

**Developing Solutions
Student Guide
Version 11**

Copyright © 2015 by Aras Corporation. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.

Distribution of the work or derivative of the work in any standard (paper) book form for commercial purposes are prohibited unless prior permission is obtained from the copyright holder.

Aras Innovator, Aras, and the Aras Corp "A" logo are registered trademarks of Aras Corporation in the United States and other countries.

All other trademarks referenced herein are the property of their respective owners. Notice of Liability

The information contained in this document is distributed on an "As Is" basis, without warranty of any kind, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement. Aras shall have no liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this document or by the software or hardware products described herein.

MARCH 2015

Table of Contents

Table of Contents	3
Introduction	5
Unit 1 Overview	11
Unit 2 Reviewing AML	23
Unit 3 Creating a Method	61
Unit 4 Debugging Client/Server Methods	85
Unit 5 Creating Custom Actions.....	99
Unit 6 Exploring the Innovator Class	115
Unit 7 Exploring the Item Class.....	131
Unit 8 Creating Server Event Methods.....	159
Unit 9 Passing Event Parameters.....	177
Unit 10 Exploring Aras Object Functions.....	197
Unit 11 Creating Client Event Methods.....	209
Unit 12 Using Federated Items/Properties.....	235
Unit 13 Integrating a .NET Application	253
Unit 14 Using an External Client.....	265
Unit 15 Importing Data with Batch Loader.....	277
Unit 16 Configuring the Scheduler Service.....	295
Appendix A: Custom Search Result Sets.....	305
Appendix B: Creating Form Dialogs.....	313
Appendix C: Customizing a Form	327
Appendix D: Creating a Configurable Grid.....	335
Appendix E: Tracking Login and Logout	359
Appendix F: Working with Relationship Grids	365
Lab Solutions	383
Index	407

This page intentionally left blank.



Introduction

Overview: In this course, you will learn how to customize and extend the Aras Innovator environment and include business logic in a solution. You will also learn how to establish integrations with other systems using the Aras Innovator Object Model (IOM) API and gain a working knowledge of AML (Adaptive Markup Language).

- Goals:**
- ✓ Use AML to add, edit and remove Items from the database.
 - ✓ Create Client and Server Methods to modify, extend or replace standard system behaviors.
 - ✓ Debug a Client or Server method using Microsoft Visual Studio
 - ✓ Associate Client and Server methods with system events and user actions
 - ✓ Explore and use the IOM methods available in the Innovator and Item classes
 - ✓ Use an alternate client with the Innovator Server
 - ✓ Integrate Aras Innovator into a Microsoft Windows or .NET web application

Aras Overview



Business	Next Generation Enterprise Software Solutions
Company	Global Organization, PLM Executives & Technologists
Solutions	Aras Innovator® Suite
Out-of-the-Box	Comprehensive Enterprise PLM <ul style="list-style-type: none">▶ NPDI New Product Development & Introduction▶ CMII Configuration & Change Management▶ APQP Advanced Product Quality Planning▶ PDM Product Data Management
Advantage	Highly Scalable, Extensible, Upgradable
Innovation	Advanced Model-based SOA Technology

Copyright © 2014 Aras All Rights Reserved.

Aras takes a very different approach to PLM. We combined enterprise open source solutions with advanced model-based SOA technology to deliver a highly scalable, flexible and secure PLM solution suite. Our extensive out-of-the-box functionality and modern Web architecture enable you to deploy quickly and continuously enhance your PLM environment in a fraction of the time required by conventional enterprise PLM / PDM systems and at a Total Cost of Ownership far below that of any leading competitor.

Aras Customers



Copyright © 2014 Aras All Rights Reserved.

Aras enterprise open source solutions are for performance driven companies of all sizes from Fortune 500 enterprises to midsized businesses looking to get an edge on the competition. Companies achieve better customer alignment, greater profit margins, and shorter development cycles with the Aras Innovator enterprise open source solutions.



