusable</i>, even if all items of that type are excluded through the Suppress Tabbing variable.</li> </ul>

	<b>Note:</b> For menu items and images, the <b>Can Focus</b> property affects including the element in the tab order. To ensure that an element is in the tabbing order, select this property.	
<b>Members Margin</b>	Optional	Specifies the number of spaces allowed between grid layouts in the layout stack.
<b>Mouse Over Method</b>	Optional	Click this property's <b>Value</b> field and select a method to invoke when users hover their mouse over a specified element.
<b>On Enter</b>	Optional	The system calls any script defined in this property. The <b>onEnter</b> property is only met if there is an input field (for example, a textfield, selectfield, and so on) that has focus when the <b>Enter</b> key is pressed. This <b>onEnter</b> property is invoked only if the input field does not have an <b>OnEnter</b> property set. This feature can be used with multiple input fields, but only one <b>onEnter</b> method to fire for all fields.
<b>Overflow</b>	Optional	<p>Defines how overflow child form elements are displayed based on the height of the layout. The default is <b>Auto</b>.</p> <p><b>Auto</b> - The child elements that do not fit within the height of the layout defined are shown via a scroll bar. The user can scroll across to see the children at runtime. In other words, the layout will scroll if the child members exceeds its specified size.</p> <p><b>Hidden</b> - The child elements that do not fit within the height of layout are NOT displayed at runtime.</p> <p><b>Visible</b> - If the child field element members size exceeds the specified layout size, the layout will grow to accommodate the child element members.</p>
<b>Show Resize Bar</b>	Optional	If true, a resize bar is created for the Vertical Stack Layout, that allows the user to resize the vertical layout by dragging it, and show/hide layout by clicking it. The orientation of the Resize bar for the Vertical Layout yields a left-and-right resize bar. The length of the resize bar is dictated by the height of the parent. Specifically, it is the height of the Vertical Layout parent or the width of the Horizontal Layout parent that determines the resize bar's length. Default is <b>default</b> , which does not show the Resize Bar.
<b>Style</b>	Optional	Choose from styles within the selected stylesheet (located in the Velocity Studio toolbar), which defines the styling of the Vertical Layout such as positioning and color. Default is <b>NONE</b> .
<b>Tooltip</b>	Optional	Defines the tooltip text that is displayed in a hover box when the cursor hovers over the Vertical Stack Layout. Default is empty, which means that a tooltip has not been defined. During runtime, the tooltip may be difficult to find as it is not the only layout on the screen and may cover a large area.
<b>Tooltip Style</b>	Optional	Defines the tooltip style of the text displayed in the hover box when the cursor hovers over the Vertical Stack Layout. The default is empty. Click the <b>Value</b> field to select a tooltip style from the list.
<b>Tooltip Width (px)</b>	Optional	Allows you to change the width of the tooltip popup window statically. Enter the width in pixels. The default value for this property is 0.
<b>Visible</b>	Optional	Defines the user permissions assigned to this layout. Pre-defined system <a href="#">Permissions</a> such as <b>addPerm</b> , <b>deletePerm</b> , <b>editablePerm</b> , <b>attachPerm</b> , <b>executePerm</b> , and <b>visiblePerm</b> can be assigned as well as Object Permissions. Defaults to <b>NONE</b> , which means no permissions assigned.
<b>Width</b>	Optional	Width used or consumed by the Vertical Layout. The value is defined as the number of pixels (integer) or as a percentage of page width or of the parent element. For example, 30% of the page width or 150 pixels.

\*If this element is the first element that appears in the Element Tree pane, you can set the cell alignment. If this element has a parent layout (for example, vstacklayout or hstacklayout with the width and height set) this element uses these values as a base to set its alignment. If the parent layout does not contain a width/height, then this element does not have a base to which it can set its alignment. If the element has a parent that contains width and height settings, but this form is displayed under another layout with different widths, your form alignment is restricted to the parent's width/height settings.

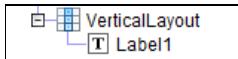
#### Notes:

If you require drawing a border or box around a label or single form item element, wrap the element with Grid Layout instead of Horizontal Layout, Vertical Layout, Horizontal Stack Layout, or Vertical Stack Layout. The following are examples of drawing a border around a label:

- It is recommended that you use a Grid Layout with the **Edge** property selected and a Label element under the layout:



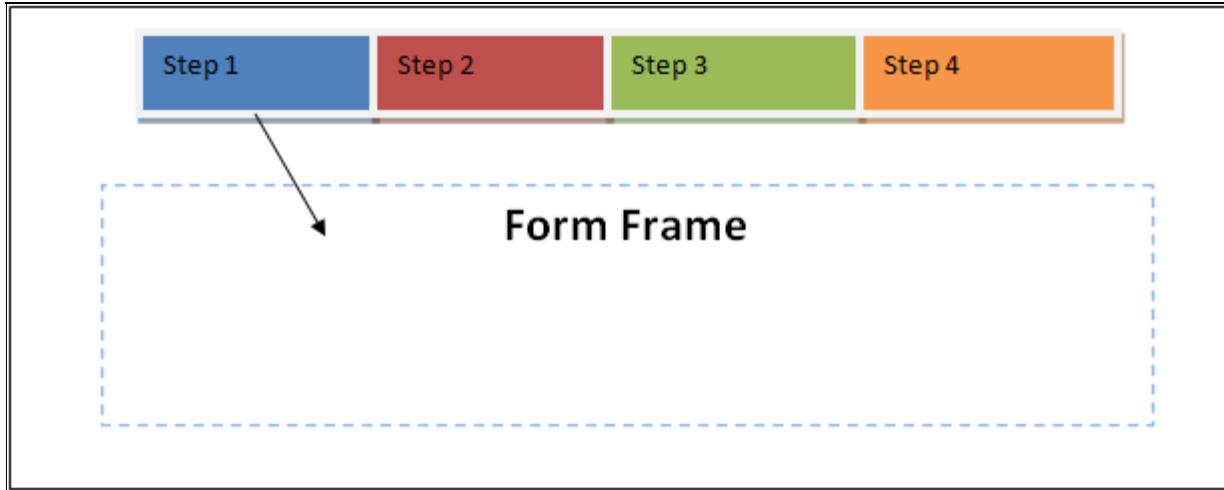
- It is not recommended that you use a Vertical Layout with the **Edge** property selected and a Label element under the layout:



The reason for using the Grid Layout is that at runtime, the label and any form element that is not under Grid Layout is already wrapped with Grid Layout.

# Navigation Bar

The Navigation Bar is one of the metadata objects available under the top-level Presentation item. It displays a series of steps contained within an procedure. Each step (such as an order) is configured as a Navigation Bar Item and is represented as a clickable image. When a user clicks an image, the related data for that step displays in a Form Frame that appears beneath (or when using a vertical setup to the right) of the Navigation Bar image.



The Navigation Bar acts as a progress bar displaying information about each step in a procedure. Each step, represented by a Navigation Bar object, uses a graphic image to display the state of the step. As the end user navigates through the steps of a procedure, the graphic images provides a visual queue indicating the current step, steps that have been completed and those that are outstanding.

The Navigation Bar is configured by extending the system User Interface object. This object uses the same patterning as MVC and includes some predefined variables, methods and metadata to enable the Navigation Bar's functionality.

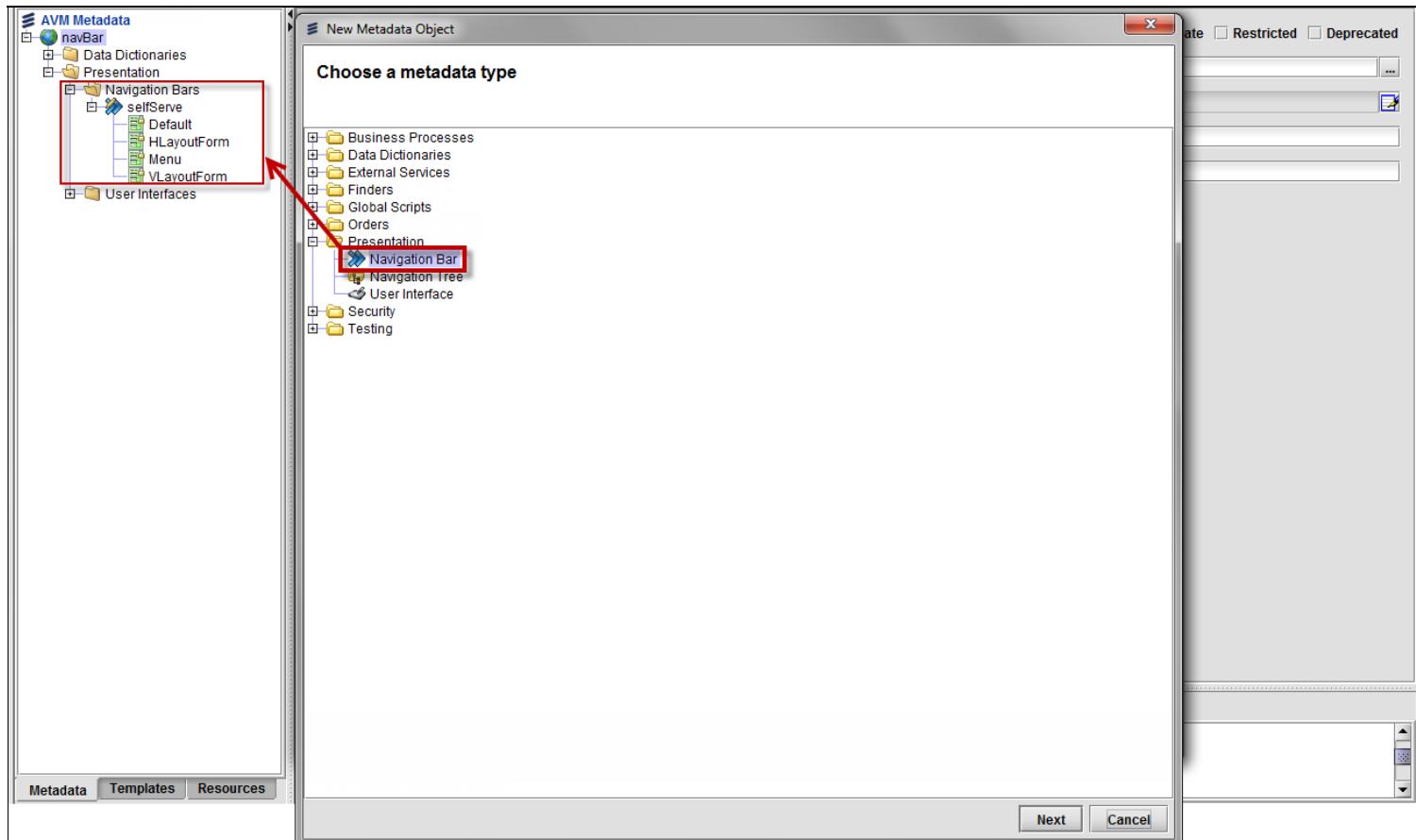
See [Creating a Navigation Bar Object](#) for more information about Navigation Bar configuration.

## Creating a Navigation Bar Object

The Navigation Bar is available under the Presentation node.

To create a Navigation Bar, do the following:

1. Right-click the Namespace and select **New**.
2. In the **New Metadata Object** dialog, expand the Presentation node, select Navigation Bar and click **Next**.
3. In the **New Navigation Bar** dialog, type the Name and Label for the Navigation Bar.
4. In the **Extends** field, by default, the *com.conceptwave.system.NavigationBar* base User Interface object is selected.
5. In the **Help** field, click the help icon. In the Help dialog, type the contents of the help that will appear for the end user.
6. In the **Time duration (minutes)** field, specify an interval where the timer calls the *on Timer* method each time an interval elapses.
7. Click the **Finish** button.



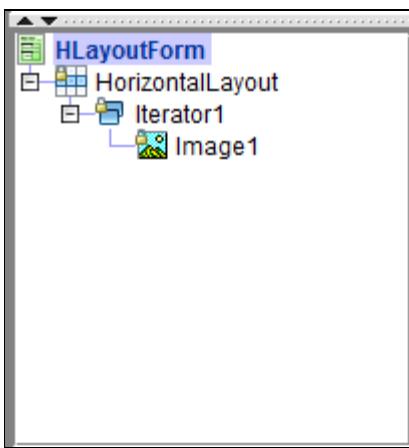
The new Navigation Bar appears beneath the Presentation node. When you expand the Navigation Bar node, there are four default Forms that are created. For more information see, [Navigation Bar Form and Form Elements](#).

## Navigation Bar Form and Form Elements

---

Under the top-level Navigation Bar node, several default Forms are created, two of which control the orientation of the Navigation Bar. The **HLayoutForm** Form controls the horizontal layout and the **VLayoutForm** Form controls the vertical layout. Each Form also controls how the clickable images of the Navigation Bar are controlled. In each of the Forms, two elements appear by default, the Iterator element and the Image element. The Iterator element iterates through the Navigation Bar items (defined and listed in the [Configuration Tab](#)) using the *NavigationItems* variable, see [Navigation Bar Variables](#). For each navigation item, an instance of the image element (the base element used for each navigation item) found under the Iterator is created using the properties of the *NavigationItem*.

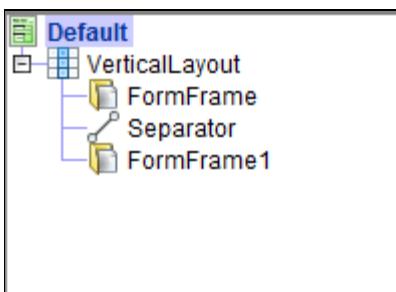
**Caution:** Do not delete the Iterator and Image elements from the HLayoutForm or VLayoutForm Forms or the Navigation Bar will not display at runtime.



### Displaying Navigation Bar Data Using Form Frames

To display data in runtime, you must create a User Interface object and configure its Form using a Form Frame object. In this example, the **FormFrame** displays the Navigation Bar while the **FormFrame1** displays any of the Navigation Bar objects that are returned when an image is clicked.

**Note:** You must create a Form Frame to be able to display data. This is *not* already configured in the Navigation Bar Item



## Navigation Bar Images

---

The Navigation Bar Item displays different states using four images: disabled, focused, over and normal. All these images must exist in the Resource folder of the associated project metadata. An image for each state needs to be created and then copied into the Resources folder in the project directory.

Once the navigation bar item is clicked, the image that is presented is the <image name>\_Focused.png . The state of the image is set on this image. In other words, the image that will be fetched on state change will be <image name>\_Focused\_Over.png when the cursor hovers over the already focused image and <image name>\_Focused\_Down.png when the image is clicked and pressed down on the already focused image. When an image is disabled the <image name>\_Disabled.png is fetched and when the cursor hovers over the disabled image, the image fetched is <image name>\_Over.png. Similarly, when a disabled image is clicked and held down, the <image name>\_Down is fetched. Use the following names when creating your images.

Image	Description
<image name>.png	Displays an image, as the initial image, on loading of the application. This file name is entered in the Image URL field of <a href="#">Configuration Tab</a> of the Navigation Bar object.
<image name>_Over.png	Displays an image when the cursor is hovering over an image and the image is disabled.
<image name>_Down.png	Displays an image when the clicking and holding down the disabled image.
<image name>_Disabled.png	Displays an image as disabled.
<image name>_Focused.png	Displays the current image after clicking an item.
<image name>_Focused_Over.png	Displays the image once the image is focused and the cursor is hovering over the image.
<image name>_Focused_Down.png	Displays the image once the image is focused and the mouse is held down over the image.

## Navigation Bar Focused Item Logic

---

For every navigation item click, the following actions will occur:

1. Calling the NavigationBar.onNavigationClick. This function contains the necessary information to call the right click method associated with the navigation that was just clicked.
  1. If the navigation bar's currentObject variable contains a value, validation on this object will be called based on the following logic:
    - NavigationBar.cwDoValidate is set to true
    - NavigationBar.cwDoValidate is set to null and the clicked item comes after the currently focused item
  2. If the current object is not valid, the transaction is cancelled and the focus is left on the current item.
  3. If the current object is valid or not provided, the focus is set on the clicked item.
  4. The click method is then invoked. If this method returns an object, it will be set as the value for the currentObject variable.

This added functionality will ensure there is no jumping forward/back of the focus on each navigation item.

The focus will only change if:

- o There is a manual call on setFocus().
- o The current object on the screen is valid

## Navigation Bar Configuration Settings

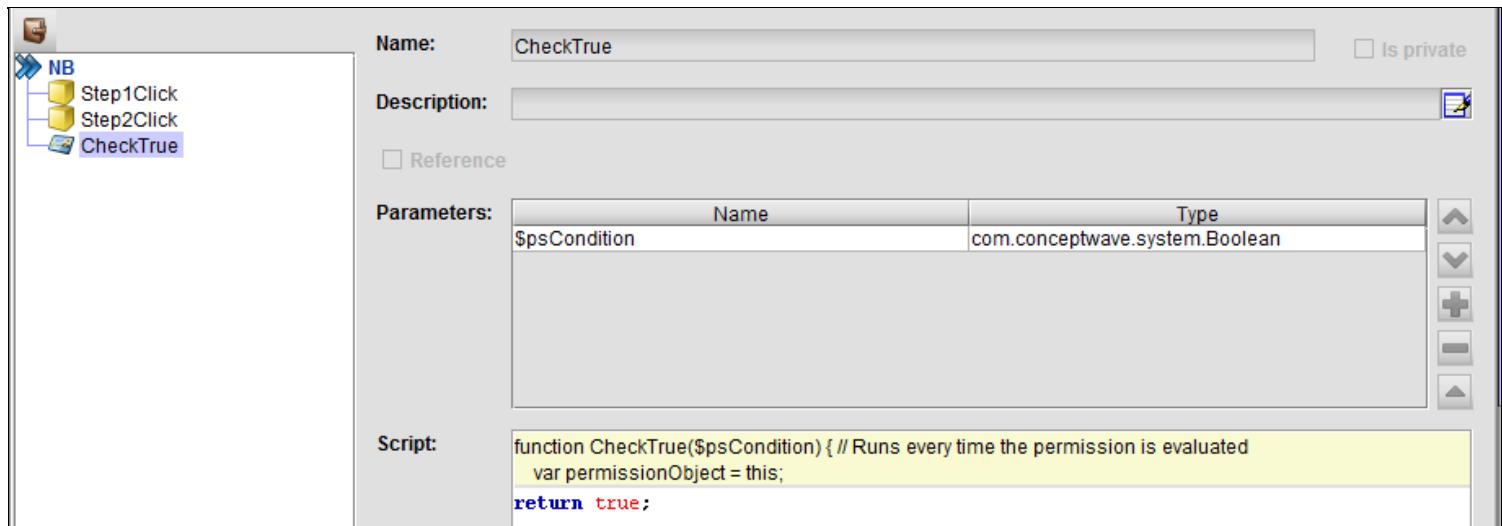
---

Before creating Navigation Bar items, it is recommended that you create [Permissions](#), [Variables](#) and [Methods](#) that will be referenced by the Navigation Bar items. After you have created these items, you can configure the Navigation Bar settings on the [Configuration tab](#).

## Configuring Permissions

Configuring permissions allows particular Navigation Bar item to display or be hidden based on configured privileges. Permission settings are used in two of the Navigation Bar configuration fields, Editable and Visible. Permission control if an image is available in the Navigation Bar at runtime. To configure Navigation Bar permissions, in the Methods tab, right-click the top-level Navigation Bar node and select **New Permissions**.

The simplest permissions example would return a True value for a permission without any conditions:



By configuring a permissions script, declaring variables previously defined, and setting related privileges you can hide information to a user. For example, you are building a Navigation Bar that enables a user to select cable and phone options. The first step of the procedure asks the user to select either cable or phone options. Depending upon his selection, the next step will include additional information for his selection and hidden information about the option that he did not select.

For more information about Permissions, see [Permissions](#).

Name:	<input type="text" value="showBundle"/>
Description:	<input type="text"/>
<input type="checkbox"/> Reference	
Script:	<pre>function showBundle() { // Metadata type method. Can be called by scripts.     return this.isCableEnabled &amp;&amp; this.isPhoneEnabled;  }</pre>
Return:	<input type="button" value="com.conceptwave.system.Boolean"/> 
Privileges:	<input checked="" type="checkbox"/> Privileges and participant operations <ul style="list-style-type: none"><li><input checked="" type="checkbox"/> Add Shared Favorites</li><li><input checked="" type="checkbox"/> Administration App</li><li><input checked="" type="checkbox"/> ConceptWave API Access</li><li><input checked="" type="checkbox"/> Delegate Task</li><li><input checked="" type="checkbox"/> Delete Shared Favorites</li><li><input checked="" type="checkbox"/> Everyone</li><li><input checked="" type="checkbox"/> Get Available</li><li><input checked="" type="checkbox"/> Group Manager</li><li><input checked="" type="checkbox"/> Process Manager Administrator</li><li><input checked="" type="checkbox"/> Return Task</li><li><input checked="" type="checkbox"/> Runtime Administrator</li><li><input checked="" type="checkbox"/> Set Process Priority</li><li><input checked="" type="checkbox"/> Show Error Details</li><li><input checked="" type="checkbox"/> Take on Task</li><li><input checked="" type="checkbox"/> User Profile Administrator</li><li><input checked="" type="checkbox"/> Workgroup Select</li><li><input checked="" type="checkbox"/> Worklist Administrator</li></ul>

## Navigation Bar Variables

Navigation Bar variables are used for a variety of purposes. For example, you can create a binding to a Permission method that controls the privileges of a Document. When you create a Navigation Bar item, several system-defined variables associated with the Navigation Bar are created. You can use these variables or create new variables that refer to objects in your application.

The screenshot shows the 'Variables' tab selected in the top navigation bar of a configuration interface. Below it, a table lists various variables:

Name	Type	Methods	Vis	Opt	Edit
confirmObject	Object	<input type="checkbox"/>			
model	Model	<input type="checkbox"/>			
cwCurrentClicked	String	<input type="checkbox"/>			
currentObject	User Interface	<input type="checkbox"/>			
previousObject	User Interface	<input type="checkbox"/>			
NavigationItems	Object	<input type="checkbox"/>			
isConfirmShown	Boolean	<input checked="" type="checkbox"/>			
isPersonalEnabled	Boolean	<input checked="" type="checkbox"/>			
confDetail	String	<input checked="" type="checkbox"/>			
moreOpt	String	<input checked="" type="checkbox"/>			
persInfo	String	<input checked="" type="checkbox"/>			
billOpt	String	<input checked="" type="checkbox"/>			

Two specific variables are highlighted with callouts:

- System-defined variables (blue box):** A blue box highlights the first four variables: `confirmObject`, `model`, `cwCurrentClicked`, and `currentObject`.
- User-defined variables (black box):** A black box highlights the remaining variables: `previousObject`, `NavigationItems`, `isConfirmShown`, `isPersonalEnabled`, `confDetail`, `moreOpt`, `persInfo`, and `billOpt`.

Below the table, the 'Methods' tab is selected, showing configuration details for the `confirmObject` variable:

**General** tab (selected):

- Name:** confirmObject
- Data type:** com.conceptwave.system.Object
- Type error code:** [empty]
- Mandatory error code:** [empty]
- Element properties:** <Inherited>

**Methods** tab:

Name	Value
[empty]	[empty]

The following table describes the system-defined variables for the Navigation Bar.

Variable	Type	Description
<code>cwCurrentClicked</code>	String	The Method to call. Refers to the name of the navigation item click action.
<code>currentObject</code>	User Interface	The current object on the screen.
<code>previousObject</code>	User Interface	The previous object on the screen.
<code>NavigationItems</code>	Object	The list of Navigation Bar Items in the Configuration tab.

Of special note is the **NavigationItems** variable. This variable is of type Object and refers to all of the navigation items (children items) created in the Navigation Bar item. By referencing this variable in the Iterator element of the HLayoutForm Form, the variable is bound to the Navigation Bar children items that reside in the Navigation Bar.

Name	Value
AutoSave	false
Cell Height	0
Cell Width	0
Double Click Method	
Editable	
Header Height	0
Header Style	
Height	
Hide Header	false
Hover Method	
Hover Style	
Name	Iterator1
On Enter	
Show Filter	false
Style	
Tooltip	
Variable	NavigationItems
Visible	
Width	

## Navigation Bar Methods

Navigation Bar Methods tab defines methods that contain JavaScript functions. You can use several system-defined or user-defined methods to configure information for the Navigation Bar. The following system Navigation Bar Method types are available:

Method	Description
<b>Script</b>	Specifies a defined JavaScript function that can be defined in User Interface and explicitly invoked by other methods or Form Elements.
<b>Permissions*</b>	A permission method that can be associated to properties of Form elements to determine participant's access of the Form Element such as <i>Visible</i> , <i>Optional</i> , and <i>Editable</i> .
<b>User Action*</b>	User Action method that implements the response of user actions, such as clicking a button or image, by returning an object presented in a specified Form to the user.
<b>onNavigationClick</b>	This method controls, by default, the display of the previous and the current objects based on click actions performed in the interface at runtime.
<b>getCurrentObject</b>	This method returns the current content model object displayed in the interface at runtime.
<b>getPreviousObject</b>	This method returns the previous object variable.
<b>setCurrentObject</b>	This method sets the current object variable.
<b>setpreviousObject</b>	This method sets the previous object variable.

\* When the user action only contains a JavaScript, these methods behave the same as the Script Method.

The system-defined Method, onNavigationClick, is important because it assigns the state of objects that are displayed at runtime in the user interface. The code in this Method defines two states; previous object and the current object. The *currentObject* and *previousObject* refer to the system-defined variables located on the Variables tab.

```
if(this.cwCurrentClicked !=null){  
//invokeMethod performs a lookup for the click method  
//objects returned from this call would be saved as  
//the current object for this navigation bar  
var method = "this."+this.cwCurrentClicked+"()";  
var cObj = eval(method);  
  
if(cObj!=null){  
this.previousObject = this.currentObject;  
this.currentObject = cObj;  
}  
//this variable must be reset after this method is called  
//  
this.cwCurrentClicked = null;  
}
```

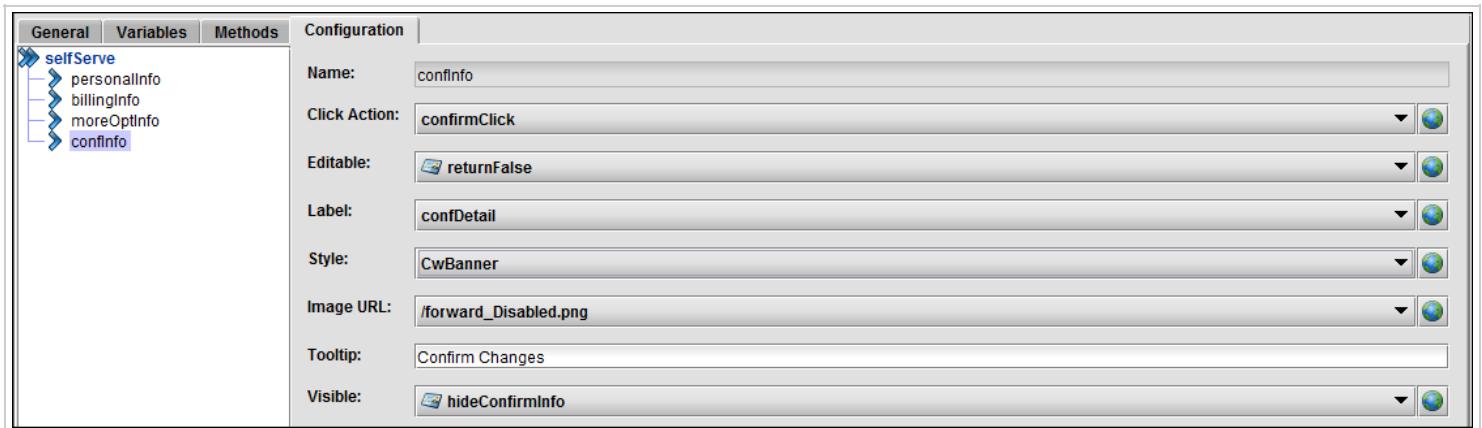
When the user clicks an image in the user interface, the objects that are returned from this call are saved as the current object. The *previousObject* variable is reset to the last current object. For example, if the user is at the third step of an operation, this step is defined as the current state. If they decide to navigate back to the second step, this becomes the current state and the third step is represented as the previous state.

## Configuring the Properties of the Navigation Bar Item

The Configuration tab enables you to create navigation items, each representing a step in a procedure.



For example, you can create a Navigation Bar that consists of the following items: personal information, billing information, more information and confirmation. These four steps are child items of the Navigation Bar, but together create a single end-to-end procedure.



The following fields are available to configure each Navigation Bar child item.

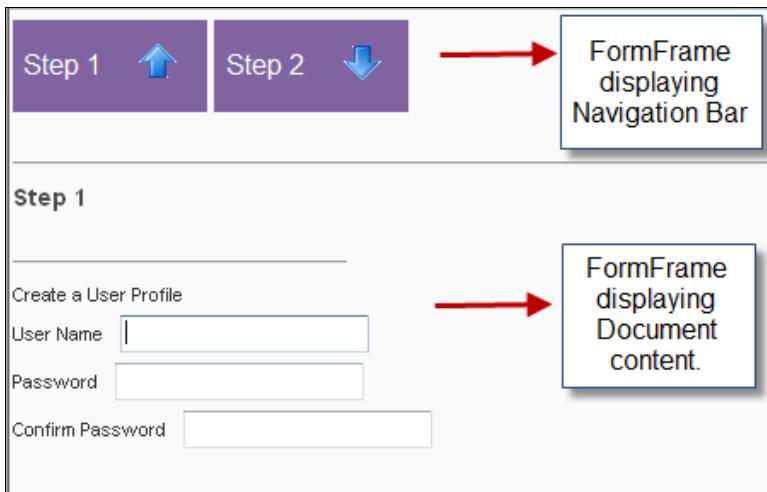
Field	Description
Name	Type the name of the Navigation Bar item.
Click Action	Select a Method from the list menu. This method refers to a script, which provides a description of what will display in the user interface when it is clicked.
Editable	Controls the click action of the related Method, that is configured using permissions. Provides edit permissions.
Label	Select a Variable of type String from the list menu. Assigns a value to the NavigationItem label.
Style	Select a style from the menu. Selecting a style sets CSS properties. For more information about available configuring a CSS, see <a href="#">URL Mapping</a> .
Image URL	Select an image from the menu. Refers to the graphic that displays the initial default image. See <a href="#">Navigation Bar Images</a> for more information.
Tooltip	Type the text of the tooltip for the object.
Visible	Select a Method from the menu. This method refers to a script, which determines whether the NavigationItem is visible or not.

**Note:** Before configuring the child Navigation Bar objects, you need to create permissions, variables, methods, images and objects.

## Setting Up a Navigation Bar

The following example, explains the steps required to setup a Navigation Bar. This example assumes that you are configuring the Navigation Bar using Documents, Navigation Bar Object and User Interface Object. You also need to have configured images that display at runtime.

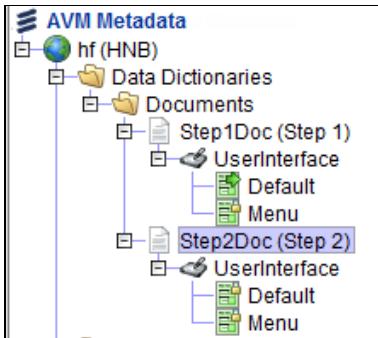
The result of this setup will create the following Navigation Bar and User Interface at runtime. The following diagram shows a user interface that contains two form frames at runtime. The top form frame contains the navigation bar containing two steps, Step 1 and Step 2. The second form frame contains the content triggered by the navigation bar step. This runtime example shows Step 1 enabled and its associated Default form showing the content of the document user interface (named Step1Doc).



### Creating a Document

The Navigation Bar has been designed to work with Document User Interfaces. In this example, the **Default** form of the Document UserInterface will be triggered when the user clicks one of the navigation bar steps. In the Navigation Bar object, the Default form will be called on one of the User Action methods which will be assigned to one of the Navigation Bar Items:

1. Add a **Document User Interface** object by right-clicking the **Data Dictionaries** node of your Namespace and selecting **Documents**.

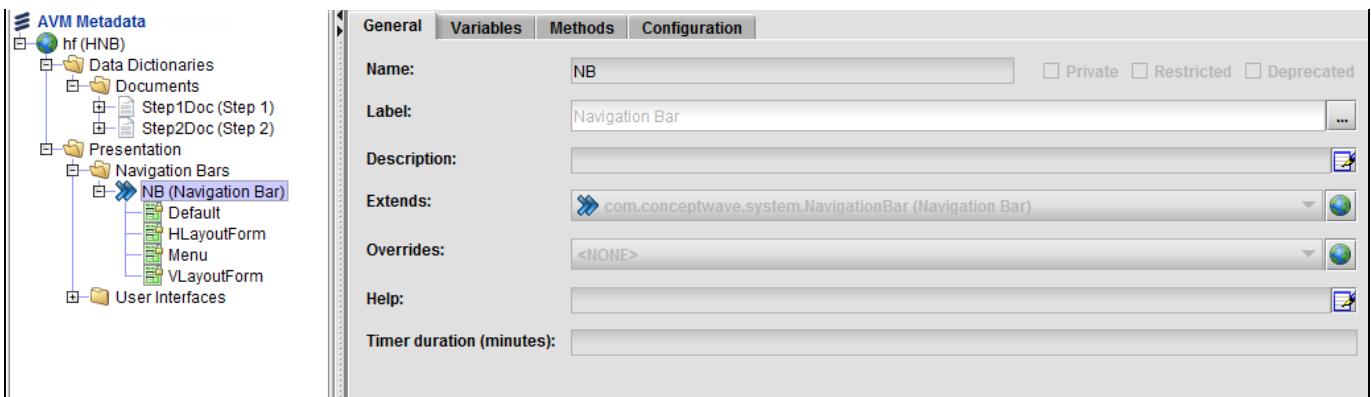


2. More than one Document may be needed depending upon the complexity of your Navigation Bar procedure. Each Document contains forms that are used to display the user interface presentation of the Document. You can configure a form to display information that shows in one of the Navigation Bar steps. In this example, the User Action method of the navigation bar item will trigger the Default form of these documents and a document UserInterface is created for each navigation bar item.

### Creating a Navigation Bar Object

You would use the Navigation Bar object for to configure Variables, Methods and your Navigation Bar Items. This Navigation Bar object is later called in the user interface FormFrame.

1. Create a **Navigation Bar** object by right-clicking the **Presentation** node of your Namespace and selecting **New Navigation Bar** (*com.conceptwave.system.NavigationBar*).



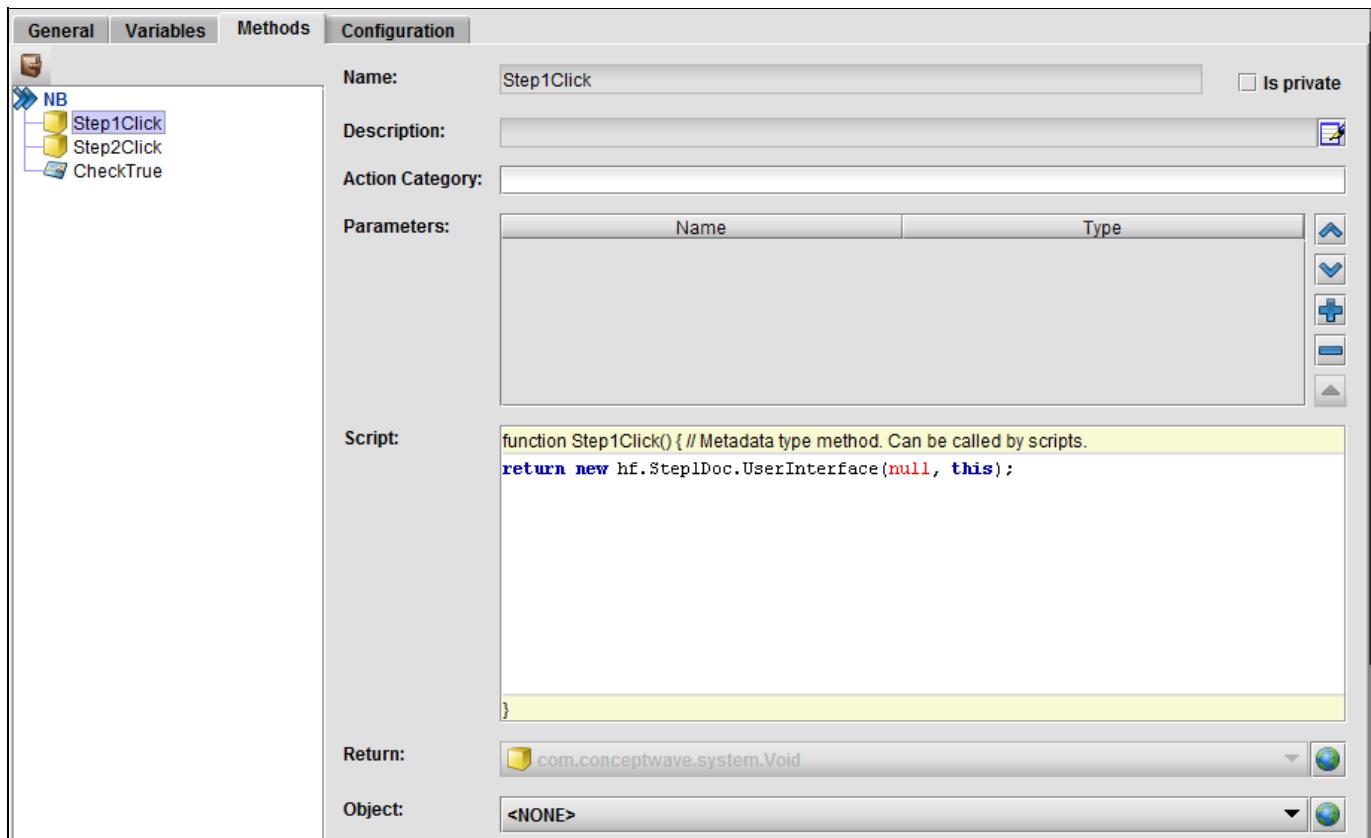
Four tabs appear in the Navigation Bar item: *General*, *Variables*, *Methods*, *Configuration* that are used to configure the Navigation Bar.

2. **Variables:** The *CurrentObject* variable appears in the system variable list. This variable will be used by the *onNavigationClick* method to call the object to which to display in the FormFrame of the UserInterface. In this example, no additional [Navigation Bar Variables](#) have been added.
3. Create a **New User Action** click method. For this example, add a script that returns the document that was created earlier. This script will be added to the Navigation Bar Item's **Click Action** field.

To create Navigation Bar Methods:

1. Navigate to the **Methods** tab, and right-click the node. Select **New User Action** method.
2. A methods form is displayed, enter a **Name** for this new user action method and **Script**. The **Script** must return a user interface. In the script field, type:

```
return new <node>.document_name.UserInterface(null, this);
```



This script will be linked to the Navigation Bar Item's **Click Action** property. At runtime when the Navigation Bar Item is clicked, it will return the **Default** form of the UserInterface document.

In the click method above, the return statement above will set the *currentObject* system User Interface variable. If a return statement is not used, the *currentObject* variable must be set. Another way to assign the *currentObject* variable is by creating a *UserInterface* variable (like *VarStep1*) and assigning it to the *currentObject* the *Step1Click* script:

```
if (this.VarStep1 == null)
```

```

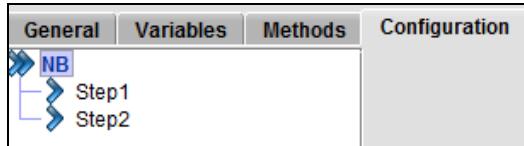
this.VarStep1 == new hf.Step1Doc.UserInterface(null, this);
this.currentObject = this.VarStep1

```

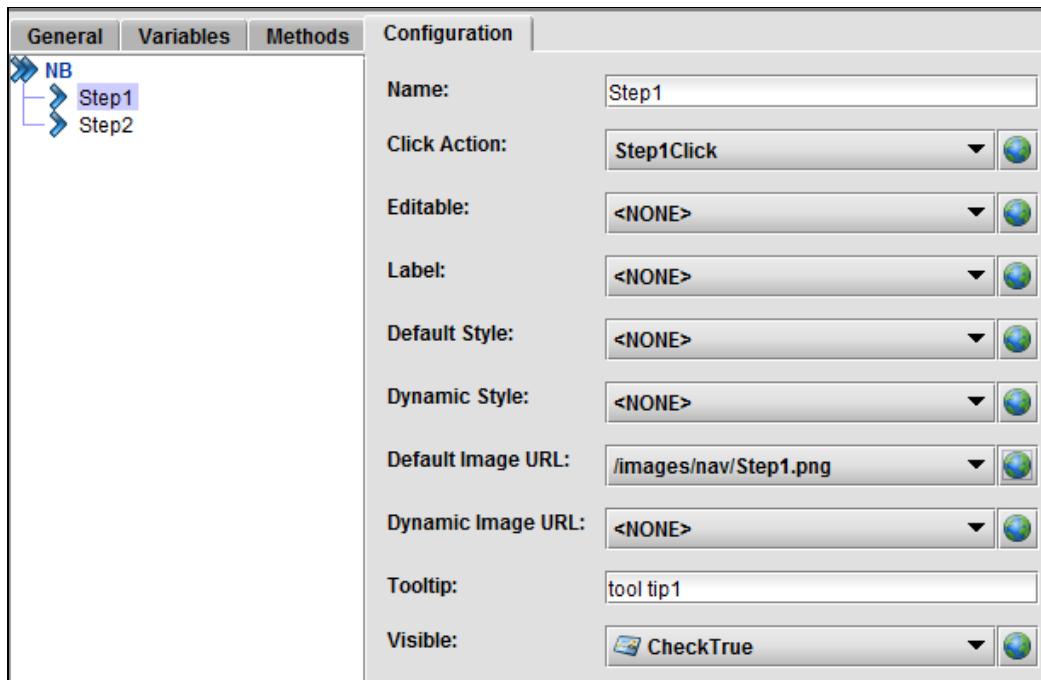
3. Continue to configure methods for all of the actions required. For this example, a similar User Action Method was created for Step 2 with the following script: `return new hf.Step2Doc.UserInterface(null, this);`
4. Create and configure a **Navigation Bar Item**.

The Configuration tab enables you to create Navigation Bar Items and then configure them. You can think of each item as a single step of a procedure.

1. From the **Configuration** tab, right-click the Navigation Bar node and select **New Navigation Bar Item**.
2. In the **New Navigation Bar Item** dialog, type the name of the item and click **Finish**.



3. Click the item to view its properties and then configure each property. Refer to the [Navigation Bar Configuration Tab](#) document for more details on the configuration of the Navigation Bar Items.



The **Click Action** property is binded to the Click Action method `Step1Click` created previously. The **Default Image URL** is assigned to the `Step1` image as described in the [Navigation Bar Images](#) document and the **Visible** property is binded to the `CheckTrue` [permission](#) that returns a `True` value.

## Creating a User Interface Object

To display the Navigation Bar and related information at runtime, you need to create a User Interface object. This object should contain two Form Frame elements. One Form Frame will contain the Navigation Bar and the second is used to display the data displayed by each Navigation Bar item.

1. Add a **User Interface** object by right-clicking the **Presentation** node of your Namespace and selecting **New User Interface** (`com.conceptwave.system.UserInterface`).

2. In the User Interface object, create a Navigation Bar variable that points to the Navigation Bar object created above (for this example, the **Data Type** points to **hf.NB**).

3. Within the User Interface, the Navigation Bar variable must be initialized. One way for initializing the variable is by overriding the **onInit** method. Click the **Methods** tab and override the **onInit** method to initialize the Navigation Bar variable.

4. To the **Default** form of the User Interface object, add the following Form Elements:

- VerticalLayout**
- FormFrame**
- Separator**
- FormFrame**



5. The first Form Frame is where the Navigation Bar displays. For the first FormFrame Element, first assign the Navigation Bar variable to the **Variable** property. Second, depending on the desired horizontal or vertical orientation, assign either the **HLayoutForm** or **VLayoutForm** of the Navigation Bar presentation object to the **Form** property.

<b>Default</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Cell Alignment</td> <td></td> </tr> <tr> <td>External URL</td> <td></td> </tr> <tr> <td><b>Form</b></td> <td>NBNav.Forms.HLayoutForm</td> </tr> <tr> <td>Height</td> <td>25px</td> </tr> <tr> <td>Name</td> <td>FormFrame</td> </tr> <tr> <td>On Enter</td> <td></td> </tr> <tr> <td>Overflow</td> <td></td> </tr> <tr> <td>Show Resize Bar</td> <td><input type="checkbox"/></td> </tr> <tr> <td>Style</td> <td></td> </tr> <tr> <td>Tooltip</td> <td></td> </tr> <tr> <td>Unmanaged</td> <td><input type="checkbox"/></td> </tr> <tr> <td><b>Variable</b></td> <td>NBNav</td> </tr> <tr> <td>Visible</td> <td></td> </tr> <tr> <td>Width</td> <td>100px</td> </tr> </tbody> </table>	Name	Value	Cell Alignment		External URL		<b>Form</b>	NBNav.Forms.HLayoutForm	Height	25px	Name	FormFrame	On Enter		Overflow		Show Resize Bar	<input type="checkbox"/>	Style		Tooltip		Unmanaged	<input type="checkbox"/>	<b>Variable</b>	NBNav	Visible		Width	100px
Name	Value																														
Cell Alignment																															
External URL																															
<b>Form</b>	NBNav.Forms.HLayoutForm																														
Height	25px																														
Name	FormFrame																														
On Enter																															
Overflow																															
Show Resize Bar	<input type="checkbox"/>																														
Style																															
Tooltip																															
Unmanaged	<input type="checkbox"/>																														
<b>Variable</b>	NBNav																														
Visible																															
Width	100px																														

6. For the second FormFrame Element, first assign the currentObject variable of the Navigation Bar object to the variable property. Second, assign the Default form of the Navigation Bar's current object to the Form property.

<b>Default</b>	<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Cell Alignment</td> <td></td> </tr> <tr> <td>External URL</td> <td></td> </tr> <tr> <td><b>Form</b></td> <td>currentObject.Forms.Default</td> </tr> <tr> <td>Height</td> <td>100px</td> </tr> <tr> <td>Name</td> <td>FormFrame1</td> </tr> <tr> <td>On Enter</td> <td></td> </tr> <tr> <td>Overflow</td> <td></td> </tr> <tr> <td>Show Resize Bar</td> <td><input type="checkbox"/></td> </tr> <tr> <td>Style</td> <td></td> </tr> <tr> <td>Tooltip</td> <td></td> </tr> <tr> <td>Unmanaged</td> <td><input type="checkbox"/></td> </tr> <tr> <td><b>Variable</b></td> <td>NBNav.currentObject</td> </tr> <tr> <td>Visible</td> <td></td> </tr> <tr> <td>Width</td> <td>500px</td> </tr> </tbody> </table>	Name	Value	Cell Alignment		External URL		<b>Form</b>	currentObject.Forms.Default	Height	100px	Name	FormFrame1	On Enter		Overflow		Show Resize Bar	<input type="checkbox"/>	Style		Tooltip		Unmanaged	<input type="checkbox"/>	<b>Variable</b>	NBNav.currentObject	Visible		Width	500px
Name	Value																														
Cell Alignment																															
External URL																															
<b>Form</b>	currentObject.Forms.Default																														
Height	100px																														
Name	FormFrame1																														
On Enter																															
Overflow																															
Show Resize Bar	<input type="checkbox"/>																														
Style																															
Tooltip																															
Unmanaged	<input type="checkbox"/>																														
<b>Variable</b>	NBNav.currentObject																														
Visible																															
Width	500px																														

## Setting Up a Navigation Bar

---

The following example, explains the steps required to setup a Navigation Bar. This example assumes that you are configuring the Navigation Bar using Documents, Navigation Bar and User Interface items. You also need to have configured images that display at runtime.

### Data Type and Documents

To create a Data Type and Documents:

1. In Velocity Studio, under the Namespace, add a Data Type object of type String.
2. Under the Data Type, add a Document object. More than one Document may be needed depending upon the complexity of your Navigation Bar procedure. Each Document contains Forms that are used to display the User Interface presentation of the Document. You can configure a Form to display information that shows in one of the Navigation Bar steps (that is, use a Label Form Element to display a title or instruction of a step).

### Navigation Bar Item

The Navigation Bar Item displays as a node in the navigation tree. You can add child Navigation Bar items that nest under the top Navigation Bar node. See [Navigation Bar Configuration](#).

To create a Navigation Bar:

1. Add a Navigation Bar item by right-clicking the Namespace and then select **New**. The New Metadata Object opens.
2. From the **New Metadata Object** wizard, expand the **Presentations** node and select the **Navigation Bar** and then select **Next**. The New Navigation Bar wizard appears.
3. In the New Navigation Bar dialog, type the **Name**, **Label** and **Description** for the Navigation Bar and click **Finish**.
4. In the **Navigation** pane, click the **Navigation Bar** item that you created.
5. Four tabs appear in the Navigation Bar item: **General**, **Variables**, **Methods**, **Configuration** that are used to configure the Navigation Bar.

### Navigation Bar Variables

Navigation Bar variables are used for a variety of purposes. For example, you can create a binding to a Permission method that controls the privileges of a Document.

To configure Navigation Bar Variables:

1. On the **Variables** tab, click the **Add** toolbar button to create a new variable.
2. In the **Add Element** dialog, locate the metadata object that you want to add and click **Save**. For our purposes, we are going to focus on the existing **NavigationItems** variable of type Object appears in the list of variables. This variable controls all of the Navigation Bar child items that appear under the Navigation Bar top-level item.

### Navigation Bar Methods

Methods contain scripts that perform an action. For our example, we are going to add a script that returns the document that we created earlier.

To create Navigation Bar Methods:

1. On the **Methods** tab, right-click the Navigation Bar item node to view the available methods.
2. Select the **New User Action Method**.
3. In the **Name** field, type a name for this new user action method.
4. In the **Script** field, type `return new <document_name>(null, this);`
5. Continue to configure methods for all of the actions required. The methods are used to configure some of the properties listed on the Configuration Tab.