Course Goals

- When you have completed this course you should be able to:
 - Query, add, delete and modify Items using AML
 - Use the IOM to create and debug client and server methods to replace or extend solution behaviors
 - Work with federated properties
 - Use an alternate client
 - Integrate a .NET application
 - Schedule Method execution
 - Import data using the Batch Loader

Topics



- Overview
- Reviewing AML
- Creating a Method
- Debugging Client and Server Methods
- Creating Custom Actions
- Reviewing Innovator Class methods
- Reviewing Item Class methods
- Creating Server Methods with the IOM
- Creating Client Methods with the IOM



Topics

- Passing Event Parameters
- Using Federated Items and Properties
- Integrating a .NET Application
- Using an Alternate Client
- Using the Scheduler Service
- Using the Batch Loader

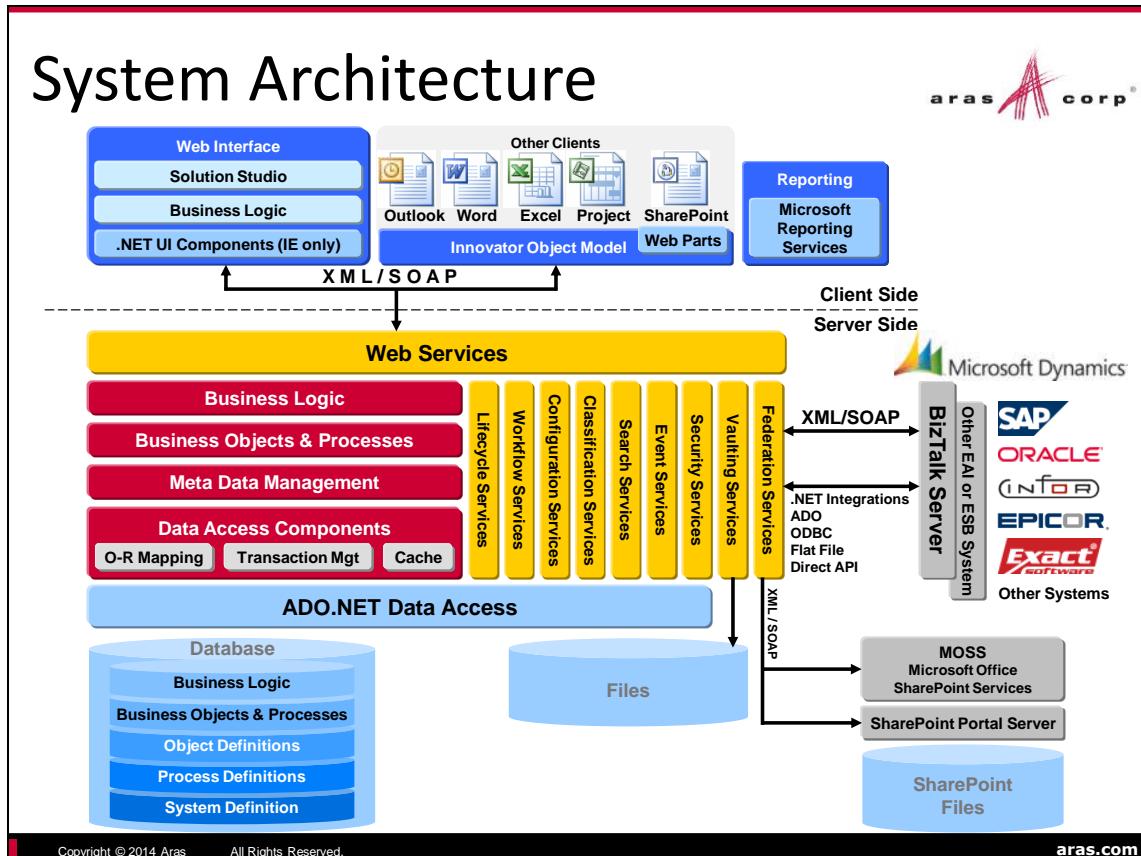


Unit 1 Overview

Overview: In this unit you will learn about the Aras Innovator Architecture and how you can programmatically interact with the Innovator Server. You will also review the Aras Innovator Methodology which describes the basic rules you should follow when developing solutions.