### Creating Permissions

Any New Permission methods that are added are not invoked unless used on the Navigation Bar Configuration tab. For example, the permission can bind to a variable, which then binds to a Document.

To add permissions:

1. On the **Methods** tab, right-click the Navigation Bar item node and select **New Permissions**.
2. In the **New Permission** dialog, type the name of the permission and click **Finish**. The properties for the permission method appears.
3. In the **script** area, type the script that you want to use. For the purpose of this example, type `return this. isdocumentEnabled;`

### Navigation Bar Configuration

The Configuration tab enables you to create Navigation Bar child items and then configure them. You can think of each item as a single step of a procedure.

1. From the **Configuration** tab, right-click the Navigation Bar node and click **New Navigation Bar Item**.
2. In the **New Navigation Bar Item** dialog, type the name of the item and click **Finish**.
3. Click the item to view its properties and then configure each property.
  - Name - contains the name of the Navigation Bar child item.
  - Click Action - drop-down list contains a list of methods. This is what displays in the User Interface when this item is clicked.
  - Editable - a drop-down list containing methods configured using Permissions.
  - Label - a drop-down list containing a list of variables which assigns a value to the NavigationItem label.
  - Style - A drop-down list containing a list of CSS properties.
  - Image URL - a drop-down list containing a list of graphics.
  - Tooltip - a text field in which you can type a tooltip.
  - Visible - a drop-down list contains a list of methods. Determines if the NavigationItem is visible or not.

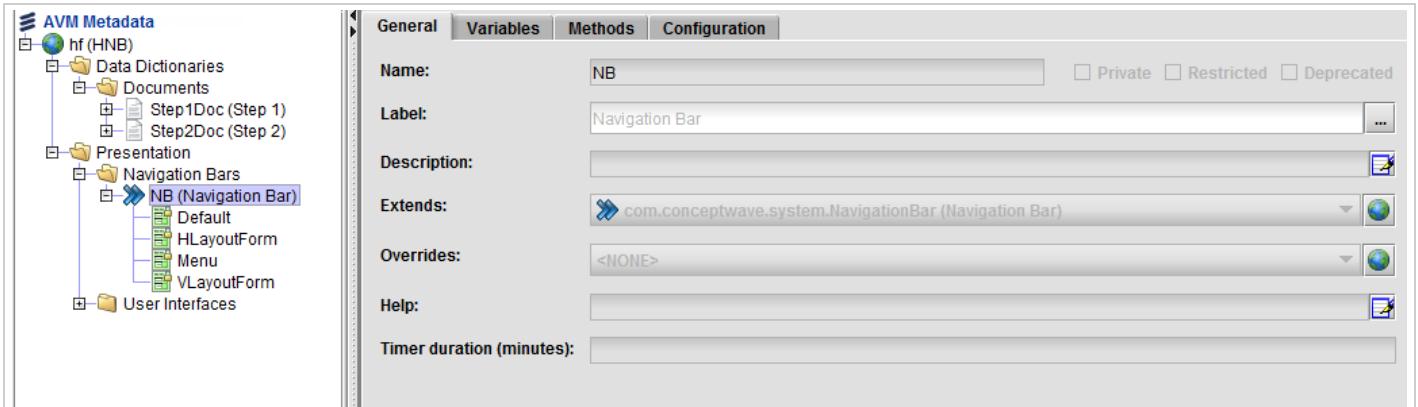
## Configuring the Navigation Bar Layout

The Navigation Bar item contains Forms that configure the orientation of the Navigation Bar. You can choose between the horizontal (HLayoutForm) and vertical (VLayoutForm) display. This example uses the HLayoutForm format. Reference this Form in the User Interface Forms that you will be configuring (next step).

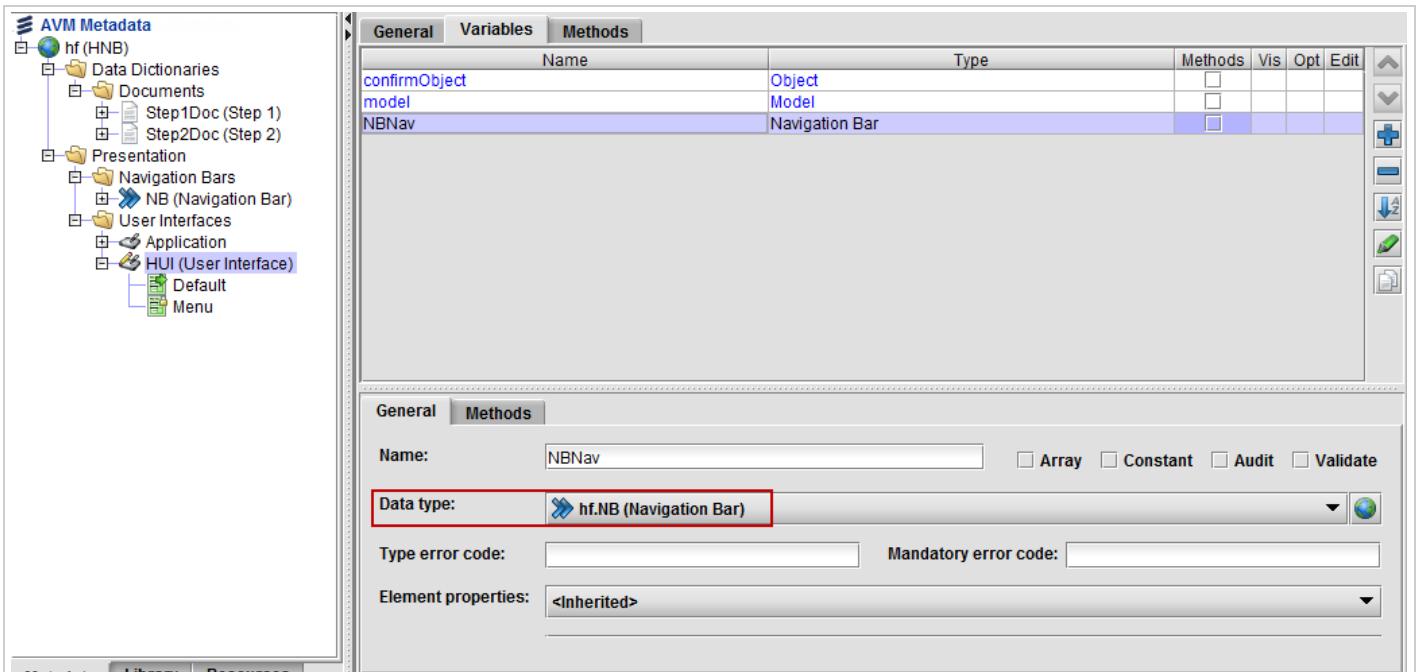
## Creating a User Interface

To display the Navigation Bar and related information at runtime, you need to create a User Interface object. This object should contain two Form Frame elements. One Form Frame will contain the Navigation Bar and the second is used to display the data contained in each Navigation Bar item.

1. Add a **User Interface** and a **Navigation Bar** object by right-clicking the **Presentation** node of your Namespace (using `com.conceptwave.system.UserInterface` and `com.conceptwave.system.NavigationBar`).

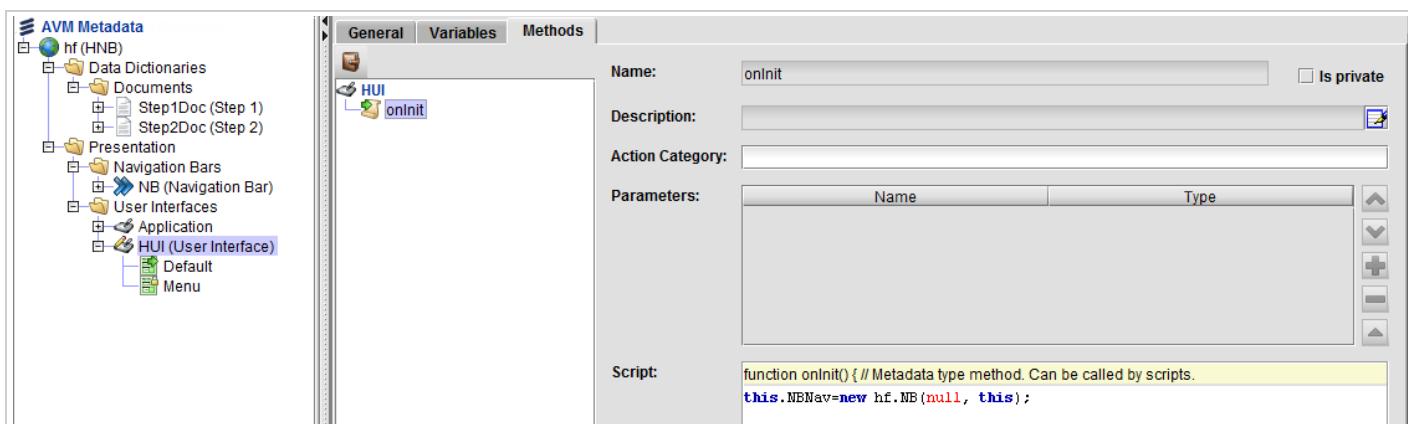


2. In the User Interface object, create a Navigation Bar variable that points to the Navigation Bar object created above (for this example, the **Data Type** points to `NavBar.selfServe`).

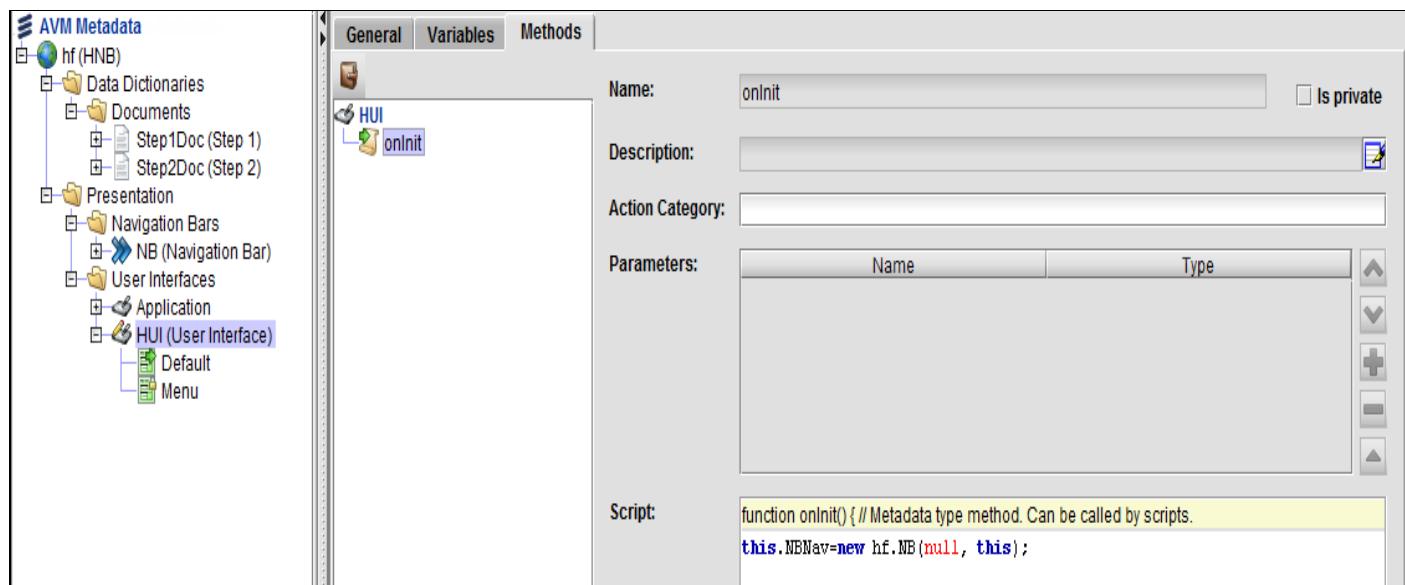


3. Within the User Interface, the navigation bar variable must be initialized. One way for initializing the variable is by overriding the `onInit` method. Click the

Methods tab and override the **onInit** method to initialize the navigation bar variable.



The Navigation Bar object can also be initialized via the LeafInitAction method.



4. To the **Default** form of the User Interface object, add the following Form Elements:

- o **VerticalLayout**
- o **FormFrame**
- o **Separator**
- o **FormFrame**

5. The first Form Frame is where the Navigation Bar displays. For the first FormFrame Element, first assign the Navigation Bar variable to the **Variable** property. Second, depending on the desired horizontal or vertical orientation, assign either the **HLayoutForm** or **VLayoutForm** of the Navigation Bar presentation object to the **Form** property.

Name	Value
Cell Alignment	
External URL	
Form	NBNav.Forms.HLayoutForm
Height	25px
Name	FormFrame
On Enter	
Overflow	
Show Resize Bar	<input type="checkbox"/>
Style	
Tooltip	
Unmanaged	<input type="checkbox"/>
Variable	NBNav
Visible	
Width	100px

6. For the second FormFrame Element, first assign the currentObject variable of the Navigation Bar object to the variable property. Second, assign the Default form of the Navigation Bar's current object to the Form property.

Name	Value
Cell Alignment	
External URL	
Form	currentObject.Forms.Default
Height	100px
Name	FormFrame1
On Enter	
Overflow	
Show Resize Bar	<input type="checkbox"/>
Style	
Tooltip	
Unmanaged	<input type="checkbox"/>
Variable	NBNav.currentObject
Visible	
Width	500px

## Navigation Trees

The Navigation Tree is the metadata object type for presentation purposes that can model nodes of a tree-structure, which can be presented and navigated by the user in a [Tree Form Element](#) by default. Different nodes in the Navigation Trees can represent various metadata objects in your project, which provide the navigational means to navigate through nested models and access details by clicking the corresponding node.

The screenshot shows a 'Customer Tree' interface. On the left is a tree view with nodes like 'customerTree', 'Daniel Smith', 'Bobby Smith' (selected), 'Mary Smith', 'John Smith', 'Mary Smith', and another 'Mary Smith'. On the right is a 'customer' form with fields: 'System document ID' (16100001), 'First Name' (Bobby), 'lastName' (Smith), 'active' (True selected), 'duedate' (5/28/2010), and a 'Save' button at the bottom.

Common uses of the Navigation Tree include:

- **Presenting Hierarchical Structures** - When there is a data model that is hierarchical, even dynamically and/or recursively hierarchical, Navigation Tree is the ideal metadata object to present this information to user. For example, Product and Service Catalogs are often hierarchical with service components packaged into a sellable product, and with products bundled into an offer. Indeed, the **Product Catalog** uses the **Navigation Tree** to enable users to browse and select desired items from the catalog.
- **Presenting Order** - [Orders](#) are typically non-flat in structure due to the need to model different aspects such as product items, customer-oriented attributes as well as service-level and network-related provisioning attributes. It is not uncommon that Orders are deeply and broadly nested, particularly when including related entities such as subscriber and dwelling information. Navigation Tree is the best means to present an Order to a user -- to organize and display only the requested portions of the Order in detail as user navigates the Order tree.
- **Loosely-related entities composed into a single page** - Stitch together a management page by artificially structuring these information into tree nodes of the Navigation Tree for easy navigational purposes. For example, the [User Profile Manager](#) in System Administration Application, which manages entities such as users, groups and privileges, is implemented with a Navigation Tree.

In the latest case concerning User Profile Manager, you can study its composition in the **Library** tab of **Navigation** pane in Velocity Studio, located in **cwf\_up > Presentation > Navigation Trees > upadminTree**.

### Modelling Overview

The modelling of the Navigation Tree is accomplished by creating hierarchical tree nodes from root Navigation Tree metadata object, which is considered the root node of the Navigation Tree. There are several types of child nodes that can be created:

Node Type	Description

<b>Tree Node</b>	A regular child node that models a static node in the Navigation Tree. The defined node is exactly the same in runtime as in design time.
<b>Finder Node</b>	A node that generates child nodes at runtime based on Finder results. This enables Navigation Tree to represent dynamic children. This is often needed when a parent node is to contain multiple and dynamic number of "items" beneath it. As an example, Navigation Tree that shows all subscribers under an account.
<b>Order Item Node</b>	A child node that is similar to a Tree Node, to present an item of an Order. This item may be an Order Document or an Order Collection; in the latter case, multiple children nodes may be generated at runtime. An Order Item Node can only be created when there is an ancestor node that represents an Order metadata object.
<b>Data Structure Item Node</b>	A child node that is similar to a Tree Node, to present an item of a Data Structure. Depending on the type of Data Structure Item, multiple children nodes may be generated at runtime. A Data Structure Item Node can only be created when there is an ancestor node that represents a Data Structure metadata object.
<b>Recursive Node</b>	A node that refers to any ancestor node, to represent a recursive tree structure. For example, in a catalog tree, a product may contain other products, and recursively so; in this case, a recursive node can be created underneath the product "parent" node, while referring the parent node as the object to recur. A Recursive Node may also refer to any Root Node of other Navigation Tree, where the metadata of the Recursive Node assumes the metadata of the referenced tree.

The nodes hierarchy defined in a Navigation Tree simply represents the tree structure that is to be displayed. It should not be used as the modelling of a hierarchical object -- that should be accomplished with [Data Structure](#) and [Order](#) metadata objects.

When showing Data Structure or Order as Navigation Tree, the structure of the Navigation Tree do not necessarily have to mirror the true structure of the model. The Navigation Tree may be customized to deepen, flatten or unionize branches of the true model to the user's browsing convenience. Shown below is an illustrative example:

Model tree		Navigation Tree					
Order	Peripherals	<table border="1"> <tr> <td>Order</td> </tr> <tr> <td>Equipment</td> </tr> <tr> <td>1. Wireless Router</td> </tr> <tr> <td>2. Cable Box</td> </tr> <tr> <td>3. Tivo</td> </tr> </table>	Order	Equipment	1. Wireless Router	2. Cable Box	3. Tivo
Order							
Equipment							
1. Wireless Router							
2. Cable Box							
3. Tivo							

In essence, the Navigation Tree provides the presentation liberty to show a custom hierarchy of the models that are otherwise strictly-defined at the modelling layer.

There is no limit on the number of nodes or depth in a Navigation Tree.

### Presentation Overview

The Navigation Tree is an extended metadata object type of top-level User Interface. The Navigation Tree is, indeed, a full User Interface object that is typically configured to encapsulate the complete navigational user experience. By default, at the **Default** Form of the root node of the Navigation Tree, a [Tree Form Element](#) is rendered at the left side of the Form to display the tree structure, while the right side of the Form is the *detail pane* which displays node details when corresponding node is clicked in the Tree.

An extended variety of the Navigation Tree is capable of collapsing any child node into its parent node at runtime, as a separate Tab in the parent node's detail pane. This extended variety is referred to as *Navigation Tree with Tabs*.

See [Runtime Navigation Trees at Runtime](#) for details.

### Metadata Object Overview

The following metadata object components constitute the Navigation Tree:

Icon	Component	Description
------	-----------	-------------

	<b>Root Node</b>	The root node of the Navigation Tree, extends from base Navigation Tree object <code>com.conceptwave.system.NavigationTree</code> or <code>com.conceptwave.system.NavigationTreeWithTabs</code> . The latter base object enables the Navigation Tree to display children nodes as tabs.
	<b>Tree Node</b>	A regular child tree node of the Navigation Tree, extends from base Tree Node object <code>com.conceptwave.systemTreeNode</code> .
	<b>Order Item Node</b>	A child tree node of the Navigation Tree, extends from base Tree Node object <code>com.conceptwave.systemTreeNode</code> .
	<b>Data Structure Item Node</b>	A child tree node of the Navigation Tree, extends from base Tree Node object <code>com.conceptwave.systemTreeNode</code> .
	<b>Finder Node</b>	A Finder child tree node of the Navigation Tree, extends from base Tree Node object <code>com.conceptwave.systemTreeNode</code> . A Finder tree node always has a child node beneath it, the <i>Finder Child Node</i> node.
	<b>Finder Child Node</b>	The corresponding child tree node of the Finder Node, extends from base Tree Node object <code>com.conceptwave.systemTreeNode</code> . It always has the name <i>FinderChild</i> , which represents the output Document of the Finder as Finder result is returned.
	<b>Recursive Node</b>	A Recursive child tree node of the Navigation Tree. It is not extended from any tree-specific base metadata objects.

In inheritance terms, the base Navigation Tree object `com.conceptwave.system.NavigationTree` extends from `com.conceptwave.system(TreeNode`, which is the base object for Tree Node and Finder Node. Therefore, the root node of the Navigation Tree is a derivation of Tree Node metadata object, with extra metadata defining functionality for the overall tree. In general, the metadata of these components have many common properties of one another, but also have distinctive properties of their own; these commonalities and distinctions are detailed throughout the sections of this documentation topic.

The Navigation Tree with Tabs object `com.conceptwave.system.NavigationTreeWithTabs` extends from `com.conceptwave.system.NavigationTree`. The derived object implements the configurability function to display child node as tab. Note that there is no change to the base metadata object for children nodes in Navigation Tree with Tabs; they are still directly extended from `com.conceptwave.system(TreeNode`.

Furthermore, `com.conceptwave.system(TreeNode` extends from `com.conceptwave.system.UserInterface`, meaning that nodes of the Navigation Tree are derivations of top-level User Interface objects (specifically built for the purpose of presenting navigational tree structure). The collection of these User Interface objects, as nodes, in the Navigation Tree can be traversed in script by using variables `parentNode` and `childNodes`.

Currently, pathing is not supported (for example, `ns.navTreeRoot.childNodeA`). Access nodes in a Navigation Tree by traversing through parent and child nodes using variables in `TreeNode`.

Lastly, there exists Order User Interface `com.conceptwave.system.OrderUserInterface` which is extended from `com.conceptwave.system.NavigationTreeWithTabs`. A Navigation Tree can be based on this metadata object, such that you may create common forms, menus and functionality for Orders.

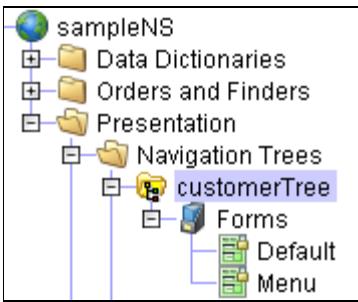
## Create New Navigation Tree

To create a new Navigation Tree, do one of the following:

- In the **Metadata** tab of **Navigation** pane, either:
  1. Right-click a Namespace and select **New ...**
  2. **New Metadata Object** wizard appears as a popup. Expand **Presentation**, select **Navigation Tree**, and then click the **Next** button.
  3. Step two of the wizard appears. Enter the name of Navigation Tree (must conform to JavaScript naming conventions), and populate other fields, as necessary. Then, click the **Finish** button.
- OR
  1. Right-click the **Navigation Trees** folder or the **Presentation** folder in a Namespace, if it is present. Select **New Navigation Tree**.
  2. Follow step 3 from the previous procedure.

After the Navigation Tree is created by the wizard, you can no longer change the base Navigation Tree that extends it. The newly created Navigation Tree is added under **Navigation Trees** folder. This is the root node of the Navigation Tree. Beneath the Navigation

Tree node is the **Forms** container node, which contains the Forms of the Navigation Tree.

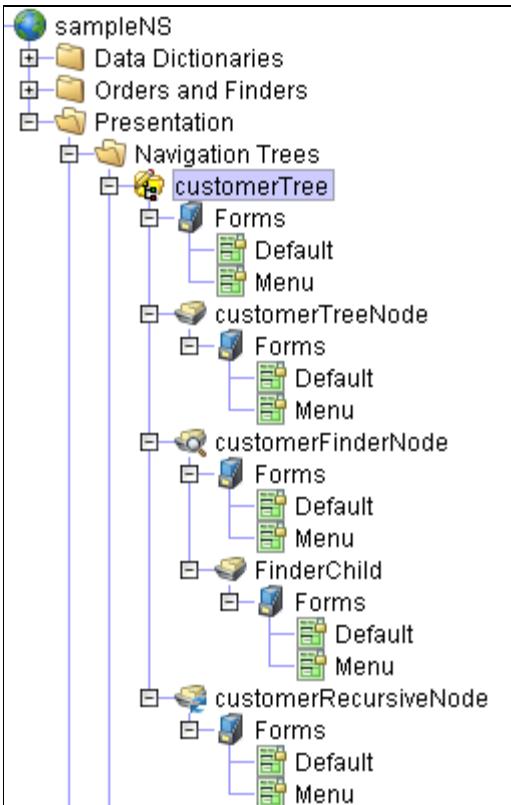


### Create New Node in Navigation Tree

You can create a new child **Tree Node**, **Finder Node** or **Recursive Node** in any (non-Form) node in the Navigation Tree, except for in **Recursive Node** which cannot have child nodes. Conversely, you can only create child **Order Item Node** or **Data Structure Item Node** when the selected node or any of its ancestor nodes has **Object** in **General** tab assigned to an Order or Data Structure respectively. To create a new child node:

- Right-click the node of the Navigation Tree where the child node should be located beneath. Select **New Tree Node**, **New Finder Node**, **New Order Item Node**, **New Data Structure Item Node** or **new Recursive Node** as desired.
- New Metadata Object** wizard appears as a popup. Type in the name of the node (must conform to JavaScript naming conventions), and populate other fields as necessary. Then click **Finish**.

The newly created node is added under the node in the Navigation Tree that is right-clicked. Beneath the new node is the **Forms** container node, which contains the Forms that would represent the new node. The example below illustrates all three types of nodes are added to the root Navigation Tree node. Notice that in the case of Finder Node, a **FinderChild** node is automatically added as its child node to represent the Finder results as nodes.



### Synchronize Action for Data Structures with a Complex Hierarchy

The following example shows the synchronize action for a navigation tree:

- Create three data structures that extended one another. Proceed to add array variables (for example, elements or containers) to each data type.

2. Create a new navigation tree.
3. Set the root **Object** field to the extended data structure.
4. Perform the synchronize action for the tree, which creates tree nodes for both the arrays and instances.
5. Populate the tree with data in `onInit()` of the navigation tree and add instances for each array variable.
6. At runtime, display the tree on the page, which shows all arrays.

## Navigation Tree General Properties

The general properties of the any Navigation Tree node are on the **General** tab.

Because there are different node types of Navigation Tree node, and they have slightly different properties in this tab, the properties are tabularized below with each node type indicating yes/no on its applicability.

- *Root Node*
- *Tree Node*
- *Order Item Node*
- *Data Structure Item Node*
- *Finder Node*
- *Finder Child Node*

On the contrary, the remaining node type, *Recursive Node*, has its properties described in a separate table further below.

For illustrative purposes, the **General** tab of a Navigation Tree root node is shown below.

General	Variables	Navigation Tree Items	Methods
<b>Name:</b> customerTree	<input type="checkbox"/> Private <input type="checkbox"/> Restricted <input type="checkbox"/> Deprecated		
<b>Label:</b> customerTree	<input type="checkbox"/>		
<b>Description:</b> <input type="text"/>	<input type="checkbox"/>		
<b>Help:</b> <input type="text"/>	<input type="checkbox"/>		
<b>Extends:</b> com.conceptwave.system.NavigationTree	<input type="checkbox"/>		
<b>Image:</b> /cwf/group.gif	<input type="checkbox"/> Exclude <input checked="" type="checkbox"/> Auto expand <input type="checkbox"/> Auto save <input type="checkbox"/> Hide expand icon if empty <input type="checkbox"/> Sort children by visual key <input type="checkbox"/> Show children count		
<b>Object:</b> sampleNS.customer (customerLabel)	<input type="checkbox"/>		
<b>Detail UI:</b> sampleNS.customer.UserInterface	<input type="checkbox"/>		
<b>Form:</b> Default	<input type="checkbox"/>		
<b>Table Document:</b> sampleNS.customerTable	<input type="checkbox"/>		
<b>Table Conversion Map:</b> sampleNS.navTreeMap	<input type="checkbox"/>		

Field	Root Node	Tree Node	Order Item Node	DS Item Node	Finder Node	Finder Child Node	Description
Name	Yes	Yes	Yes	Yes	Yes	Yes	Name of the metadata object (used in scripts; must conform to JavaScript naming conventions). Read-only after creation. To change it, use the <b>Rename</b> command in pop-up menu by right-clicking the metadata object.

<b>Private</b>	Yes	No	No	No	No	No	If checked, the metadata object is hidden from display when the metadata is packaged as Product Metadata (that is, library).
<b>Restricted</b>	Yes	No	No	No	No	No	If checked, all JavaScripts contained within the object are blocked from being displayed when the metadata is packaged as Product Metadata (that is, library).
<b>Deprecated</b>	Yes	No	No	No	No	No	If checked, the metadata object is designated for archival purposes only. The object becomes hidden from all metadata object selection lists.
<b>Label</b>	Yes	Yes	Yes	Yes	Yes	Yes	Node's display label in the Navigation Tree, only if the <b>Object</b> does not have a Visual Key / Label, which has precedence as the node's visual key by default (as defined by method <code>onNodeVisualKey()</code> , which may be overridden).
<b>Description</b>	Yes	Yes	Yes	Yes	Yes	Yes	Description of the metadata object for documentation.
<b>Help</b>	Yes	Yes	Yes	Yes	Yes	Yes	Context sensitive user help.
<b>Extends</b>	Yes	Yes	Yes	Yes	Yes	Yes	<p>Base metadata object to derive from.</p> <ul style="list-style-type: none"> <li>For Navigation Tree root node, choose between the regular base object <code>com.conceptwave.system.NavigationTree</code>, or <code>com.conceptwave.system.NavigationTreeWithTabs</code>. The latter base object enables the configurability of Navigation Tree to display children nodes as tabs.</li> <li>For Tree Node, Finder Node and Finder Child Node, it is always extended from <code>com.conceptwave.systemTreeNode</code>.</li> </ul> <p>This field is unchangeable after metadata object creation.</p>
<b>Image</b>	Yes	Yes	Yes	Yes	Yes	Yes	The image icon of the node at the Navigation Tree. If unassigned, node without any children node (that is, leaf) in the Navigation has a Document icon  , while node with children nodes (that is, non-leaf) has a Folder icon  by default. This default behaviour is driven by method <code>onNodeIcon()</code> , which may be overridden. For example, in this method, you may assign different icon images based on state of node (for example, expanded/collapsed).
<b>Exclude</b>	Yes	Yes	Yes	Yes	Yes	Yes	If checked, the node would not be displayed. However, children nodes of this node, if any, will be displayed under this node's parent. This effectively "flattens" the tree at runtime.
<b>Auto expand</b>	Yes	Yes	Yes	Yes	Yes	Yes	<p>If checked, this node is expanded when the Navigation Tree is displayed, showing all children nodes of this node automatically by default. When unchecked, the property is applied to the Navigation Tree node when it contains children nodes. The Miller Column style of presentation does not automatically expand the columns with this parameter.</p> <p>In your metadata, you can set <b>Auto Expand</b> on the Navigation Tree node, which automatically expands this node once tree is open. Once running, you can call on the tree by using <code>setNodeSelected()</code> or use the <code>expandCollapse()</code> API, or both.</p> <p>As an example, if you have a tree with ten nodes, and want to automatically expand the third node through scripting. When you load the tree, node three is automatically expanded by doing the following:</p> <ol style="list-style-type: none"> <li>1. Compute what node needs to be selected, based on your criteria.</li> <li>2. Store the node in a variable.</li> <li>3. Override the tree's <code>onInit()</code> script.</li> <li>4. Use the <code>setSelection()</code> method to set the selection to this node and achieve dynamic selection of the node when the tree displays.</li> </ol>
<b>Auto save</b>	Yes	Yes	Yes	Yes	Yes	Yes	<p>This property is a selection of 3 choices if this is a Tree Root Node:</p> <ul style="list-style-type: none"> <li>• <b>All</b> – overrides auto-save setting for all nodes in the tree, the behaviour is like all children including the root have auto-save setting selected.</li> <li>• <b>Root only</b> – auto-save applies only to the root node and saves only</li> </ul>

							<p>the root node's model when selection leaves the root or on Auto-save of the User Action. Auto-save for children is set at the node's level for each node individually.</p> <ul style="list-style-type: none"> <li><b>None</b> – Auto-save is off for the root node, auto-save for children is set at the node's level for each node individually.</li> </ul> <p>For all other node types, this property is a boolean checkbox. If checked, any value changes made to the <b>Object</b> in the detail pane of the Navigation Tree are automatically saved when user navigates away (for example, to another node in the Navigation Tree).</p>
<b>Hide node if empty</b>	No	Yes	Yes	Yes	Yes	Yes	If checked, this node is hidden if it has no children nodes, or if none of them are visible. For example, this is useful when a Finder Node is desired to be hidden when it returns an empty result. When unchecked the property is applied to the Navigation Tree node when it contains children nodes.
<b>Hide expand icon if empty</b>	Yes	Yes	Yes	Yes	Yes	Yes	If checked, this node has no expand icon if it has no children nodes. This is usually desirable when a Finder Node returns an empty result.
<b>Sort by visual key</b>	No	No	Yes	Yes	No	Yes	If checked, the appearance of tree nodes created by this metadata node are sorted (alphabetically) by their visual key, if more than one tree node is generated. See <a href="#">Navigation Tree at runtime</a> to understand how the visual key of a node is computed.
<b>Show children count</b>	Yes	Yes	Yes	Yes	Yes	Yes	If checked, the node's display label is appended with a number in brackets, which is the number of children nodes that is under this node.
<b>Display in parent tabs</b>	No	Yes	Yes	Yes	Yes	Yes	Only visible when Root Node of the Navigation Tree extends from <code>com.conceptwave.system.NavigationTreeWithTabs</code> . If checked, the node appears to be assimilated into the parent; the detail <b>Form</b> of the node <b>Object</b> is displayed as a Tab Frame at parent node's Form (which becomes a Tabset), while the node is no longer visible in the Tree Form Element of the Navigation Tree.
<b>Display tab always</b>	Yes	Yes	Yes	Yes	Yes	Yes	<p>Only visible when Root Node of the Navigation Tree extends from <code>com.conceptwave.system.NavigationTreeWithTabs</code>. If checked, all children nodes with <b>Display in parent tabs</b> enabled must be displayed in tabs, even if there is only one such child node.</p> <p>Note that if unchecked, and there is only one visible <b>Display in parent tabs</b> child node, and this node does not have own details (for example, collections or top order), runtime will display the child node's detail from in the parent detail form, as-is without tabs.</p>
<b>Object</b>	Yes	Yes	Yes	Yes	Yes	Yes	<p>The metadata object that this node represents, which is a model object of:</p> <ul style="list-style-type: none"> <li><a href="#">Finder (of any type)</a> if this is a Finder Node, or</li> <li>fixed as Finder's Output Document of its parent node if this is a Finder Child Node, or</li> <li>either <a href="#">Document</a>, <a href="#">Data Structure</a>, <a href="#">Order</a>, or <a href="#">Finder (of any type)</a> if this is a Tree Node or Root Node.</li> <li>Item in a Data Structure if this is a Data Structure Item Node</li> <li>Item in an Order if this is an Order Item Node</li> </ul> <p>For Order Item Node or Data Structure Item Node, this field becomes read-only, and assigned based on the item type in <b>Order Item Path</b> or <b>Data Structure Path</b> respectively.</p>
<b>Detail UI</b>	Yes	Yes	Yes	Yes	Yes	Yes	The User Interface of <b>Object</b> to present the object's view at the detail pane of the Navigation Tree <b>Default</b> Form, when the node is clicked. This property may be set to <NONE>, which this node (which is a derived User Interface object in itself) is set to be the detail User Interface. See method <code>onNodeDetail()</code> for details, which returns the detail UI based on variable <code>modelUI</code> .
<b>Form</b>	Yes	Yes	Yes	Yes	Yes	Yes	The Form of the <b>Detail UI</b> to display at the detail pane of the Navigation Tree <b>Default</b> Form, when the node is clicked. The choices are restricted to

							the Forms of the selected <b>Detail UI</b> User Interface. If <b>Detail UI</b> is <NONE>, then the local Forms of this node are available to be chosen.
<b>Table Document</b>	Yes	No	No	No	No	No	<p>The Document that is the model metadata object for the Navigation Tree's Tree Form Element. This Document must be or extended from <code>com.conceptwave.system.TreeDocument</code>, which has TreeNode-defining Variables such as <code>label</code>, <code>image</code>, <code>expanded</code>, etc that controls the appearance of each node as shown in the Tree Form Element. See description following this table for details.</p> <p>When a Table Document is selected that is different from the base Navigation Tree's Table Document, the Variable <code>NavTree</code> is created in the <b>Variables</b> tab to override the Data Type of the existing <code>NavTree</code> Variable to this specified Table Document.</p>
<b>Search Conversion Map</b>	No	No	No	No	Yes	No	<p>The Conversion Map that translates the parent node's <b>Object</b> to the Input Document of the Finder. This map effectively defines the value of Finder search input based on the object in parent node, and thus obtains the result Documents as children of the Finder Node. This may be &lt;NONE&gt;, <b>Default</b>, or a Conversion Map can be assigned. If assigned, the Conversion Map is applied on the node's initialization, as well as whenever an action event (such as Add, Save, Update, Delete) occurs to the <b>Object</b> of the parent node in the Navigation Tree. <b>Default</b> chooses the Conversion Map, if there are multiple of them for the same source/target pair defined in project, that has boolean <b>Default</b> property enabled.</p>
<b>Table Conversion Map</b>	Yes	Yes	Yes	Yes	Yes	Yes	<p>The Conversion Map that translates the <b>Object</b> to the <b>Table Document</b> of the Root Node. This may be &lt;NONE&gt;, <b>Default</b>, or a Conversion Map be assigned. If assigned, the Conversion Map is applied on node's initialization, as well as whenever an action event (such as Add, Save, Update, Delete) occurs to the <b>Object</b> of the Tree Node in the Navigation Tree. <b>Default</b> chooses the Conversion Map, if there are multiple of them for the same source or target pair defined in project, that has boolean <b>Default</b> property enabled.</p> <p><b>Note:</b> When displaying a tree node, the system metadata works as follows:</p> <ol style="list-style-type: none"> <li>1. Applies the table conversion map to convert the model document to the tree document.</li> <li>2. Sets the tree document's image to the value returned from the <code>onNodeIcon()</code> method.</li> </ol> <p>If the image was set through the table conversion map, its value is lost as it is overridden by the <code>onNode()</code> method. If <code>onNodeIcon()</code> is not overridden, it is still invoked and returns the default tree icons. It is recommended that you move your logic for computing the image value from your table conversion map to your <code>onNodeIcon()</code> script of your corresponding tree nodes.</p>
<b>Order Item Path</b>	No	No	Yes	No	No	No	The path of the Order Item that is to be represented by the node. The Order Item must be in an Order that is modelled as the <b>Object</b> of any ancestor node to this node.
<b>Data Structure Path</b>	No	No	No	Yes	No	No	The path of the Data Structure Item that is to be represented by the node. The Data Structure Item must be in a Data Structure that is modelled as the <b>Object</b> of any ancestor node to this node.
<b>Timer duration</b>	Yes	No	No	No	No	No	If a timer interval is specified, the timer calls the <code>on Timer</code> method each time an interval elapses.
<b>Default Folder Image</b>	Yes	No	No	No	No	No	The default image icon of nodes that have children nodes.
<b>Default Leaf Image</b>	Yes	No	No	No	No	No	The default image icon of nodes that do not have children nodes.

The significant properties in this tab are the **Object**, **Detail UI** and **Form** properties, which bind a metadata object to the node and specifies its User Interface and Form when the node is clicked at runtime.

Another important property is the **Table Document**, applied only to the Root Node. This property casts the metadata object type of two important Variables in the Navigation Tree, *tableDoc* and *navTree*, which controls much of the runtime behaviour of the Navigation Tree. You can simply choose the system Document *com.conceptwave.system.TreeDocument* here, or extend a Document from it and assign this Document as the Table Document. In the latter case, you can add Variables (or Methods) to the extended Document, and display these additional Variables as columns of the Tree Form Element via Tree Table. This is a great feature when additional description or properties are required to be displayed alongside nodes within the Tree Form Element. See [Navigation Tree at runtime](#) for runtime samples.

#### Design Note: why is Table Document necessary?

Unlike other metadata objects, all node types in the Navigation Tree are enforced to be derived from system metadata tree node objects only, and can never be derived from user-defined objects (for example, create *TreeNodeA*, and then create *TreeNodeA1* by extending from *TreeNodeA*). To overcome the inability to declare (User Interface) Variables across all nodes in a Navigation Tree (by creating a base node, create Variables, then have all nodes extend from the base node) and be presented in the Tree Form Element, the Table Document is the means to model extensibility that is assigned to the root node of the Navigation Tree.

Lastly, the **Exclude** flag may be enabled on a container node or a Finder Node to flatten any undesired hierarchy in the Tree, and to have different list of metadata objects (for example, different Documents) appear as siblings of each other.

Below are the general properties of Recursive Node.

<b>General</b>	<b>Methods</b>
<b>Name:</b>	customerRecursiveNode
<b>Label:</b>	customerRecursiveNode <input style="width: 20px; height: 20px;" type="button" value="..."/>
<b>Description:</b>	<input type="text"/> <input style="width: 20px; height: 20px;" type="button" value="..."/>
<b>Recursive item:</b>	sampleNS.customerTree.customerFinderNode (Contacts) <input style="width: 20px; height: 20px;" type="button" value="..."/>
<input checked="" type="checkbox"/> <b>Exclude</b>	

Field	Description
<b>Name</b>	Name of the metadata object (used in scripts; must conform to JavaScript naming conventions). Read-only after creation. To change it, use the <b>Rename</b> command in the popup menu by right-clicking the metadata object.
<b>Label</b>	Node's label. This is not used at runtime; the label is replaced by the referenced <b>Recursive item</b> 's label.
<b>Description</b>	Description of the metadata object for documentation.
<b>Recursive item</b>	Associate any node of any Navigation Tree to become a sub-tree of this Navigation Tree structure, with the chosen node as the root of the sub-tree and its children and their successors, if any, spanned as the sub-tree. To have recursive behaviour, however, the chosen node must be an ancestor of the Recursive Node.
<b>Exclude</b>	If checked, the root of the Recursive sub-tree would not be displayed. However, children nodes of this node, if any, will be displayed under this node's parent. This effectively "flattens" the tree at runtime.

Often, a Recursive Node is instructed to recur on an ancestor Finder Node, to display a dynamic, hierarchical data structure on the Navigation Tree. By simply setting the correct search criteria of the Finder (on Finder Node's initialization), the recursiveness naturally ends at the data structure's leaf nodes.

Another generic way to stop the recursiveness is by using the *visiblePerm* permission.

## Navigation Tree Variables

The **Variables** tab is used to manage Variables of a node in the Navigation Tree. The **Variables list** contains one entry for each variable.

All node types of the Navigation Tree extend from system base metadata objects which have many variables. These base Variables are shown in blue color. Variables added to this Navigation Tree node are in black color.

The screenshot shows the Velocity Studio interface with the 'Variables' tab selected. The main pane displays a table of variables:

Name	Type	Methods	Vis	Opt	Edit
confirmObject	Object	<input type="checkbox"/>			
model	Model	<input type="checkbox"/>			
label	String	<input type="checkbox"/>			
image	String	<input type="checkbox"/>			
expanded	Boolean	<input type="checkbox"/>			
parentNode	Tree Node	<input type="checkbox"/>			
childNodes	Tree Node	<input type="checkbox"/>			
loaded	Boolean	<input type="checkbox"/>			
dirty	Boolean	<input type="checkbox"/>			
tableDoc	Document	<input type="checkbox"/>			
modelUI	User Interface	<input type="checkbox"/>			
tabUI	User Interface	<input type="checkbox"/>			
enableEvents	Boolean	<input checked="" type="checkbox"/>			
errorList	Object	<input type="checkbox"/>			
navTree	Document	<input type="checkbox"/>			
detailNode	Tree Node	<input type="checkbox"/>			

Below the table, a detailed view for the 'confirmObject' variable is shown:

General	Methods				
Name: confirmObject	<input type="checkbox"/> Array <input type="checkbox"/> Constant <input type="checkbox"/> Audit <input type="checkbox"/> Validate				
Description:					
Data type:	com.conceptwave.system.Object				
Type error code:					
Mandatory error code:					
Element properties:	<Inherited>				
<table border="1"><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td></td><td></td></tr></tbody></table>		Name	Value		
Name	Value				

These buttons beside the list follow the [Manage Array Elements](#) presentation convention in Velocity Studio. See this section for details about the aforementioned and additional buttons.

The Navigation Tree Variable details appear in the detail pane when a Variable entry is highlighted in **Variables list**. This Variables tab operates in the same manner as the [Document's Variables tab](#), so please consult that page for any details.

The following table describes the base Variables for all of Tree Nodes, Finder Nodes, Finder Child Nodes and Root Nodes in Navigation Tree. Note that Recursive Node has no **Variables** tab.

Variable	Type	Description
confirmObject	Object	The generic object for confirmation.
model	Model	The model of which this (User Interface) node is representing; it has the metadata object type assigned by <b>Object</b> in general tab.

<b>childNodes</b>	<i>Tree Node</i>	Children nodes of this node, represented as an array. Access its element by numeric index (for example, <code>childNodes[0]</code> is the first child node). This Variable can be null when the node has not been loaded; node may be forced to be loaded by <b>Auto-expand</b> or <b>Hide node if empty</b> flags in <b>General</b> tab. Also, method <b>loadChildren</b> may be used to load/reload child nodes explicitly.
<b>dirty</b>	<i>Boolean</i>	Flag that marks the node to be refreshed in browser. It is set to true when the node's label, icon or any other property is changed. It is set back to false after the data was sent to browser. Thus, when refreshing the page after user action, only dirty nodes and their children get reloaded and refreshed.
<b>enableEvents</b>	<i>Boolean</i>	Flag whether the system method <code>onModelEvent</code> should be triggered to handle action events at the node's model, such as user updating it in detail pane of the Navigation Tree.
<b>errorList</b>	<i>Object</i>	An array object that stores the validation errors.
<b>expanded</b>	<i>Boolean</i>	Determines whether the node is at expanded state or collapsed state in the Tree Form Element.
<b>image</b>	<i>String</i>	Stores the resource path of the image icon of the tree node.
<b>label</b>	<i>String</i>	The label Variable that determines the label of the tree node.
<b>loaded</b>	<i>boolean</i>	Determines whether this node has been loaded in the Tree Form Element or not.
<b>modelUI</b>	<i>User Interface</i>	User Interface object that connects the model to the Detail Form.
<b>parentNode</b>	<i>Tree Node</i>	The parent node of this node.
<b>tabUI</b>	<i>User Interface</i>	Tab User Interface object that contains label, icon, detail UI, owner (which is reference to parent node), etc. This object is reused when the node needs to be displayed in the tab again.
<b>tableDoc</b>	<i>Tree Document</i>	The Table Document Variable, which is a Document with various tree-related variables in the Document that controls the node's behaviour in the Navigation Tree. The Table Document is casted to the metadata object type of <b>Table Document</b> in <b>General</b> tab at runtime; this mechanism enables the node to store additional model Variables for the Tree Form Element of the Navigation Tree.

The following table describes *additional* base Variables for Root Nodes only.

Variable	Type	Description
<b>actionNode</b>	<i>Tree Node</i>	The node of the current hover/double click action.
<b>cwShowHover</b>	<i>Boolean</i>	Instructs the Web browser to show hover data by evaluating it at the <b>Hover Method</b> property of Root Node's Tree Form Element. If set to true, the hover form will be displayed. For an example, see the implementation in <code>com.conceptwave.system.OrderUserInterface</code> .
<b>detail</b>	<i>User Interface</i>	The User Interface for the Detail Form.
<b>detailForm</b>	<i>String</i>	The string Variable that specifies the Detail Form.
<b>detailMenu</b>	<i>User Interface</i>	The User Interface for the Menu Form of the Detail Form.
<b>detailNode</b>	<i>Tree Node</i>	The Tree Node that is shown through the Detail Form. This Tree Node's <b>Detail UI</b> in General tab, by default, is assigned to the <i>detail</i> Variable.
<b>navTree</b>	<i>Document</i>	Array of tree nodes that is binded to the Tree Form Element to serve as its model. The data type of this array Variable is overridden to the <b>Table Document</b> in General Tab if this property is different from <code>com.conceptwave.system.TreeDocument</code> . You can access elements in the array by node ID, such as <code>this.navTree["myTestTree/node1/customer"]</code> ; however, numeric index (for example, <code>navTree[0]</code> ) does not work. <b>The true data type of this Variable is not of tree node's Table Document as specified in Velocity Studio, but of tree node itself.</b> You can simply access the tree node's Table Document by <code>navTree[&lt;pathname&gt;].tableDoc</code> . The type parameter of this Variable is exploited as “interface◆? to define a list of variables to bind tree columns under the Tree Form Element.
<b>treeSel</b>	<i>Tree Node</i>	The array of Tree Nodes that are selected in the Tree Form Element by the user.

The following table are further *additional* base Variables for Root Nodes that are extended from `com.conceptwave.system.NavigationTreeWithTabs`.

Variable	Type	Description
<b>currentForm</b>	<i>String</i>	The string variable that specifies the Form to display in the detail Form Frame. By default, the Form options are <i>Single</i> or <i>Tabs</i> Forms.

<b>tabs</b>	<i>User Interface</i>	Array of Tab User Interface that represents the Tabs in the Tabset, binded via the Iterator Form Element, to control the Tab Frames of the detail pane. This is for children nodes with <b>Display in parent tabs</b> in <b>General</b> tab to be displayed in tabs.
<b>selectedTab</b>	<i>String</i>	The string variable that has the numeric index (zero-indexed) of the selected Tab within the TabSet.

Some of the more significant Variables are:

- *model, modelUI* - the model and its User Interface that represent the node's **Object** in **General** Tab, and displayed via the detail pane when the node is clicked.
- *tableDoc* - the Table Document, to model the node's representation in the Tree Form Element.
- *navTree* - the array of tree nodes, at the tree-level, that is binded to model the node's representation in the Tree Form Element. It has some read-only properties similar to *result variable in Finders* that could be useful, such as *.length* as count of all loaded nodes, *.selected* as array of selected nodes, and *selectedCol* as the action column.
- *detailNode, detail* - the Tree Node, and its **Detail UI** in **General** Tab, that is currently shown in the detail pane.