Objectives:

- ✓ Reviewing Aras Innovator Architecture
- ✓ Interacting with Aras Innovator
- ✓ Defining the Aras Innovator Methodology



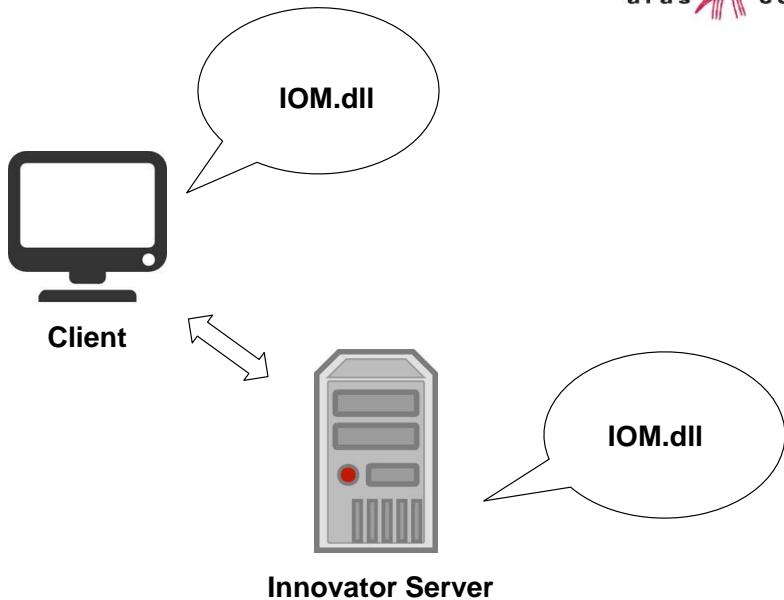
The Client uses a set of .browser neutral controls sharing a single XML-SOAP communication port to the server. (Currently, the Internet Explorer browser also utilizes several .NET controls.)

The Aras Innovator client is a “thin” interface - requiring the user access the Innovator Server using either MS Internet Explorer or Mozilla Firefox.

You can also create your own client or use an alternative client which you learn about later in this course.

Because Aras Innovator uses a SOA Architecture it can easily be incorporated into an enterprise solution with other providers.

Using the Innovator Object Model



Copyright © 2014 Aras All Rights Reserved.

aras.com

You will be interacting with the Innovator Object Model (IOM) which is a collection of classes and methods that can generate, send and respond to requests made on the server.

The IOM can be accessed from both the client (using JavaScript) as well as the server (using C# or VB.NET) and uses the same method calls (with a few exceptions). This makes it easy to learn the IOM API since the calls are similar regardless of location.

Interacting with Aras Innovator



- **Using Internal Innovator Methods**
 - Allow for custom business logic
 - Can run on the client or server
 - Can react to system events
 - Can modify, extend or replace existing behavior
- **Using External Programs**
 - .NET integration in a Microsoft Visual Studio project
 - .COM integration in a Microsoft Windows application
 - Using SOAP requests with the Microsoft XMLHTTP object

Copyright © 2014 Aras All Rights Reserved.

aras.com

You have several ways you can interact with the Innovator Server including:

- Creating Methods in Aras Innovator
- Invoking requests and responses from a .NET application using the IOM
- Using a COM integration in a MS Windows application (like MS Word or Excel) that supports VBA macros
- Creating SOAP requests with the MS XMLHTTP object and sending them directly to the server.

In this course, you will learn how to use all the various approaches available.

Defining the Aras Innovator Methodology



- Everything is an Item
- AML language describes Items
- AML messages are passed and returned from the Server
- Innovator Object Model (IOM) can be used to build AML messages
- Innovator Methods:
 - Define custom business logic
 - Modify, extend or replace Item behavior

The AML is the language that drives the Aras Innovator Server.