### Tree Document

The Variable *tableDoc* is a Tree Document which has some Document Variables that overlap some of the base Variables at the node level. You may use either of them to get/set the node's runtime properties in the Navigation Tree. For example, the expanded/collapsed state of the node in the tree can be determined by either *this.tableDoc.expanded* or *this.expanded*. Setting one of the variables automatically sets the other.

The base Document Variables in the Table Document *com.conceptwave.system.TreeDocument* are listed below.

Variable	Type	Description
<b>id</b>	<i>String</i>	Identifier of the tree node.
<b>label</b>	<i>String</i>	The tree node label.
<b>parentId</b>	<i>String</i>	Identifier of the tree node's parent node.
<b>image</b>	<i>String</i>	Image pathname for the tree node's icon.
<b>isFolder</b>	<i>Boolean</i>	Flag whether the node is a leaf or not in the Navigation Tree.
<b>expanded</b>	<i>Boolean</i>	Determines whether the node is at expanded state or collapsed state in the Tree Form Element.
<b>excluded</b>	<i>Boolean</i>	If true, the node would not be displayed. However, children nodes of this node, if any, will be displayed under this node's parent.
<b>visible</b>	<i>Boolean</i>	Determines whether the node is visible.
<b>sortOrder</b>	<i>Integer</i>	Keeps the current sort order of the visible node under visible parent in the browser. The sort order is reset every time when data source request is performed for the tree from browser. Thus, this Variable maintains nodes are sorted after refresh.
<b>singleCell</b>	<i>Boolean</i>	May be used in table trees, to indicate that the Tree Node label (first column cell) should spread across all columns.

## Navigation Tree Node Items

All node types of Navigation Tree, except Recursive Nodes, may define children nodes in the metadata. (Recursive Nodes may have children nodes at runtime, but they are defined at the metadata of the associated Recursive Tree Node.) The **Navigation Tree Items** tab in Root Node, and the **Tree Node Items** tab in Tree Node, Finder Node and Finder Child Node contains the list of children nodes. The two aforementioned tabs are equivalent; only the tab label is different based on the node type.

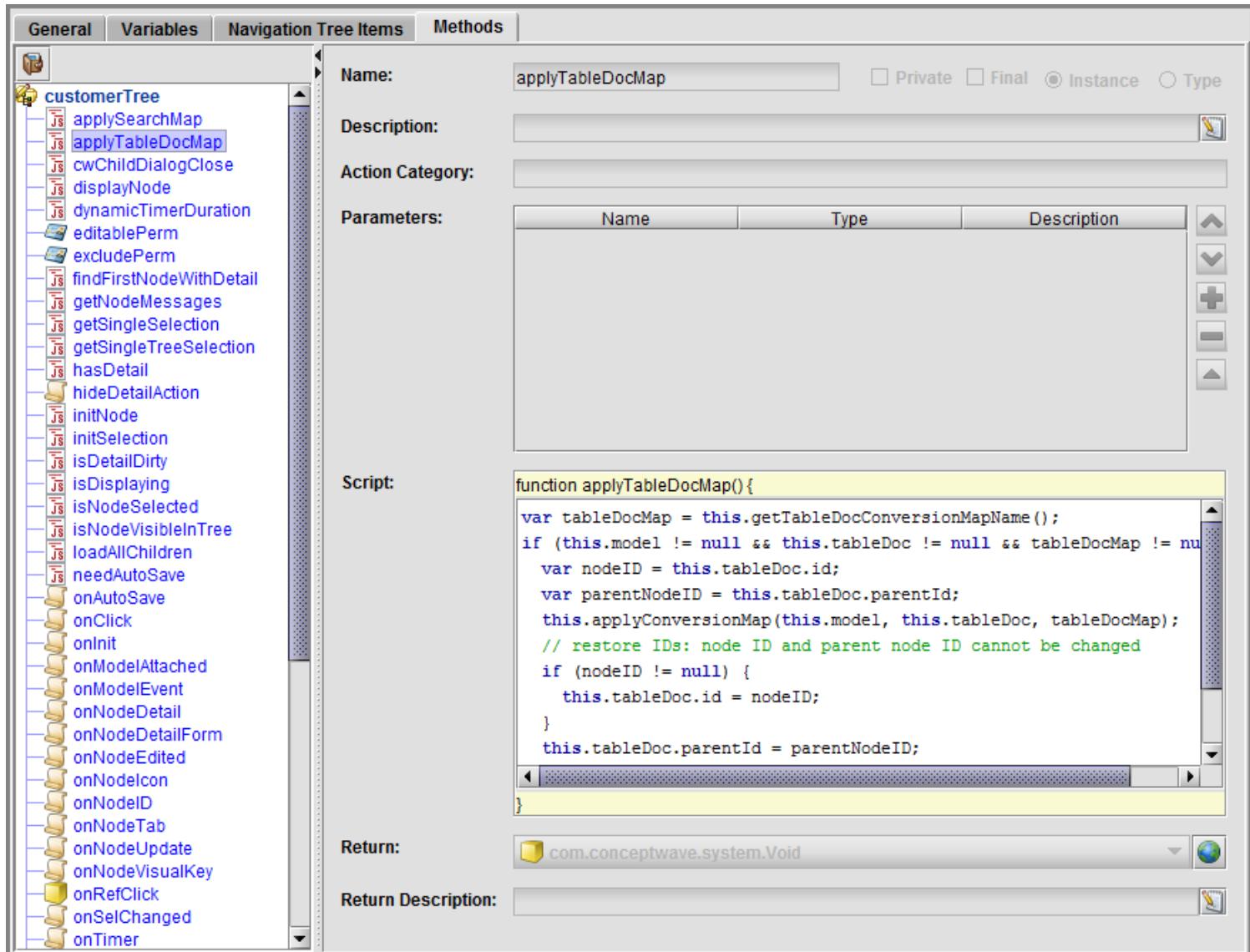
The children nodes appear according to their order in the list, from top to bottom, in both Velocity Studio's Navigation Pane, as well as Tree Form Element at runtime. The arrow buttons located at the side of the list are used to move the selected element in the list to the desired order.

A screenshot of the Velocity Studio interface showing the 'Navigation Tree Items' tab selected. The tab bar includes 'General', 'Variables', 'Navigation Tree Items' (selected), and 'Methods'. Below the tabs is a list of three items: 'sampleNS.customerTree.customerTreeNode', 'sampleNS.customerTree.customerFinderNode' (which is highlighted with a blue selection bar), and 'sampleNS.customerTree.customerRecursiveNode'. To the right of the list are two small arrow buttons for reordering. At the bottom left is a 'Sort' button.

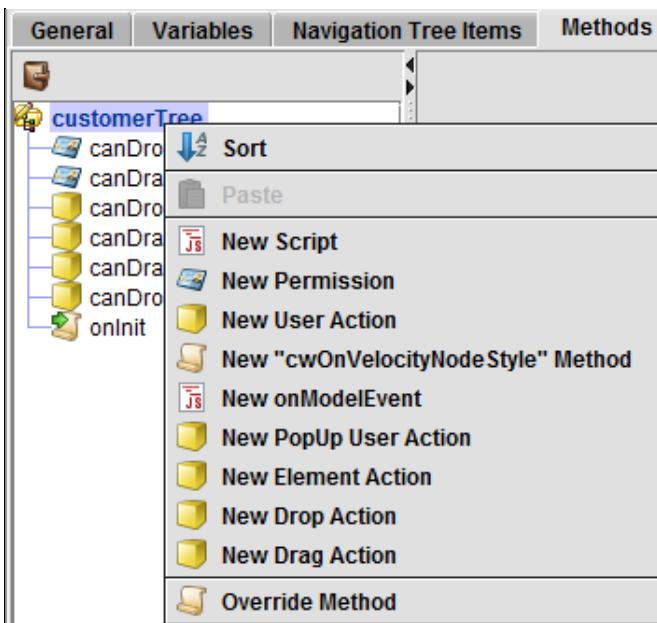
## Navigation Tree Methods

You can use the **Methods** tab to add scripts to a node in a navigation tree. There are different types of methods that can be defined for the navigation tree; they are all consolidated into this tab.

**Note:** For navigation tree variable-level scripts, they can be managed under the **Variables** tab by selecting the variable that you want in the variables list, and then selecting the **Methods** tab in the variable detail pane to edit the variable's methods. These variable-level methods also appear and are editable on the navigation tree's **Method** tab. These methods can be differentiated from other methods by their method names, which are appended by `$.<variable name>`.



The Methods tab is a [script management screen](#) in Velocity Studio, which is composed of the **Method list pane** on the left that lists all scripts of the Navigation Tree, and the **Details pane** that displays the details of a script when selected from the Method list pane. To create a new method, or to override an existing method of its base navigation tree, right-click the navigation tree in the Method list pane. A pop-up menu appears.



If you have a long list of methods in the left pane, you can press the first letter of the method you are looking for to jump to the list of methods beginning with that letter. For example, pressing the I key highlights the first instance of a method that begins with the letter I. The method displays in the right pane.

The following table lists the wide-ranging type of methods that you can either add or override at the Navigation Tree level.

#### New user-defined methods

Method Type	Description
Script	Generic user-defined JavaScript function that can be defined in a Navigation Tree and explicitly invoked by other scripts; the function is not triggered by other built-in means in the Navigation Tree.
Permission	Permission Method that can be associated to determine permission for a certain property of the Navigation Tree. (for example, Element properties). See section <a href="#">Permissions</a> for details.
User Action	Navigation Tree <a href="#">User Action</a> method that may be assigned to Form Elements to respond to a user action such as a mouse click.

#### New system pre-defined methods

Method Type	Script Name	Parameters	Return Type	Description
Tree Node Style	<i>cwOnVelocityNodeStyle</i>	<i>tableDoc</i>	<i>String</i>	Declare this method to return custom row style, in string, for the tree node.

#### System-defined methods that can be overridden for all types of tree nodes, except Recursive Node

Method Type	Script Name	Parameters	Return Type	Description
Apply Search Map	<i>applySearchMap</i>	None	None	This function applies the assigned <b>Search Conversion Map</b> in General tab to populate the search Input Document value. Only applicable for Finder Child node.
Apply Table Document Map	<i>applyTableDocMap</i>	None	None	This function applies the assigned <b>Table Conversion Map</b> in General tab to translate the Tree Node model to the Table Document.
Get Drag Data	<i>cwGetDragData</i>	<i>object</i>	<i>object</i>	When called under a tree node, this method gathers the list of models for each tree node being dragged and return this list. If the object passed in is a collection of objects in a tree, then the children's models under this collection are returned. This method is called on the source controller before <i>cwGetDropData</i> is called.
Get Drag	<i>cwGetDragObjectType</i>	None	<i>String</i>	This method is called for every tree node that is generated as part of

<b>Object Type</b>				<p>the tree data. This type represents the object that is about to be dragged. This type will be checked against the list of types returned from cwGetDragTypes. When this method is called from a tree node, the tree does the following:</p> <ul style="list-style-type: none"> <li>• If the tree node represents a finder, cwGetDragObjectType is called on the node's finder model.</li> <li>• If the tree node represents a datastructure, the following happens: <ul style="list-style-type: none"> <li>◦ If the node is a top level data structure, the model's cwGetDragObjectType is called.</li> <li>◦ If the node is a collection, its element metadata type is returned.</li> </ul> </li> <li>• If the node is anything else, cwGetDragObjectType on the model is called (if it exists).</li> </ul>
<b>Get Drag Types</b>	<code>cwGetDragTypes</code>	None	<code>String</code>	<p>This method returns a comma-separated string of metadata type names. The default implementation for Navigation Tree User Interface is as follows:</p> <ul style="list-style-type: none"> <li>- Gather a list of metadata type names that correspond to the tree node's model metadata type, and any objects in the metadata which extend this model's type.</li> <li>- Called during the fetch of the data. When an object's type is not included in this list, it is marked as <code>canDrag=false</code>.</li> </ul>
<b>Get Drop Data</b>	<code>cwGetDropData</code>	<code>object</code>	<code>object</code>	The default implementation of this method returns the object passed in. This method is called on the target controller near the beginning of <code>cwOnDrop</code> after <code>cwGetDropData</code> has been called.
<b>Get Drop Types</b>	<code>cwGetDropTypes</code>	None	<code>String</code>	This method returns an array of metadata type names. This method will be called as part of the data fetch. The list returned from this method is sent to the browser for easy verification of object acceptance when a user attempts to drop an object.
<b>Exclude Permission</b>	<code>excludePerm</code>	<code>\$psCondition</code>	<code>Boolean</code>	Exclude Permission that specifies whether the tree node is to be displayed (false) or be hidden (true) such that the node's children will be displayed directly under the node's parent.
<b>Expand All</b>	<code>expandAll</code>	None	None	Expands the node and its children nodes.
<b>Get Node Messages</b>	<code>getNodeMessages</code>	<code>messagesArray, includeChildren</code>	<code>Object - (Array of messages)</code>	Retrieves and returns an array of Validation Error messages for the node. If boolean parameter <code>includeChildren</code> is true, recursively retrieves node messages from descendant nodes.
<b>Initialization</b>	<code>initNode</code>	None	None	Implementation function that initializes the tree node.
<b>Auto-Save</b>	<code>onAutoSave</code>	None	<code>Boolean</code>	System method that is triggered on auto-save.
<b>On Click</b>	<code>onClick</code>	None	None	The generic system method that is triggered when there is a mouse-click.
<b>Initialization</b>	<code>onInit</code>	None	None	The system method that is triggered on node's initialization. Calls <code>initNode</code> .
<b>On Model Event</b>	<code>onModelEvent</code>	<code>event</code>	None	The system method that is triggered when an action event occurs at the node's model, such as user updating the model in the Form. This event handling is enabled at node's initialization by registering to the model's notification service.
<b>Detail UI</b>	<code>onNodeDetail</code>	None	<code>User Interface</code>	The system method that is triggered when the node is selected in the tree, to return the detail User Interface of the model object.
<b>Detail Form</b>	<code>onNodeDetailForm</code>	None	<code>String</code>	The system method that is triggered when the node is selected in the tree, to return the name of the detail Form of the model object to display to the user.
<b>On Node Edited</b>	<code>onNodeEdited</code>	<code>event</code>	None	The system method that is triggered when the row that represents the node in Tree Form Element, the model of which is the Table Document presented as Table Tree, is editable and updated by user. Typically, logic may be inserted to update the node's model based on the updated Table Document.
<b>ID</b>	<code>onNodeID</code>	None	<code>String</code>	The system method that calculates the node identifier string.

<b>Icon</b>	<code>onNodeIcon</code>	None	<code>String</code>	The system method that calculates the resource path of the tree node icon image. It can have overlay images, similar to those in Navigation Pane of Velocity Studio itself such as a "pencil" overlay for dirty marking, or "green arrow" overlay for overridden objects. To overlay an image, simply return a list of images separated by comma in this method. As an example, <code>return this.cw\$super_onNodeIcon() + "/cwf/add.gif(top-right)";</code> returns all icon images with the "add" icon overlay embedded at top-right.
<b>On Node Tab</b>	<code>onNodeTab</code>	None	<code>User Interface</code>	The system method that is invoked only for children nodes with <b>Display in parent tabs</b> enabled in <b>General</b> tab, to get the tab UI for the node. By default, it creates Tab User Interface <code>tabUI</code> if not previously created, and otherwise re-uses it.
<b>Update</b>	<code>onNodeUpdate</code>	<code>node, action, param1, param2</code>	None	The system method that is invoked to update the node as well as the overall Navigation Tree based on an action event; invokes by <code>onModelEvent</code> method.
<b>Visual Key</b>	<code>onNodeVisualKey</code>	None	<code>String</code>	The system method that computes the node's Visual Key (that is, label of the node).
<b>Validate</b>	<code>onValidate</code>	None	None	<p>The system method that is triggered when validation happens:</p> <ul style="list-style-type: none"> <li>• On User Action method invocation, if its <b>validate</b> checkbox is selected</li> <li>• On selection changed in the navigation tree</li> <li>• On tabs selection changed.</li> </ul> <p>By default, validation executes all validation methods for all Variables in the User Interface as well as validation methods on the model (if exists). Note that, however, validation at <code>tableDoc</code> is not executed by default.</p>
<b>Refresh Node</b>	<code>refreshNode</code>	<code>reloadChildren, keepExpanded</code>	None	Helper method that forces to refresh the node and its children.
<b>Refresh Privileges</b>	<code>refreshPrivileges</code>	None	None	Helper method that refreshes visible and exclude privileges for the node and its children.
<b>Save Node</b>	<code>saveNode</code>	<code>startNode, validate, recursively</code>	<code>Boolean</code>	Saves the model of <code>startNode</code> and all its successors if <code>recursively</code> is true. Performs validation if <code>validate</code> is true. If validation fails, the save is aborted when returns false.
<b>Set Validation Errors</b>	<code>setValidationErrors</code>	<code>validationErrorList</code>	None	Populates validation error list Variable <code>errorList</code> from parameter <code>validationErrorList</code> .
<b>Update Node</b>	<code>updateNode</code>	<code>updateTableDocument, reloadChildren, keepExpanded</code>	None	Helper method to update the node in the Navigation Tree such as to update Table Document, reload children nodes, and keeping expanded states of the node and its children at refresh, if the corresponding boolean parameter is enabled.
<b>Update Table Document</b>	<code>updateTableDoc</code>	None	None	The helper method that implements the Table Document update. Invokes <code>applyTableDocMap()</code> .
<b>Visible Permission</b>	<code>visiblePerm</code>	<code>\$psCondition</code>	<code>Boolean</code>	Visible Permission that determines whether the node is visible in the Navigation Tree or not.

Some of the more significant methods are as follows:

- `initNode` - the initialization method for the tree node.
- `onModelEvent` - event handler for the node when its model has an action event such as update.
- `onNodeEdited` - the function that handles a user updating a node at the Tree Form element itself. For example, if node label is changed, override this function to update the node's model, such as the model object's label, based on Table Document's label.

#### Design Note:

Unlike many automatic mechanisms in the Navigation Tree metadata object, `onNodeEdited` is an explicit handler that must be custom-developed because the specified **Table Conversion Map** in the General tab converts from the node's model to a table document, but not inversely. You must, in this method, specify how to update the model based on the changed table document.

Additional system-defined methods that can be overridden for the Root Node only

Method Type	Script Name	Parameters	Return Type	Description
Display Node	<code>displayNode</code>	<code>node</code>	<code>Tree Node</code>	The method to select and display the node's detail. The difference between <code>setSelection()</code> and <code>displayNode()</code> is that the latter one works with <b>display in parent tabs</b> nodes as well. It finds the appropriate (closest) visible Tree Node, sets the Tree selection and, selects the necessary Tab ID.
Get Single Selection	<code>getSingleSelection</code>	<code>None</code>	<code>Tree Node</code>	The method to return a single selection of tree node, in the event where multiple nodes are selected in the Navigation Tree.
Hide Detail Action	<code>hideDetailAction</code>	<code>None</code>	<code>None</code>	The method that hides the detail pane in the Navigation Tree.
Is Displaying	<code>isDisplaying</code>	<code>node</code>	<code>Boolean</code>	The method that determines if the node is being displayed or not.
Is Node Selected	<code>isNodeSelected</code>	<code>node</code>	<code>Boolean</code>	The method that determines if the node is being selected or not.
On Changed Selection	<code>onSelChanged</code>	<code>None</code>	<code>None</code>	The method that is triggered when node selection at the Tree Form Element is changed.
Set Node as Selected	<code>setNodeSelected</code>	<code>node, state</code>	<code>None</code>	The method that includes a particular <code>node</code> as a selected node at the Tree Form Element.
Set Node Selection	<code>setSelection</code>	<code>nodes, fireEvent</code>	<code>None</code>	The method that sets an array of <code>nodes</code> as the list of selected nodes at the Tree Form Element, changing Root Node Variable <code>treeSel</code> . Method <code>onSelChanged</code> is invoked if parameter <code>fireEvent</code> is true or missing.
Show Detail Action	<code>showDetailAction</code>	<code>None</code>	<code>None</code>	This method is triggered when the detail pane is to be displayed based on the node selection.

Additional system-defined methods that can be overridden for Root Node with tabs only

Method Type	Script Name	Parameters	Return Type	Description
Add Tab	<code>addTab</code>	<code>tabsArray, node</code>	<code>None</code>	Adds the <code>node</code> as a tab at the end of the <code>tabsArray</code> .
<b>currentForm's Initialization</b>	<code>cwLeafInitAction\$currentForm</code>	<code>None</code>	<code>String</code>	The initialization method of Variable <code>currentForm</code> that specifies the Form to display in the detail Form Frame, which defaults to Form "Single".
Changed Tab	<code>onTabChanged</code>	<code>None</code>	<code>None</code>	The method that is triggered when tab selection is changed among Tabset in the detail pane.
On Timer	<code>onTimer</code>	<code>None</code>	<code>None</code>	Generic timer method that is periodically invoked.
Select Tab	<code>selectTab</code>	<code>index</code>	<code>None</code>	The method that sets a tab selection based on <code>index</code> .

Recursive Node has only one system method: `cwOnVelocityNodeStyle`.

For method types that can be created, you can create multiple methods of the same method type (for example, Permission 1, Permission 2...), each identifiable by a unique method name. However, for methods that are to be overridden, you can only define one method for each base method.

For scripts that are already overridden, they appear in the Method list pane and are no longer available by right-clicking from the context popup menu.

## Use a Script Method for Calculating a Tree Column

When creating any script method that will be used in the variable of a calculated column of a tree, use the following two parameters in your script:

- `record` is the tree document, which is of type `com.conceptwave.system.TreeDocument`
- `node` is the tree node, which is of type `com.conceptwave.systemTreeNode`

The following is an example of defining these parameters in a script called `hasAttachmentImage`:

General Variables Navigation Tree Items Methods

**UserInterface**

- provision
- provision\_confirm
- provision\_menu
- cwAutosaveConfirmation
- hasAttachmentImage**

**Name:** hasAttachmentImage  Private  Final  Instance  Type

**Description:**

**Action Category:**

**Parameters:**

Name	Type	Description
record	com.conceptwave.system:TreeDocument	
node	com.conceptwave.system:TreeNode	

**Script:**

```
function hasAttachmentImage(record,node) {
    if (node != null && node.model != null && node.model.hasAttachment)
        return "/cwfv/attach.png";
    return null;
}
```

**Return:** com.conceptwave.system.String

For the navigation tree's column, hasAttachmentImage, its **Variable** field uses the hasAttachmentImage script that was previously defined to calculate the tree's column.

AVM Metadata

- Imd (Training MD)
- Business Processes
- Data Dictionaries
- External Services
- Finders
- Orders
- Orders
  - shoppingCart (Shopping Cart)
    - UserInterface (Shopping Cart)
      - Default
      - hLayout
      - hLayout2
      - tree
      - label
      - hasAttachmentImage
- Presentation

The screenshot shows the AVM Metadata interface. On the left, there's a tree view of metadata objects. In the center, there's a detailed view of a 'Default' node under 'UserInterface (Shopping Cart)'. On the right, there's a table for configuring the 'hasAttachmentImage' column. The 'Variable' row is circled in red, showing the value 'hasAttachmentImage'.

Name	Value
Name	hasAttachmentImage
Label	
Click Method	attachAction
Image URL	
Variable	hasAttachmentImage
Action Auditor	
Can Focus	<input type="checkbox"/>
Can Sort	<input checked="" type="checkbox"/>

## Programmatically Display the Node During Tree Initialization

If you have a display detail form, you can display it automatically after the tree view appears by following these steps:

1. The document ID that needs to be selected must be known.
2. During initialization, find the node by the document ID.
3. Select the node and trace back to expand the entire path from leaf to root.
4. Trigger the onSelChanged method at the root to show the detail form.

The navigation tree's `findNodeByID` method takes the `nodeId` as a parameter. This method must be implemented at all levels of the tree node for tracing the node from root to leaf as follows:

```
// Helper method: finds child tree node by node ID
if (this.tableDoc.id == nodeId) {
    return this;
}

this.loadChildren();
var nodeFound = null;
```

```

if (this.childNodes != null) {
    var len = this.childNodes.length;
    for(var i = 0; i < len; i++) {
        nodeFound = this.childNodes[i].findNodeByID(nodeID);
        if (nodeFound != null) {
            return nodeFound;
        }
    }
}

return null;

```

The trace back script on the navigation tree's onInit method is as follows:

```

this.globalUI = new cwt_pc.globalUI(null, this);
this.initNode();
this.loadChildren();
this.ads = true;

var node = this.findNodeByID("<Enter node ID here>");
var root = this.getRoot();

if (!root.isDetailView) {
    //display the search form at the very beginning
    var nodes = new Array();
    nodes[0] = node;
    var root = node.getRoot();
    root.treeSel = nodes;

    // expand paths
    if (nodes != null) {
        for (var i = 0; i < nodes.length; i++) {
            var parentN = nodes[i].parentNode;
            while (parentN != null) {
                parentN.expanded = true;
                parentN = parentN.parentNode;
            }
        }
    }
    root.onSelChanged();
}

```

## Dynamically Change Icons for Nodes in a Navigation Tree

Support for dynamic icons in the first column of a navigation tree is available. The icons are in a separate column is due to being able to invoke an action when you click an icon.

When you create a new navigation tree, it is created with one table column. You must ensure that is is not replace with an icon field or something similar.

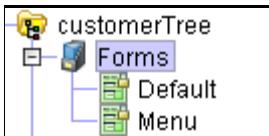
To have both the node text and the icon be dynamic, you need to override the following methods for each tree node:

- The onNodeVisualKey method for text
- The onNodeIcon method for icons

**Note:** Placing an Image element as the first column does not show any image due to SmartClient's restrictions. The workaround is to use a label with a script returning the HTML image tag with paths dynamically and to set the **Escape HTML** field to false.

## Navigation Tree Forms

Except for Recursive node, each node type in Navigation Tree is inherited as User Interface metadata object which may have Forms. They can be found under the **Forms** folder in Velocity Studio's **Navigation Pane**.



Below is an overview of all Forms provided by default among the various node types in Navigation Tree. See [Navigation Trees at Runtime](#) on how some of these Forms may be modified to customize some common runtime features of the Navigation Tree. Also, see the [Tree Form Element](#) to examine how this Form Element may be customized to serve your Navigation Tree presentation needs.

Node Type	Form	Description
Tree Node	<b>Default</b>	An empty default Form. You can override this Form to present the tree node's model specifically for this Navigation Tree, and assign it to the node by choosing <NONE> in <b>Detail UI</b> and this Form in the <b>Form</b> properties in <b>General</b> tab of the node.
	<b>Menu</b>	The generic menu Form that presents the tree node model's Menu Form.
Finder Node	<b>Default</b>	An empty default Form. You can override this Form to design the Finder's search Form specifically for this Navigation Tree, and assign it to the node by choosing <NONE> in <b>Detail UI</b> and this Form in the <b>Form</b> properties in <b>General</b> tab of the node.
	<b>Menu</b>	The generic menu Form that presents the Finder's Menu Form.
Finder Child Node	<b>Default</b>	An empty default Form. You can override this Form to present the Finder's output document specifically for this Navigation Tree, and assign it to the node by choosing <NONE> in <b>Detail UI</b> and this Form in the <b>Form</b> properties in <b>General</b> tab of the node.
	<b>Menu</b>	The generic menu Form that presents the Finder output document's Menu Form.
Root Node	<b>Default</b>	The default Form that displays the overall Navigation Tree, which includes: <ul style="list-style-type: none"><li><i>Tree Form Element</i> - located at the left side of the Form, displays the structure of the Navigation Tree as a browsable tree. The user can expand or collapse tree nodes in this Form Element as well as view details of the tree node's model by clicking the node</li><li><i>Detail Pane</i> - located at the right side of the Form, is a Form Frame that displays the details of the tree node's model when the node is clicked.</li></ul>
	<b>Menu</b>	The generic menu Form that presents the tree node model's Menu Form.
Root Node with Tab	<b>Default</b>	The default Form that displays the overall Navigation Tree with Tabs. This is the same as the Navigation Tree's <b>Default</b> Form, except that the detail Form Frame may show <b>Single</b> Form or <b>Tabs</b> Form depending on Variable <i>currentForm</i> .
	<b>Menu</b>	The generic menu Form that presents the tree node model's Menu Form.
	<b>Single</b>	The non-tab version of detail Form that can be displayed in the detail pane.
	<b>Tabs</b>	The tabset version of detail Form that can be displayed in the detail pane. The Form is implemented by the use of Iterator Form Element on array UI Variable <i>tabs</i> to display child nodes' model as Tab Frames in the Tabset.

**Note:** Velocity Studio features require a separate license key that is activated in Velocity Studio.

## Navigation Trees at Runtime

Presented below is a sample Navigation Tree at runtime, [displayed within an application](#) in a project via menu item in the application. This is an illustrative sample which demonstrates the significant runtime features of Tree Node, Finder Node and Recursive Node.

In addition, a great reference implementation of the Navigation Tree is the User Profile Manager in System Administration App; you can find the metadata in `cwf_up / Presentation / Navigation Trees / upadminTree` in **Library** tab of the Velocity Studio's Navigation Pane.

The screenshot shows a user interface for managing customer data. On the left, a tree view titled "Customer Tree" displays a hierarchy of contacts under a "customerTree" node. A specific contact, "Bobby Smith", is selected and highlighted with a blue background. To the right of the tree, a "customer" form is displayed, containing fields for "System document ID" (16100001), "First Name" (Bobby), "lastName" (Smith), "active" status (True), and a date field "duedate" (5/28/2010). At the bottom of the screen, there is a "Save" button. Three red boxes are overlaid on the interface to identify its components: a box around the tree view labeled "Tree Form Element", a box around the detail form labeled "Detail Pane", and a box around the "Save" button labeled "Menu".

The Navigation Tree runtime is rendered via the **Default** Form in the Root Node of the Navigation Tree. By default, this Form displays several important components:

- *Tree Form Element* at the left-side - which enables the user to browse the Navigation Tree. Nodes with children can be toggled to expand/collapse by clicking the expand/collapse icon.
- *Detail Pane* at the right-side - it is empty until user clicks on a node, which displays the node's model object Form.
- *Menu* at the bottom - appears and displays the model object's menu when user clicks on a node. This menu is not within the **Default** Form of the Navigation Tree, but rather rendered via the Application's Menu Form.

Many of the runtime appearance and behaviour of the Navigation Tree can be changed, via Variables and Methods. Please see the [Variables](#) and [Methods](#) tab for the list of the system-defined variables and methods for details.

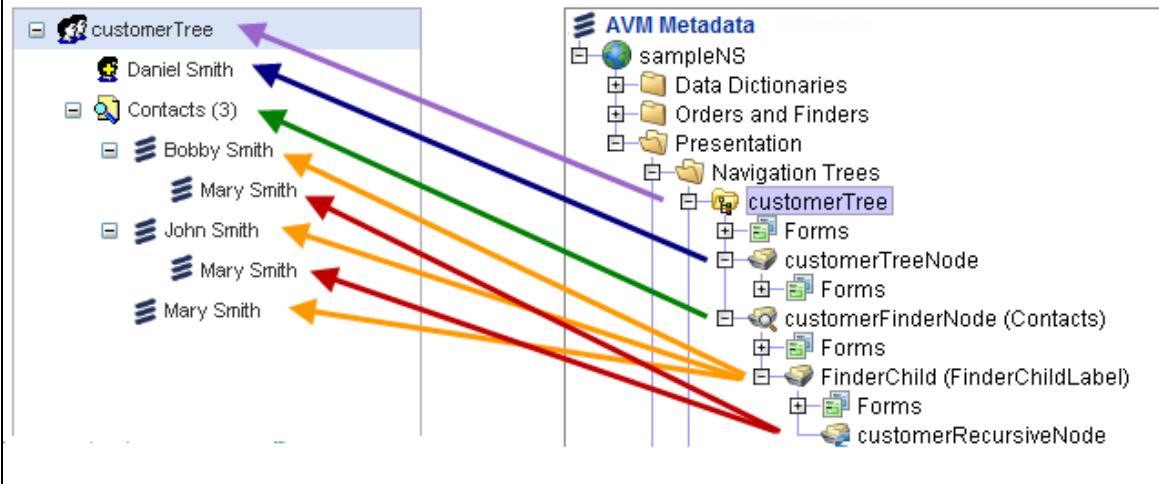
For example, the label appearing on node is determined by `onNodeVisualKey` method. By default, the node's visual key is determined by the object's Visual Key method (for example, `cwOnDocVisualKey`) if specified, the object's label or the node's label, in that order of precedence.

As another example, the icon of the node can be modified by the `onNodeIcon` method or use the **Image** property in the **General** tab.

Below diagram shows the mapping of tree nodes rendered at runtime to the tree nodes in Velocity Studio at design time.

## Runtime

## Design time

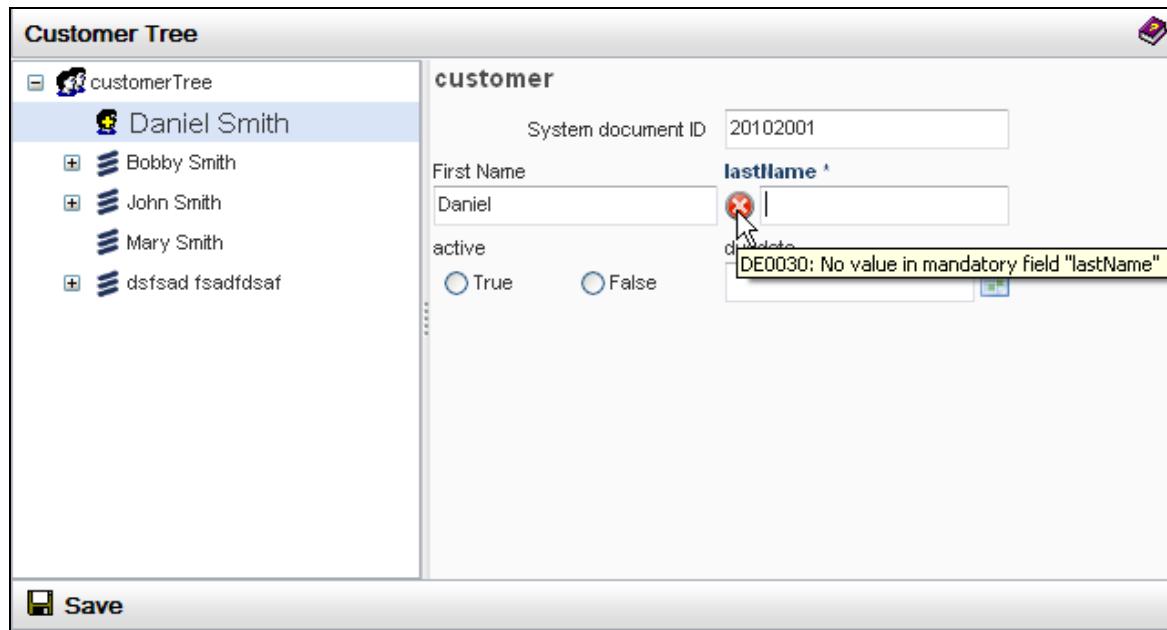


Note that the Tree loads a tree node of most types "on-demand", when user (or programmatically) expands the node. Then the node loads and creates all child nodes; for Finder nodes, finder Select operation is performed (see Finder Node below). The exception is with Order or Data Structure Item Nodes, where the source data is loaded at once.

### Tree Node

A tree node at design time is a single node at runtime, as exemplified by *customerTreeNode* in our example. Clicking the node at runtime shows its **Form** specified in **General** tab at the detail pane.

Runtime validation occurs similarly on the tree node's model when the Form is modified by user and click **Save** (or, if **Auto save** is enabled at design time, whenever the user clicks away to another node in the Navigation Tree).



### Finder Node (and its children)

A Finder Node at design time becomes a single node of its children nodes at runtime, which is the parent node of the Finder results and represented by the Finder child node at design time. In our example, the Finder Node *Contacts* has three children nodes as returned by the Finder's execution, *Bobby Smith*, *John Smith* and *Mary smith*. These children nodes are characterized by the *FinderChild* node at design time.

By default, the Finder Node displays the **Search** Form of the Finder (but can be customized by **Form** property in **General** tab of the Finder Node).

**Customer Tree**

The screenshot shows a desktop application window titled "Customer Tree". On the left is a navigation tree with a root node "customerTree" containing "Daniel Smith" and a child node "Contacts (3)" which contains "Bobby Smith", "John Smith", and "Mary Smith". On the right is a search form titled "customerSearch" with two input fields: "First Name" and "lastName". Below the search form are standard window controls: "Search", "Select", "New", "Edit", and "Delete".

The user can populate the Search Form and click the **Search** menu command to invoke the Finder, which updates the children nodes in the Navigation Tree.

The screenshot shows the same application window after a search. The "Contacts (3)" node now has only one child node, "Bobby Smith", while "Daniel Smith" remains in the tree. The search form still displays "First Name" and "lastName" fields with the values "B" and "Smith" respectively. The "Search" button is highlighted with a yellow glow, indicating it was just clicked. The other buttons ("Select", "New", "Edit", "Delete") are visible below.

The Finder Child node always displays the Form of the *output* object of the Finder (in our example, an output Document *customer* as Contact).

**Customer Tree**

The screenshot shows a navigation tree on the left with a single node 'Bobby Smith'. The main panel displays a 'customer' record with the following fields:

- System document ID: 16100001
- First Name: Bobby
- Last Name: Smith
- active: True (radio button selected)
- duedate: 5/28/2010

**Save**

At design time, a popular property enabled at the Finder node is the **Exclude** flag in **General** tab, which hides the Finder node, and directly positions Finder Children nodes at its place. This is useful when the Finder mechanics are to be hidden from the user.

The screenshot shows a navigation tree on the left with a node 'Bobby Smith' expanded, revealing three recursive children: 'Mary Smith', 'John Smith', and another 'Mary Smith'. The main panel displays a 'customer' record for 'Bobby Smith' with the same fields as the previous screenshot.

**Save**

## Recursive Node

In our example, the Recursive Node is positioned below the Finder Child node, and the target of the recursion at the Finder node. In essence, the objective of the example here is to recursively invoke the Finder to find descendants of *Contacts*. This is accomplished by, at the Finder node, defining initialization conditions of the Finder Input Document to search based on the node's parent model. Furthermore, by checking **exclude** flag in **General** tab of the Recursive Node, the Finder Nodes are hidden on recursion. Thus at runtime, the children Contacts appear directly below its parent Contact.

In the sample data of our example, *Mary Smith* is a descendant of both *John Smith* and *Bobby Smith*. Meanwhile, *Mary Smith* has no descendants, and thus the recursion ends at her. (Otherwise, she will have children nodes and the recursion continues.)

The "Recursion on Finder Node" is a commonly-used approach to present dynamically recursive data structures at tree nodes in the Navigation Tree. See the implementation of *Org chart* in User Profile Manager of System Administration App as a reference of this approach; pay attention on *onInit* Method to understand the setup of Finder Input Document that makes possible the recursion.

## Navigation Tree with Tabs

A variation of the Navigation Tree is capable of displaying children nodes of a tree node as tabs in a Tabset in the Detail Pane.

**Customer Tree with Tabs**

customer

System document ID: 12312342134

First Name: **lastName \***  
Mary

active:  True  False

duedate:

**Save**

Navigation Tree with Tabs are typically desirable in showing an array of items as a single node in the Navigation Tree. The display of individual item is then organized by tabs at the Tabset. Common examples include line-items of an Order, equipment list of an Order, etc. In our example as shown in screenshot above, we have condensed the *Contacts* Finder node to show all the results of the Finder (*Mary Smith*, *John Smith* and *Bobby Smith*) as tabs in the Detail Pane. Notice that these items no longer appear as children nodes of the Finder node. (Note: the Recursive Node from the original example is removed here).

To enable this tabularization feature in Navigation Tree, you must initially create your Navigation Tree based on `com.conceptwave.system.NavigationTreeWithTabs`, which is a derived system metadata object of the base Navigation Tree. In our example, we then enable **Display in parent tabs** flag in **General** tab at the Finder Child node. Furthermore, the **Form** property is deselected as `<NONE>` at the Finder node, to avoid displaying Finder Search Form as the first tab of the Tabset.

The presentation of Navigation Tree with Tabs works by having two extra Forms at the Root node -- the **Single** Form and the **Tabs** Form, which renders the Detail Pane. The developer may choose to display either of the two Forms (or even, other customized Forms) by the `currentForm` Variable. The Single Form renders the detail pane similarly as the original Navigation Tree, which is limited to display the Form of the tree node's model. On the contrary, the Tabs Form renders the detail pane with a Tabset Form Frame with Iterator on the `tabs` User Interface array Variable, which are the User Interface objects to be shown in the Tab Frames.

## Tree Table

The Tree Form Element in the root Default Form of the Navigation Tree may be configured differently to create different presentations of the Navigation Tree. For instance, the Tree Form Element may contain multiple children Form Elements (note: the default has the Tree Form Element contain one child Label Element, which displays a read-only label for the node) to become a *Tree Table*, which displays multiple columns at the Tree.

**Customer Tree**

customer

System document ID: 16100000

First Name: **lastName \***  
John

active:  True  False

duedate:

**Save**

Typically, the additional columns of the Tree displays essential Variables of the tree node models. This information is modelled by adding Variables to the

Table Document (see property **Table Document** in [General Tab](#)), and the data is sourced from the node's model object mapped by the Table Conversion Map. In our exemplary screenshot above, the **active** checkbox is created by first adding such Variable at the Table Document, map it against the existing **active** Variable of node's model *Customer* at the Table Conversion Map, and then create the Checkbox Form Element underneath the Tree Form Element using Variable *navTree.active*.

You can see some more details on how to setup Tree Table at [Tree Table](#) section of Tree Form Element.

These columns in the Tree Table are automatically updated when the node's model is updated. In our example, the **active** checkbox is updated in the Tree whenever a user clicks on a customer node, edits the customer at the detail pane and clicks **Save**.

The columns at the Table Tree can also be made editable by the **Editable** property of the child Form Element of the Tree. However, the editing at Table Tree does not automatically save at the node model object. (for example, if our **active** checkbox is made editable, by default, toggling the checkbox does not get reflected into the model *customer*.) You must overwrite the **onNodeEdited** Method to do so; see this method at the [Methods tab](#) for details.

## Miller Table

Another style of presentation of the Navigation Tree is the Miller Columns. Instead of showing children nodes underneath the parent node in an indented way, Miller Columns expands an extra column at the right to display children nodes when the parent node is clicked, and recursively so.

	Charles Madigen (10)	John Garrison (13)	Galina Ugarova
 Charles Madigen	 Ralph Brogan  Carol Finley  Gene Porter  Rogine Leger  Abigail Lippman  John Garrison  Rui Shu  Kirill Amirov  Joan Little	 Anna Galetti  Glenda Stockton  Rong Xu  Galina Ugarova  Brenda Cox  Alexia Badger  Pat Freel  Cameron Miller  Lori Newton	No items to show.

Miller Columns are typically used to present homogeneous trees (that is, all nodes are of the same object type). To use Miller Columns in Navigation Trees, simply override the root **Default** Form, then **Copy & Replace** the Tree Form Element, and set true to **Miller Columns** property of the Tree Form Element. You can also set false to **Hide Header** property, which displays the selected parent node at the header label of the child column, as depicted in the previous screenshot.

**Note:** You can deselect a parent node in a Miller table. Additionally, single- and double-click operations are available to select the child node display. Single-clicking highlights the record and double-clicking proceeds with the operation.

You may want to move the detail pane to be displayed below (instead of to the right) of the Tree Form Element, since Miller Columns require a large horizontal real estate.

Name	Value
Name	tree
Label	
Variable	navTree
Auto Fit Max Height	
AutoFit Data	
AutoFit Max Width	0
Cell Height	0
Cell Width	0
Double Click Method	
Editable	
Filter Editor Height	0
Generate DoubleClick on Enter	<input type="checkbox"/>
Header Height	0
Header Style	
Height	100%
Hide Header	<input checked="" type="checkbox"/>
Hover Form	
Hover Method	
Hover Style	
Hover Variable	
Icon Size	0
Include in Tab Order	
Increase Pre-render	<input type="checkbox"/>
Indent Size	0
Load Data Message	
Miller Columns	<input checked="" type="checkbox"/>
No items Message	

**Note:** Velocity Studio features require a separate license key that is activated in Velocity Studio.

#### Drill-up Method for a Miller Table to Simulate Navigation Tree Expand Action

To implement a drill-up method for a Miller Table to simulate the Navigation Tree expand action, complete these steps:

1. Override the `onNodeDetail` method of the Finder node's result node.
2. Find the selected node by using the `findNodeFromModel` method.
3. Expand the Miller Table using the selected node. For the navigation tree, use the following statement to expand the tree node:

```
selectedNode.getRoot().treeSel = selectedNode
```

4. Ensure that you have the `Show Multiple Columns` property set to true for your Tree element.

# User Interface User Guide

---

[User Interface](#) is a metadata object that implements the controller of the architecture pattern. There are two different types of User Interface available:

- Included User Interface
- Top-level User Interface

User Interfaces are defined within a metadata object that may contain User Interface objects such as Documents, Orders, and Finders. With an included User Interface, the model in MVC pattern connects with the controller. Top-level User Interfaces are defined as a stand-alone metadata object, although their model is attached to the data source. Either type of User Interfaces provides data and business logic access to the View in the MVC pattern using variables and methods. In this user guide the following topics are covered on how to customize a User Interface Application with the built in functionality of User Interfaces.

- [\*\*User Interface Application\*\*](#)

This page contains an overview of how to customize an Application User Interface with the built in application for the User Interface module, how to create application mens and provide a custom logo to the application.

- [\*\*Detail User Interface\*\*](#)

This page describes how to create a Detail component User Interface, including how to use built in metadata object to create different forms such as Detail form, Tabbed component form, and Result component form.

- [\*\*Finder User Interface\*\*](#)

Finders are used to search, access, insert, update, and delete data from different data sources such as databases and external systems. This section describes how to create different types of finder user interface, implement the finder functionality, redesign an existing finder, and usage.

- [\*\*Menu Style\*\*](#)

A menu can be created with a hierarchy of Menu elements. Expandable submenus of a menu item can be created item by adding child Menu elements. This section describes how you customize style of menus and submenus.

- [\*\*Dialogs\*\*](#)

Dialogs are used to display information related to the current user interface, without leaving the page. This page contains information how to create a dialog and configure it to display a warning message and user action buttons. The page also describe how to change the grid layout style.

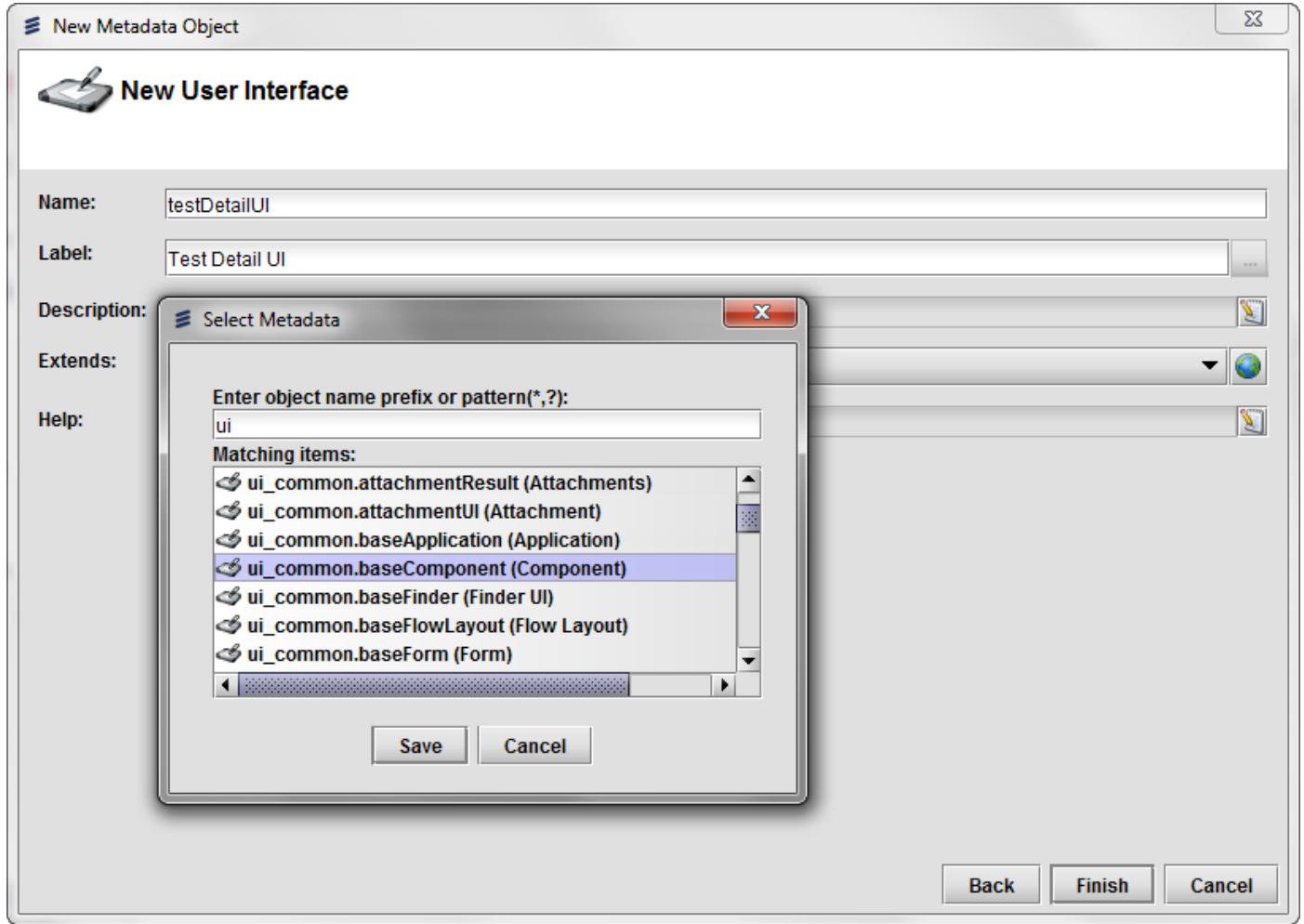
- [\*\*Messages\*\*](#)

Messages are sent to an object, specifying a request for action. This page describes types of messages and how to use them with different user interfaces.