- Every object is defined as an Item.
- Every Item is described using the AML language
- The Aras Innovator Server is a message based system in that it accepts AML scripts as messages and returns AML messages. AML document, AML script and AML message all mean the same thing.
- The IOM is the Object API used to build and apply AML messages.
- Methods implement business logic using the IOM API.
- Methods extend the Item Class when used as an "Item Action". An Action is a relationship between a Method and an Item which is configured on the ItemType. This simulates Object Oriented programming, where the ItemType is the Class and "Item Action" relationships to Methods are the Class methods.
- Methods are also "generic" arbitrary business logic that can be called like a sub routine from other Methods using the IOM Innovator.applyMethod(...) method.

Defining the Aras Innovator Methodology



- Methods follow an Item Factory design pattern
- Client and Server events operate on a Context Item
- The Context Item should only be altered before it is processed by the Server
- The Context Item is referred to using the keyword *this* (C#) or *Me* (VB.NET) in a Method
- Use the appropriate attributes of AML when performing queries to optimize performance
- Actions define a relationship between the UI and an event

- Methods follow the Item Factories design pattern; they should return a new Item and not side effect the context Item.
- Server Events are the exception because the purpose is to intercept and operate on the AML before the server parses it, and before the AML is returned to the client after the server parses it. So you do modify the context Item and return nothing.
- Implement changes/edits to the context Item in the OnBefore Event by altering the AML before the server parses it. Refrain from attempting to update the context Item after the server has already operated on it in the OnAfter Event. Use the OnAfter Events to update/include federated data in the response AML.
- Use the select, page, and pagesize attributes for the AML queries to optimize the performance for the request.
- Use the generic IOM methods to construct the AML queries rather than convenience methods like getItemBy_, getRelationships(), or use the levels attribute because the convenience methods typically return far more data than is required imposing a performance hit.
- The context Item is the keyword *this* Object for JavaScript, C# and the *Me* Object for VB.Net.
- Attributes are used to pass control switches to the Method. You can invent your own and they can be a simple way to pass additional meta-data to a Method.

Defining the Aras Innovator Methodology



- Every client window has access to the "aras" object
 - 200+ additional JavaScript methods (in addition to IOM)
 - JavaScript file aras_object.js located in ..\Innovator\Client\javascript folder

In addition to the IOM API an Innovator Client method has access to many additional JavaScript methods that are associated with an object named “aras”. These methods have been created to interact with the Innovator Server or provide convenience utilities while working on the client.

You will learn about several of these methods in this course.

Comparing Aras Innovator Terms



Aras	Object Oriented	Business
ItemType	Class	Object Template (<i>Work Order</i>)
Item	Object Instance	Single Occurrence (WO-000331)
Property	Property	Attribute (<i>Priority</i>)
Method	Method / Function	Behavior (<i>Promote</i>)

Copyright © 2014 Aras

All Rights Reserved.

aras.com

An Item in Innovator is like an “Object” in Object Oriented terminology. It is used to represent real world ‘things’.

The nature and behavior of Objects is defined in a blueprint or template, referred to as a ‘Class’ in Object Oriented terminology. This is known as the ItemType in Innovator.

Items are related to Properties, which store data with the Item similar to properties of an Object in OO terms.

ItemTypes have “Item Action” and “Server Event” relationships, which describe their behavior - similar to methods defined for a Class.



Summary

In this unit, you learned about the Aras Innovator Architecture and different ways you can interact with the Innovator Server. You also learned the basic rules for creating new programs in the environment.

- ✓ Describe the Aras Innovator Architecture
- ✓ Explain the different ways to interact with the Innovator Server
- ✓ Describe the Aras Innovator Methodology



Lab Exercise

Goal:

Be able to describe the class business case scenario and data model to be used in this course.