## Detail User Interface

To create a detail component user interface, follow these steps:

1. [Create a new user interface](#), which extends `ui_common.baseComponent`.



2. Navigate to the **Variables** tab.
3. Right-click the **model** variable and select **Override**; it creates a local copy of the variable.
4. Set the **Data type** property to the desired object.
5. To show a new data object using this component, click the **Methods** tab and override the `getNewDetailModel` method to provide the data object instance. Some data objects require a call to specific API, while others only need a simple script such as:  

```
return new Document("testApplicationUI.testDoc");
```
6. To provide the functionality of saving the data object, when you click the **Save** button, override **onSaveModel** method in the methods list.

## Detail Form

There are two ways to configure the detail form. You can use either an internal detail form or an external one.

### Use an Internal Detail Form

To internally configure the detail form, follow these steps:

1. Right-click the **detailForm** form and select **Override**.
2. To add more elements right-click the **formGridLayout** and select **Add**.
3. Set the **Column Span** property to 2 or more for each field added.

**Note:** It is not a good practice to override **formGridLayout**, as it contains styling used throughout the modules.

### Use an External detail form

To create an external detail form, follow these steps:

1. [Create a new user interface](#), which extends **ui\_common.baseForm**. By default, this user interface contains one grid layout with seven columns. This user interface allows two fields per row.
2. The **Column Span** property of each field is set to 2.
3. To start using this user interface, navigate to the **Variables** tab.
4. Right-click the **model** variable and select **Override**, which creates a local copy of the variable.
5. Set the **Data type** property to the desired object.
6. Once the model is set up, right-click the **Default** form and select **Override**.
7. To add more elements right-click the **formGridLayout** and select **Add**.
8. Set the **Column Span** property to 2 or more for each field added.

**Note:** It is not a good practice to override **formGridLayout**, as it contains styling used throughout the modules.

When the external detail form is created, it can be used to be fully functional. To display the detail form created previously, navigate to the detail component UI previously created, and complete these steps:

1. Click the **Methods** tab, and then click the **Show Base Methods** () button.
2. Right-click the **getComponentForm** method from the list and select **Override**. This method can be implemented in a few ways:
  - o If the component is re-usable, it is a good practice to create an event that returns a *String* representation of the full name of the detail form. For example, an event can be created using the event ID. If *UI\_COMMON\_ORG\_CHART\_COMPONENT\_FORM* is an event id, then in **getComponentForm** method, a publish can be called for this event as follows:

```
return ui_common.doPublishEvent("UI_COMMON_ORG_CHART_COMPONENT_FORM",
FIRST_ONE,this.model,"cwa_security.orgChartNodeForm");
```

- o If this component is not reusable, return the detail form UI full name. For example:

```
return cwa_security.orgChartNodeForm;
```

## Tabbed Component Form

To create a tabbed component form, follow these steps:

1. [Create a new user interface](#), which extends **ui\_common.baseTabbedComponent**.
2. To add a tab button, override the **default** form.
3. Right-click **tabButtonSample** and select **Copy**.
4. Right-click the **tabButtonGrid** element and select **Paste**. Specify following properties:
  - o **Label**
  - o **Click Method** – To change the currently displayed form, the **Click Method** needs to call the *setFrame* method by passing in the UI to display. This UI has an implementation of **ui\_common.baseComponent**.
  - o **Visible** – Set to **False** by default.

## Result Component Form

To create a detail user interface with result form, follow these steps:

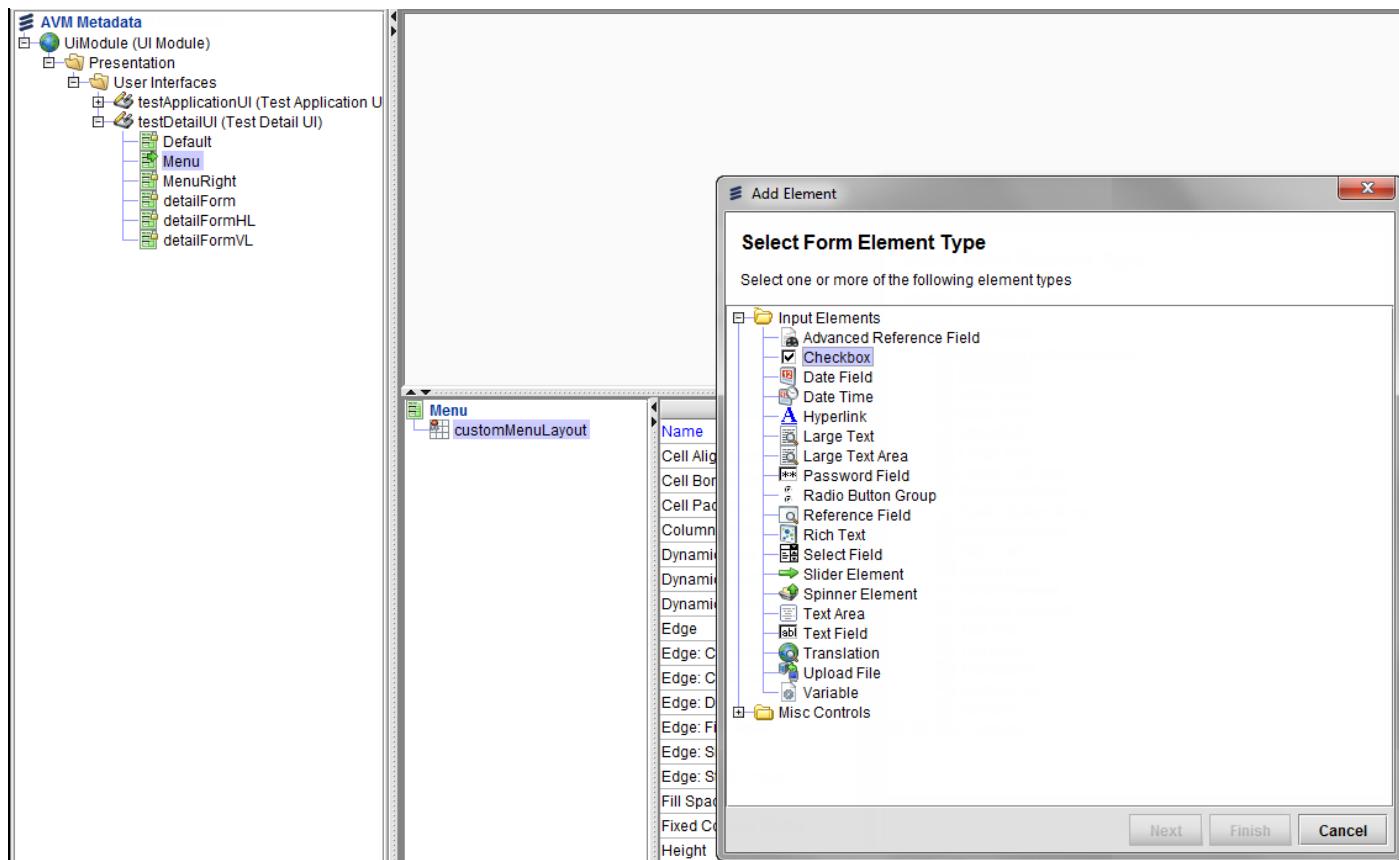
1. [Create a new user interface](#), which extends **ui\_common.baseResultComponent**.
2. Navigate to the **Variables** tab.
3. Right-click **model** variable and select **Override**, which creates a local copy of the variable.
4. Set the **Data type** property to the required data object.
5. To show a new data structure using this component, override **getNewDetailModel** method to provide the data structure instance. Showing a new data object is similar to detail component.
6. This component form also needs to know, how to save the data object, when you clicks on the **Save** button. To provide this functionality, click the **Methods** tab and override **onSaveModel** method.

Default result form is set to **Result** form and to use other result forms, assign the form to variable **resultForm**. This UI contains two result forms (see [Create a Result Form](#) for more details).

## Custom Menus

You can add more menus to the application, if needed. To add more menus, follow these steps:

1. To add a menu, right-click either the **Menu** or **MenuRight**.
2. Right-click **customMenuLayout** to add more elements (for example, text field or check box).



**Note:** The **Menu** form adds menu items to the left of the default menu items. The **MenuRight** adds menu items to the right of the default menu items.

# User Interface Application

This page describes how to customize User Interface (UI) application with the built in [application](#) functionality for the User Interface. The following topics are covered:

- [Implement an application](#)
- [Create application menus](#)
- [Provide an application logo](#)

## Implement an Application

A user interface application can be implemented with the same style built in application for the User Interface (UI) module, which consist of these steps:

1. [Create a new user interface](#) (for example, TestApplicationUI).
2. For the **Extend** field, choose **ui\_common.baseApplication** and specify the **URL Mapping** for your application (for example, /TestUIApp).

## Create Application Menus

To create application menus, complete these steps:

1. Expand your application's **User Interfaces** node, then right-click the **Page** form and select **Override**.
2. Provide a value to **Title** property (for example, TestUIBase). By default, base application displays five application menus:

Menu	Description
<b>homeMenu</b>	Display menu when you click on <b>Home</b> menu. Override the <b>homeMenu</b> to add an action to the menu.
<b>createMenu</b>	Display screens to create new objects, such as users, address, and so on. These menus appear under the <b>Create</b> Menu.
<b>manageMenu</b>	Display finders to allow you to manage objects. These menus appear under the <b>Manage</b> menu.
<b>configMenu</b>	Allows you to configure the application. These menus appear under the <b>Configure</b> menu.
<b>viewMenu</b>	Allows you to see system information, such as the event log. These menus appear under the <b>View</b> Menu.

To override the visibility of these application menus, follow these steps:

1. [Create a rule set](#) of type **Initialization**. Choose UI application as **Source** (for example, TestApplicationUI).

2. Click the **Show Base Methods** ( ) button. Provide a return value for the following variables:

- isMenuHomeVisible
- isMenuCreateVisible
- isMenuManageVisible
- isMenuConfigureVisible
- isMenuViewVisible

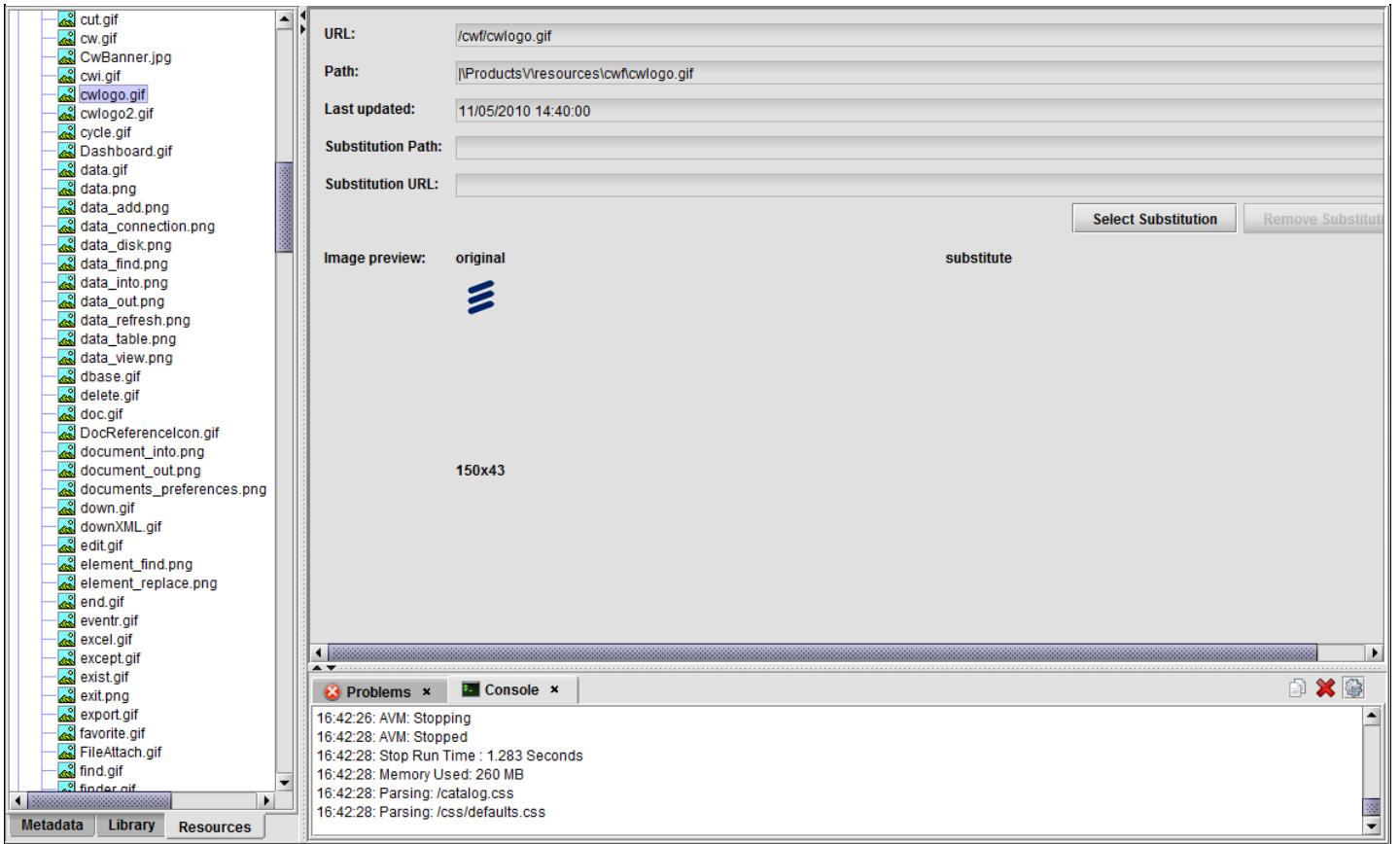
You can add more menus to the application, if needed. To add more menus, follow these steps:

1. Right-click the **Menu** form and select **Override**.
2. Navigate to the **manageMenu** element.
3. Right-click **fillMenu1** element and select **Copy**; it ensures that the base styling is used.
4. Right-click the application or user interface, where **fillMenu1** is required and select **Paste**. Set the following properties:
  - **Label**
  - **Click Method**
  - **Visible**: By default, this property is set to false. Set this property to none or a permission method.

## Provide an Application Logo

The default application logo can be provided in two ways:

- Navigate to the **Resources** tab in the Velocity Studio and complete these steps:
  1. Find **cwlogo.gif** under the **cwf** folder.
  2. Click the **Select Substitution** button and select the image to be used.



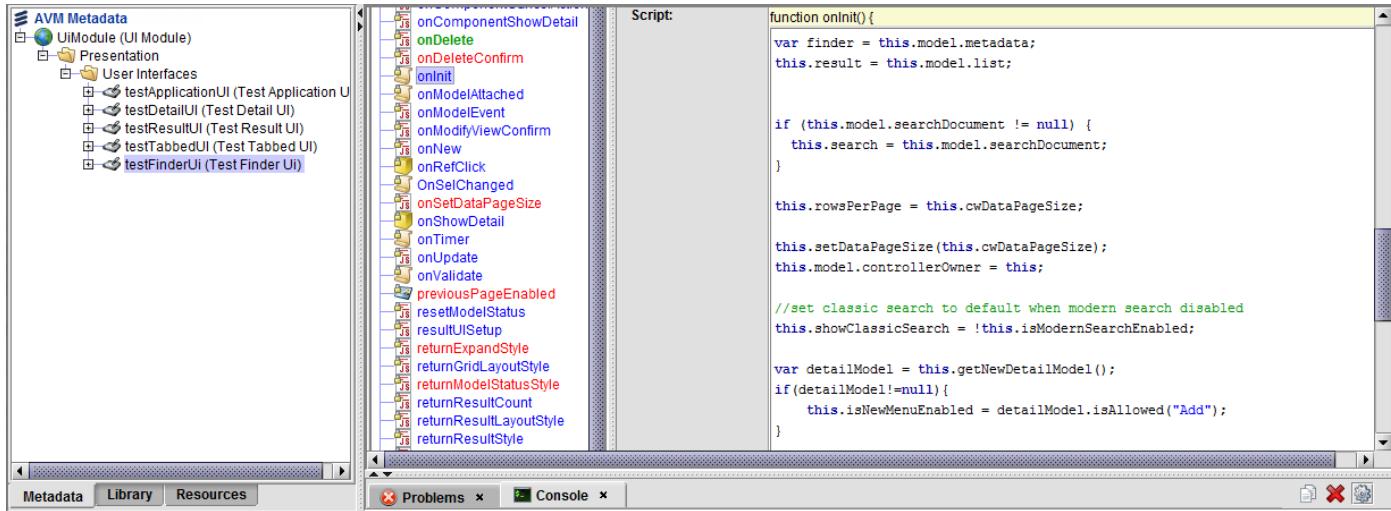
- Create a .css file and save in the metadata's resource directory.
  1. In this file, provide styling for `cwAppLabelWithLogo`.
  2. `Background-image:url('../cwf/cwlogo.gif')`.
  3. Once the .css file is created, re-open the file in the metadata, in the Velocity Studio.
  4. Navigate to the application's **Page** form and set the **Stylesheet** property to the .css file that is created.

## Finder User Interface

To create a new finder User Interface (UI), create a [new user interface](#), which extends `ui_common.baseFinder`. By default, this finder UI has two types of search forms:

- **Search** – This is the modern search form. Some data objects support Oracle-type searches. For more information, refer to the API.
- **classicSearch** – This is the basic search form, in which fields are bound to a search object.

To show either modern search or classic search, override the **OnInit** method from **Methods** tab of the finder UI, and set the values for **isModernSearchEnabled** and **isClassicSearchEnabled** variables.



The screenshot shows the AVM Metadata interface with the 'Methods' tab selected for a 'testFinderUI' component. The left pane displays a tree view of UI modules and their components. The right pane shows a list of methods with icons and their corresponding script definitions. The 'Script:' tab is active, displaying the following JavaScript code:

```
function onInit() {
    var finder = this.model.metadata;
    this.result = this.model.list;

    if (this.model.searchDocument != null) {
        this.search = this.model.searchDocument;
    }

    this.rowsPerPage = this.cwPageSize;
    this.setDataPageSize(this.cwPageSize);
    this.model.controllerOwner = this;

    //set classic search to default when modern search disabled
    this.showClassicSearch = !this.isModernSearchEnabled;

    var detailModel = this.getNewDetailModel();
    if(detailModel!=null){
        this.isNewMenuEnabled = detailModel.isAllowed("Add");
    }
}
```

**Note:** If both search types are enabled, the modern search displays automatically. To display classic search in runtime, click the arrow icon beside the search menu. If the modern search is hidden, the classic search form displays by default.

To disable the finder results editing, set the **isDetailEditable** variable to false by following these steps:

1. Select the **isDetailEditable** variable.
2. Click the **Methods** tab of the variable and right-click the finder UI (for example, `testFinderUI`).
3. Select **New Initialization** and in the script use `return false;` to set the variable to false.

Name	Type
isDeleteMenuEnabled	com.conceptwave.system.Boolean(Boolean)
isDeleteMenuVisible	com.conceptwave.system.Boolean(Boolean)
isDetailEditable	com.conceptwave.system.Boolean(Boolean)
isEditable	com.conceptwave.system.Boolean(Boolean)
isExportMenuEnabled	com.conceptwave.system.Boolean(Boolean)
isExportMenuVisible	com.conceptwave.system.Boolean(Boolean)
isGradientStyle	com.conceptwave.system.Boolean(Boolean)
isGridStyleWide	com.conceptwave.system.Boolean(Boolean)
isHasSearched	com.conceptwave.system.Boolean(Boolean)
isIconVisible	com.conceptwave.system.Boolean(Boolean)
isImportMenuEnabled	com.conceptwave.system.Boolean(Boolean)
isImportMenuVisible	com.conceptwave.system.Boolean(Boolean)
isModelStatusMenuEnabled	com.conceptwave.system.Boolean(Boolean)
isModelStatusMenuVisible	com.conceptwave.system.Boolean(Boolean)
isModernSearchEnabled	com.conceptwave.system.Boolean(Boolean)
isNewMenuEnabled	com.conceptwave.system.Boolean(Boolean)

**General Methods**

**cwLeafInitAction\$isDetailEditable**

**Name:** cwLeafInitAction\$isDetailEditable  Private

**Leaf:** ui\_common.baseFinder.isDetailEditable

**Description:**

**Script:**

```
function cwLeafInitAction(document) { // Leaf initialization
    return false;
}
```

## Create a Classic Search Form

There are two ways to implement the classic search:

- [Use an internal form in the base finder.](#)
- [Use an external form and link it to the base finder by an event.](#)

### Use an Internal Search Form

1. Navigate to the **Variables** tab and find **searchForm** variable.
2. Add an **init** method for this variable and return the string *classicSearch*.
3. To internally configure the search form, once the search model is set up, right-click the **classicSearch** form and select **Override**.
4. To add new elements, right-click the **formGridLayout** and select **Add**.
5. Set the **Column Span** property to 2 or more for each field added.

**Note:** It is not a good practice to override **formGridLayout**, as it contains styling used throughout the modules.

### Use an External Search Form

To implement a classic search form using an external form, consider these steps:

1. [Create a new user interface](#) that extends **ui\_common.baseForm**.
2. Navigate to the **Variables** tab.
3. Right-click the **model** variable and select **Override**. Provide the data type to be used for the search data object.
4. Once the model is setup, override the **Default** form and add fields to the **formGridLayout**.
5. Set the **Column Span** property to 2 or more.
6. Navigate back to the finder UI and override **getClassicSearchForm** method. This method can be implemented in two ways:
  - o If the finder UI is re-usable, it is a good practice to create an event which returns a *string* representation of the full name of the detail form. For example, an event can be created with the event id **UI\_COMMON\_ORG\_CHART\_SEARCH\_FORM**. In **getClassicSearchForm** method, a **publish** can be called for this event:

```
return ui_common.doPublishEvent("UI_COMMON_ORG_CHART_SEARCH_FORM",
FIRST_ONE,this.model,"cwa_security.orgChartNodeSearchForm");
```

- o If this finder UI is not reusable, return the detail form UI full name. For example:

```
return cwa_security.orgChartNodeSearchForm;
```

## Create a Result Form

The result form can be customized by **Selection**, **Editable**, and hiding **Detail Icon Column** (see [Customize the Result Form](#) for more details). There are two ways to implement the result form:

- [Use an internal form in the base finder.](#)
- [Use an external form and link it to the base finder by an event.](#)

### Use an Internal Result Form

1. Navigate to the **Variables** tab and find **resultForm** variable.
2. Add an **init** method for this variable and return the string **Result**.
3. To configure the result form internally, right-click the **Result** form and select **Override**.
4. To add new elements right-click the **resultList** and select **Add**.
5. Set the **Column Span** property to 2 or more for each field added.

**Note:** It is not a good practice to override **resultList**, as it contains styling used throughout the modules.

### Use an External Result Form

To display the list of objects returned by this finder, provide the search functionality. For example, if finder is a script finder then implement **cwOnFinderSel** at the finder level. Once the implementation is done, follow these steps:

1. [Create a new user interface](#), which extends **ui\_common.baseResult**.
2. Navigate to the **Variables** tab and create a new variable called **result**.
3. Set the **Data Type** to the type of data object needed and set the **Array** to true.
4. Override the **Default** form and right-click **resultList**. Select **Extend**, which allows you to change the table element's **Variable** property. Set it to the variable **result** created in step 2.
5. To add fields under the table element, right-click **resultList** and select **Add**.

When the result form is completed, navigate back to the finder UI's **Methods** tab. Override the **getResultSet** method and return the full name of the result form, by either publishing an event or just returning the full name.

## Customize the Result Form

The result form can be customized in the following ways:

Type	Details
<b>Selection</b>	<ul style="list-style-type: none"><li>• Checkbox or Simple – Use <b>Default</b> for checkbox and <b>ResultSimpleSelect</b> for simple</li><li>• Multiple or Single table selection – By default multiple is on. To restrict to single selection, set the <b>cwTableSelection</b> variable to single.</li></ul>
<b>Editable</b>	To make the result editable, use the <b>ResultEditable</b> form and set the <b>isTableEditable</b> variable to true.
<b>Detail Icon Column</b>	To disable the detail icon column, override the <b>onInit</b> method of the finder UI and add a call to <b>this.setShowDetailEnabled(false)</b> . Passing in the Boolean false disables the column. To enable the column later, pass the Boolean true.

## Custom Menus

To add additional menus to the finder, follow these steps:

1. Overriding the result user interface **Menu** form.
2. Add elements to **customMenuLayout** (see [Custom Menus](#) for more details).

## Implement Finder Functionality

The finder functionality can be implemented by providing the following:

### Provide the Detail Form

To provide the detail form to use, when you click on the detail icon, override the **getDetailForm** method by returning the full name of the detail form (for example, the detail component created in [using the detail form](#)). To display the detail form properly, the data object to display must be provided by implementing **getDetailModel** method. Some implementations of this method can be as simple as returning the parameter called **selected**. For others, an API can be called.

### Provide New Functionality

By default, the **Add** menu is shown when the finder UI is in edit mode. To hide the **Add** menu, set the **isNewMenuVisible** variable to false. To provide functionality for this menu, override the **getNewDetailModel** method on the finder UI. This method returns a new instance of the data object to be saved. The finder UI uses the form returned by **getDetailForm** method.

## Provide Delete Functionality

By default, the **Delete** menu is shown when the finder UI is loaded. To hide the **Delete** menu, set the **isDeleteMenuItemVisible** variable to false. To provide functionality for this menu, override **onDelete** method. By default, there is one parameter passed in, representing the data object selected in the UI.

## Redesign an Existing Finder

To redesign an existing finder follow these steps:

1. [Create a new user interface](#), which extends `ui_common.baseFinder`. Set the **Select input** and **Select output** objects as that of the old finder.
2. Override the **onInit** method of the finder UI, and set the **isClassicSearchEnabled** variable.
3. To provide the classic search form, ensure that the search object of the finder **search** is set with the correct data type property.
4. Right-click the **classicSearch** under the finder UI and select **Override**.
5. Copy fields from the old search form to the **classicSearchGridLayout** layout element. Ensure that the fields use variables from the search object of the finder **search**.
6. Navigate to the **Variables** tab and find **searchForm** variable. Add an **init** method for this variable and return the string `classicSearch`.
7. To provide the result form, right-click the **Result** form under the finder UI and select **Override**.
8. Right-click the **resultList** element and select **Extend**.
9. Set the **Variable** property to the **result** variable. Necessary fields can be added under this table element.
10. Navigate to the **Variables** tab and find **resultForm** variable. Add an **init** method for this variable and return the string **Result**.
11. Follow [Implement Finder Functionality](#) to provide further functionality.
12. To add custom menus, override the **Menu** form. Add menus under the **customMenuLayout** as needed.

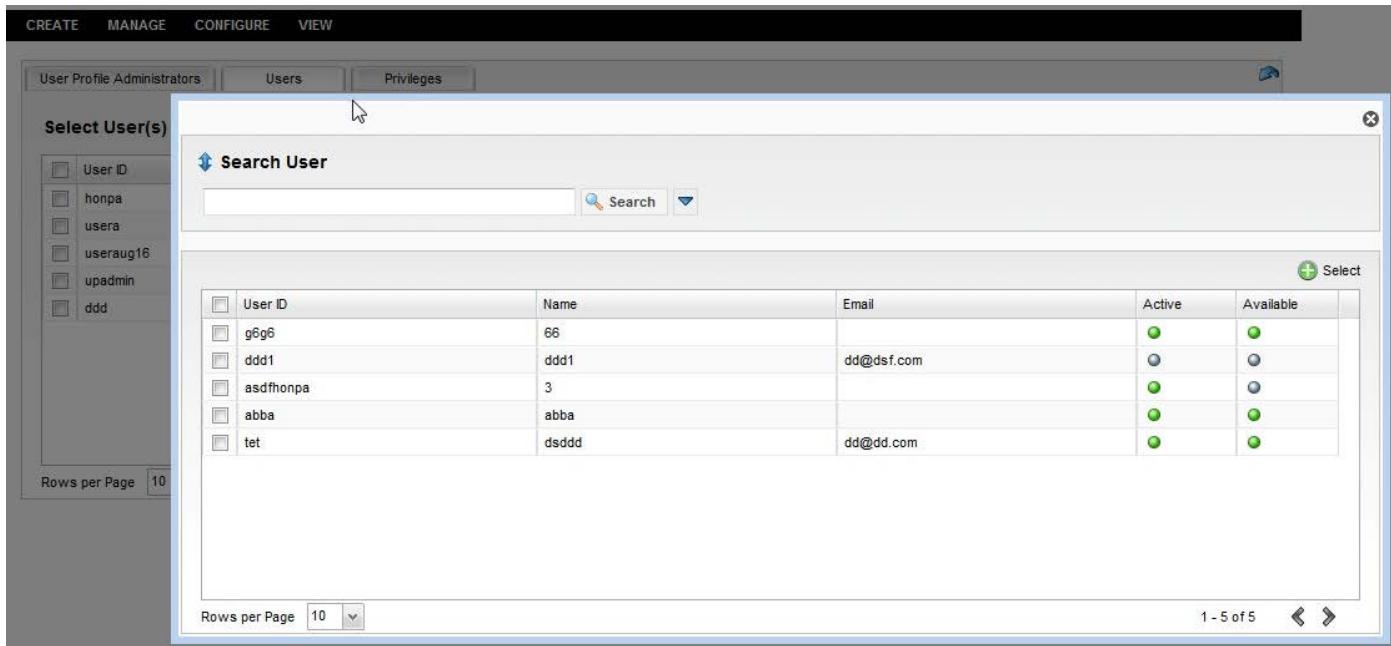
## Reference Finders

To create a finder that can be used as a top-level object and a reference finder, follow these steps:

1. [Create a new user interface](#) which extends `ui_common.basePrimaryFinder`. Set the **Select input** and **Select output** objects as that of the old finder.
2. Follow the rest of the steps in [Finder UI](#).
3. For finders that are used as second-level finders, such as the user finder, the Groups UI needs to implement `onSelect` method.

## Third-Level (popup) Finders

A third-level finder is a finder that is displayed within a second-level finder. The primary use is to add a subset of objects as a property of a component. For example, the *Group detail* form in `cwa_security` contains a **Users** tab that displays a list of users belonging to the group. The **Add** button in this users finder displays a third-level finder that allows you to select more objects to add to the list.



To create the Third-Level finder, follow these steps:

1. [Create a new user interface](#), which extends `ui_common.basePrimaryFinder`. Set the **Select input** and **Select output** objects as that of the old finder.
2. Provide a search form for the finder as in [create a search form](#) (optional).
3. Provide a result form for the finder as in [create a result form](#).

## Provide Select Functionality

The **Select** menu is shown when the finder UI is loaded. To achieve this functionality override **onSelect** method from **Methods** list. By default, there is one parameter passed in, representing the data object selected in the UI.

## Provide Result Filtering Functionality

The result filtering functionality can be achieved by overriding **compareKeys** method in the **Methods** tab and implementing with required filtering criteria, if different from **cwStructId** comparison. Implement **cwOnFinderSel** method and apply filter on search result. For example:

```
return this.controllerOwner.filterResult(result);
```

## Second-Level Finders

Second-level finders are finder user interfaces that are displayed within an object's detail component. For example, the *Group detail* form in *cwa\_security* contains a **Users** tab that displays a list of users belonging to the group. This form is a second-level finder. It differs from regular finders because it has logic to add or remove data objects to or from the list through the **Add** button.

The screenshot shows a 'Select User(s)' dialog box. At the top, there are tabs for 'User Profile Administrators', 'Users' (which is selected), and 'Privileges'. Below the tabs are three buttons: 'Add' (green plus icon), 'Remove' (red minus icon), and 'Export' (green document icon). The main area is a table with columns: 'User ID', 'Active', and 'Manager'. The table contains five rows of data:

User ID	Active	Manager
upadmin	●	
testuser	●	
arianepearlcarurucan	●	
b	●	

At the bottom left is a 'Rows per Page' dropdown set to 10. At the bottom right are navigation arrows and the text '1 - 4 of 4'.

To create the Second-Level finder, follow these steps:

1. [Create a new user interface](#), which extends **ui\_common.baseSecondaryFinder**. Set the **Select input** and **Select output** objects.
2. Provide a search form for the finder as in [create a search form](#) (optional).
3. Provide a result form for the finder as in [create a result form](#).

By default, the **Remove** menu is hidden, if the finder UI is read only. If **Edit** menu is selected, then the user interface is editable and the **Remove** menu is visible. Override the **onRemove** method and implement. By default, there is one parameter passed in, representing the data object selected in the UI.

### Provide the Add Finder Form

By default, the **Add** menu is hidden, if the finder UI is read only. Selecting the **Edit** menu makes the user interface editable and makes the **Add** menu visible. To provide the add finder form to use, when you click on the **Add** icon; override the **getAddFinderForm** method by returning the full name of the finder UI. Override the **getAddFinderForm** method and return the full name of the third-level finder form, by either publishing an event or just returning the full name.

### Popup Menus

The add finder can appear as a popup menu. The implementation can be done in the *addAction* method in *baseSecondaryFinder*. To create the pop up for additional menus, follow these steps:

1. Create a new Popup [user action](#) and specify the dialog object to **ui\_common.dialogExtend**.
2. In the script body, return the UI object to be displayed as a dialog.

### Providing the Search Criteria

Set up for search criteria can be done in two ways:

1. By using a search model. Navigate to the **Variables** tab and create a new variable, for example **searchModel**. Use this model in the select method.
2. Override and implement the **setsearchModel** method.

### Detailed View

If detail view is enabled:

- Override and implement **getDetailForm** and **getDetailModel** methods as in [providing the detail form](#).
- Ensure that the **isShowDetailEnabled** variable is set to true when using the [second-level finder](#).

### Read Only View

Once the second-level finder is initialized, you can set the user interface to read-only until edit menu is selected as follows:

```
showEditMenu = true;  
setReadOnly(true);
```

## Use the Second-Level Finder

In the calling UI, the following are the variables to be set or methods to be called after initializing the second-level finder:

Method or Variable	Details
<b>isShowDetailEnabled</b>	Enables or disables result items on double click
<b>isIconVisible</b>	Sets the visibility of result detail icon
<b>isGradientStyle</b>	False, if this variable is an embedded finder
<b>searchModel</b>	If <i>setsearchModel</i> method is set

## Menu Style

A menu can be created with a hierarchy of menu elements (see [menu elements](#) for more details). The general styling for menus is to have all menu icons as part of the styling. For example, most finder object menus have the following styling, which can be used for custom finder menus.

```
{  
background-repeat:no-repeat;  
background-position:left center;  
vertical-align:center;  
text-indent:20px;  
padding-right: 15px;  
font-family:Arial, Helvetica, sans-serif; font-size:11px;  
}
```

For every menu requiring an icon, the following styling is also added:

```
{  
Icon: '../cwt/images/16/document_plain_new.png';/*your location of the image may differ*/  
}
```

Custom menus for data objects (components) follow this styling:

```
{  
height: 22px;  
margin-top: 0px;  
margin-right: 0px;  
margin-bottom: 0px;  
margin-left: 0px;  
padding-top: 8px;  
padding-right: 5px;  
padding-bottom: 4px;  
background-position: 2px center;  
width: 63px;  
font-family:Arial, Helvetica, sans-serif;  
font-weight: bold;  
font-size: 11px;  
background-repeat:no-repeat;  
background-color: #f4f4f4;  
border: 1px solid #d9d9d9;  
color: #444444;  
vertical-align:center;  
text-indent:17px;  
text-align:center;  
}
```

All menu styles also have entries for the style suffixes: *Over*, *Down*, and *Disabled*. For example,  
.cwSaveMenu,.cwSaveMenuOver,.cwSaveMenuDown,.cwSaveMenuDisabled.

## Submenus

All menus with children specify **ShowRollOver** property to true, in order to get a mouse-over effect. All application submenus follow the base styling **cwAppSpecificSubMenu**. If there is no customization for your application menu, set the **Style** property as follows:

```
.cwAppSpecificSubMenuOver,.cwAppSpecificSubMenuDown{  
    background-color: #000000;  
}  
  
.cwAppSpecificSubMenu,.cwAppSpecificSubMenuOver,.cwAppSpecificSubMenuDown{  
    font-weight:bold;  
    font-size:9px;  
    color:#ffffff;  
    cursor: pointer;  
    border-bottom: 1px solid #98d0f2;  
}  
.cwAppSpecificSubMenu,.cwAppSpecificSubMenuDisabled{
```

```
        background-color: #53b0ea;  
    }
```

All action submenus follow the styling *cwActionSubMenu*, such as the views dropdown of the baseFinder. Like the application submenus, the action submenus have two separate styles for normal and for mouse-over. The following is the normal style:

```
.cwActionSubMenu,.cwActionSubMenuDisabled{  
    background-color: #53b0ea;  
}  
  
.cwActionSubMenu,.cwActionSubMenuOver,.cwActionSubMenuDown,.cwActionSubMenuDisabled{  
    font-weight:bold;  
    font-size:9px;  
    color:#ffffff;  
    cursor: pointer;  
}  
  
.cwActionSubMenuOver,.cwActionSubMenuDown{  
    background-color: #000000;  
}
```

Most submenus do not require a disabled styling. But in a situation where a disabled styling is required, follow the base style for *cwActionSubMenuDisabled*. The following is the mouse-over style:

```
.cwActionSubMenuDisabled,.cwViewItemMenuDisabled,.cwModifyViewMenuDisabled,.cwDeleteViewMenuDisabled{  
    color:#c0c0c0;  
    font-size:9px;  
}
```

If your submenu does not require customization, set the **Style** property to *cwActionSubMenu*. If a submenu requires more specific styling such as icons, then create a separate style for the application. For example:

```
.cwExcelActionMenu{  
    icon:'/cwf/excel.gif';  
}
```

*.cwExcelActionMenu* needs to be added to the list of style names where *.cwActionSubMenu* is used in the normal style.

## Permissions

Properties such as **visible** and **enabled** use boolean variables that belong to the user interface. If boolean variables are not used, privileges are needed. Permission methods are used to provide privileges.

# Messages

There are two types of messages related to the user Interface:

- [Status Messages](#)
- [Confirm messages](#)

## Status Messages

Status messages are displayed after actions are triggered, to inform the success or failure (for example, in finders, delete and search actions and in components, the save action). The common library has predefined status message codes. These codes can be reused or augmented with new messages. Status messages are displayed on either the application or dialog UI. When the status message is longer than the status textbox, a menu beside the status is available to trigger a new window with the full status message.

## Methods

The following methods are used to set or clear the status of a message:

- **setModelStatus** (Object result, String msg, Boolean style, String label) – This method is used to set the status of a component or finder with either a confirmation or error message. The result parameter is the result of an action performed (such as, saved model, or a fault). The msg parameter is the text code. The style parameter, sets if the message is either a confirmation or an error. The label parameter act as input parameter to translate the text code.
- **resetModelStatus()** – This method is used to clear the status of either a component or finder.

## Global Scripts

The following global scripts are used with messages:

- **ui\_common.resetStatusMessage**(Object obj) – This script is used to clear the status message. It is called from application UI menu items to clear status on page change.
- **ui\_common.setModelStatus**(Object input, Object result, String msg, Boolean style, String label) – This script is called from *setModelStatus* method either to set the application UI or dialog UI status message.

## Variables

The following variables are used in **ui\_common.baseApplication** and **ui\_common.dialogExtend** to display the status messages:

Variable	Description
<b>modelStatus</b>	Full model status message
<b>modelStatusTrim</b>	Trimmed model status. Used for display in the textbox
<b>showStatus</b>	Flag used to show status message
<b>isConfirmStyle</b>	Flag used for model status type: <ul style="list-style-type: none"><li>• True = Confirmation</li><li>• False = Error</li></ul>
<b>isModelStatusMenuEnabled</b>	Flag used for model status menu, if enabled or disabled
<b>isModelStatusMenuVisible</b>	Flag used for model status menu visibility

## Styles

The following style parameter are used to sets if the message is either a confirmation or an error:

Style	Description
<b>cwConfirmMessage</b>	This is the style used when the message is a confirmation.
<b>cwErrorMessage</b>	This is the style used when the message is an error.

## Status Message Codes

Here are examples of status message texts with codes in the common library. To customize, add messages codes to your own library.

ID	Message
COMMON_UI_DEL_002	Delete {0} completed successfully
COMMON_UI_SAVE_001	Saving failed
COMMON_UI_SAVE_002	{0} has been saved successfully

These codes are listed in Velocity Studio. Go to the left navigation panel and select the **Resources** tab. Select the **Application Resources of ui\_common**. A list of messages with related id displays.

Id	Translated Message	Message
COMMON_UI_DEL...	Delete {0} completed successfully	Delete {0} completed successfully
COMMON_UI_CON...	Do you want to set user available?	Do you want to set user unavailable?
COMMON_UI_CON...	Do you want to set user unavailable?	Do you want to set user available?
COMMON_UI_PAS...	Change Password	Change Password
COMMON_UI_PAS...	Password change failed	Password change failed
COMMON_UI_PAS...	Password changed successfully	Password changed successfully
COMMON_UI_PAS...	Passwords do not match	Passwords do not match
COMMON_UI_ACTI...	Application: {0}	Application: {0}
COMMON_UI_PRE...	Preferences	Preferences
COMMON_UI_PRE...	There are currently no preferences to personalize. Please...	No preferences defined for group {0}
COMMON_UI_PLAT...	No preferences defined for group {0}	Platform: {0}
COMMON_UI_LOGI...	Welcome	Welcome
COMMON_UI_LOGI...	Login to {0}	Login to {0}
COMMON_UI_NOP...	User {0} does not have permission to view {1}	User {0} does not have permission to view {1}
COMMON_UI_UNSA...	Unsaved changes found. Continue and lose unsaved ch...	Unsaved changes found. Continue and lose unsaved ch...
COMMON_UI_SEL...	Do you want to save data before changing node?	Do you want to save data before changing node?
COMMON_UI_ADD...	Add {0}(s) failed	Add {0}(s) failed
COMMON_UI_ADD...	Add {0}(s) completed successfully	Add {0}(s) completed successfully
COMMON_UI_REM...	Remove {0}(s) failed	Remove {0}(s) failed
COMMON_UI_REM...	Remove {0}(s) completed successfully	Remove {0}(s) completed successfully
COMMON_UI_LOGIN	Logged in as: {0}	Logged in as: {0}
COMMON_UI_NOT...	User is currently not logged in	User is currently not logged in
COMMON_UI_SAVE...	Saving failed for {0}	Saving failed for {0}

## Using Status Messages

Error messages can be customized by:

- Custom Message Text
- Success or Error Flag
- Custom Label

A status message can be reset from within the UI or from any other UI. To reset status messages from any UI, use the following script:

```
ui_common.resetStatusMessage(target UI)
```

To reset status messages from within the UI, use the following script:

```
resetModelStatus()
```

## Status Messages for Finders

To use status messages for finders, consider these points:

- For default error message, only one parameter needs to be passed to *setModelStatus method*, the fault. If the first parameter is a fault, the status message is set to the fault message with the *cwErrorMessage* style.
- To set a custom message for finders, the first parameter needs to be **null**. The second parameter is the message code.
- The third parameter sets the message style to either a confirmation or an error.
- The fourth parameter is optional and is used if a parameter is required to set in the error text. For example:

```
this.setModelStatus(null, "COMMON_UI_DEL_002", true, selected[0].metadataTypeLabel);
```

## Status Messages for Components

To simplify status messaging for components, when using the save action, only one parameter needs to be passed to *setModelStatus method* for either error or success status messages.

Message	Description
Error	If the first parameter is fault, the status message is set to the default message with the <code>cwErrorMessage</code> style.
Success	If the first parameter is not fault, the model is updated with the parameter and the standard component save success message is displayed with the <code>cwConfirmMessage</code> style.

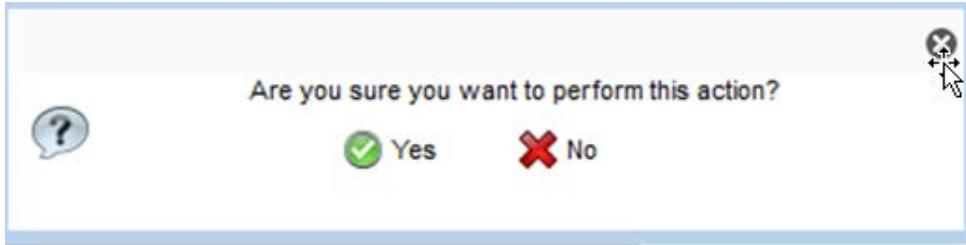
For example, to set the status message from a save action, use these steps for the `onSaveModel` method:

1. If calling an API, capture the return value and use it to set the model status (for example, `this.setModelStatus(result)`). The status message style is set automatically.
2. If performing validation check, defaults can be set in the model status in the same way (for example, `this.setModelStatus(fault)`). The status message style is `cwErrorMessage`.
3. To set a custom message, provide a custom success message code in the parameters. If the first parameter is a fault the default error message is used (for example, `this.setModelStatus(result, "LOC_UI_SAVE_ADDR_002")`).

## Confirm Message

Confirm messages are used to confirm a user action (for example, delete). To create a confirm message, follow these steps:

1. Create a new script method that shows the confirm message (for example, `return "UU0167:YN"` ; ). The script in the example returns a *Are you sure you want to perform this action?* and yes and no button (as in the image).
2. Create the user action that needs to specify as the click action of the menu or button.
3. Set the confirmation script to the one created in Step 1.
4. Set the customized confirm dialog object to `ui_common.confirmDialog`. The result looks like this image:



## Dialogs

Dialogs are used to display information related to the current user interface, without leaving the page. To use the same look and feel of **ui\_common** dialogs such as reference finders, follow these steps:

1. [Create the user action](#) that is used to display the object.
2. Add a new parameter called **dialog** of type **com.conceptwave.system.Dialog**. It ensures that the dialog object in the script is accessible.
3. Set **Show in popup** to **true**.
4. Set **Dialog Object** property to **ui\_common.dialogExtend**.

The screenshot shows the 'Dialog' configuration window in the Conceptwave UI Designer. The 'Name' field is set to 'mydialog'. Under 'Parameters', there is a table with one row named 'Dialog' of type 'com.conceptwave.system:Dialog'. The 'Script' field contains the following JavaScript code:

```
function mydialog(Dialog) {
```

The 'Return' field is set to 'api\_common.data.message'. The 'Object' field is also set to 'api\_common.data.message'. The 'Form' and 'Confirmation' fields are both set to '<NONE>'. Below these fields are several configuration options: 'Validate' (unchecked), 'Allow invalid object' (unchecked), 'Autosave' (unchecked), and 'Show in popup' (checked). There are also input fields for 'Dialog Width' (0), 'Dialog Height' (0), 'Dialog X Coordinate' (-1), and 'Dialog Y Coordinate' (-1). The 'Dialog Form' field is empty. The 'Dialog Object' field is set to 'ui\_common.dialogExtend (Dialog)'. On the right side of the window, there are various buttons for managing the dialog's properties and appearance.

5. In the script, create or initialize the user interface and return it.
6. If the user interface that is displayed requires a different width or height than being used. It can be done by providing values like the following script:

```
dialog.dialogWidth = 500;  
dialog.dialogHeight = 700;
```

### Change the Grid Layout Style

The default grid layout for the detail form or classic search is six columns, set to fill the width of the page. If a different style is required then that can be done either by using a predefined narrow grid layout style or custom grid layout style.

The default grid layout style is defined as follows:

```
.cwFormDetailGrid{  
    vertical-align: middle;  
    colWidths: 156px,156px,156px,156px,156px,156px,25px; /*25px to be removed*/
```

```
}
```

### Predefined Narrow Grid Layout Style

Once a finder UI or baseComponent UI is created, use the `setGridStyleWide` method to use the narrow grid layout style. The narrow grid layout style is defined as follows:

```
.cwFormDetailGridNarrow{  
    vertical-align: middle;  
    colWidths: 122px,122px,122px,122px,122px,122px,25px; /*25px to be removed*/  
}
```

### Define a Custom Grid Layout Style

To define a custom grid layout style, override the `returnGridLayoutStyle` method in implementation of `baseComponent` and `baseFinder`. If external forms are used either for `baseComponent` detail form or `baseFinder` classic search form, then also override `returnGridLayoutStyle` method in the implementation of the `baseForm`.

## Glossary

---

The following list describes the terms often used in the documentation:

### **Application**

Application is a shorter form of application program. An application program is a program designed to perform a specific function directly for the user or, in some cases, for another application program. It is the use of a technology, system, or product. An application, or application program, is a software program that runs on your computer. For example, web browsers, e-mail programs, word processors are all applications. The word application is used because each program has a specific application for the use; they may have different file extensions but serves the same purpose.

### **User Interface**

A user interface (UI) is the means in which you can control a software application. UI works as a connection between a user and a computer program. An interface is a set of commands or menus. A good user interface provides a User-Friendly experience, allowing the user to interact with the software. An application can have different user interfaces. But they can share many of the same elements that results in providing a consistent user experience across multiple programs.



# Trademarks

Ericsson, the Ericsson logo and the Globemark are trademarks of Ericsson.

Ericsson is a recognized leader in delivering communications capabilities that enhance the human experience, ignite and power global commerce, and secure and protect the world's most critical information. Serving both service provider and enterprise Customers, Ericsson delivers innovative technology solutions encompassing end-to-end broadband, Voice over IP, multimedia services and applications, and wireless broadband designed to help people solve the world's greatest challenges. Ericsson does business in more than 150 countries. For more information, visit Ericsson on the Web at [www.Ericsson.com](http://www.Ericsson.com).

# Disclaimer

This document may contain statements about a number of possible benefits that Ericsson believes may be achieved by working with Ericsson. These might include such things as improved productivity, benefits to end users or cost savings. Obviously, these can only be estimates. Gains might be qualitative and hard to assess or dependent on factors beyond Ericsson's control. Any proposed savings are speculative and may not reflect actual value saved. Statements about future market or industry developments are also speculative.

Statements regarding performance, functionality, or capacity are based on standard operating assumptions, do not constitute warranties as to fitness for a particular purpose, and are subject to change without notice.

This document contains Ericsson's proprietary, confidential information and may not be transmitted, reproduced, disclosed, or used otherwise in whole or in part without the expressed written authorization of Ericsson.