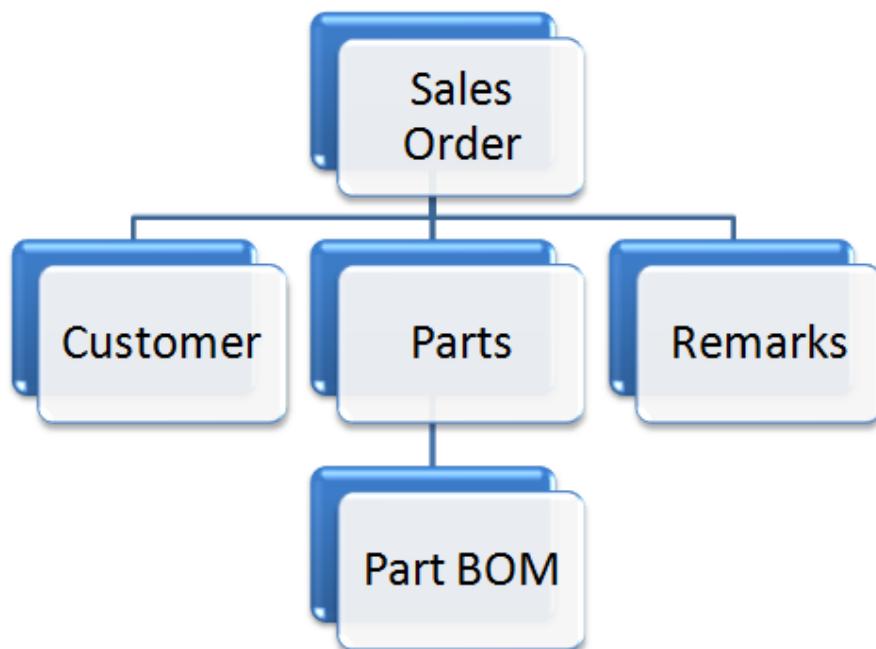
Description:

This course continues with the business items you were introduced to in the *Configuring Solutions* course.

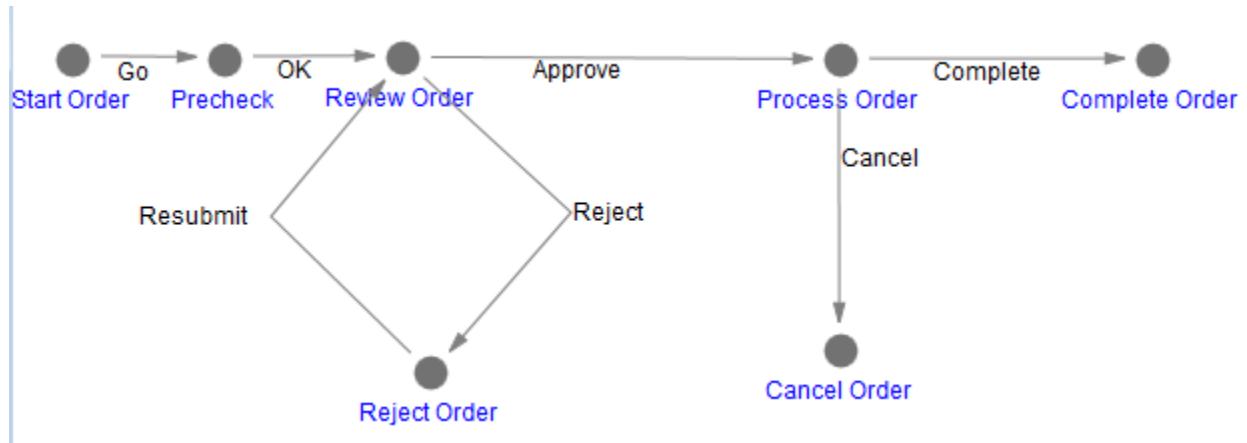
These include:

- Design Requests
- Change Requests
- Sales Orders
- Customers

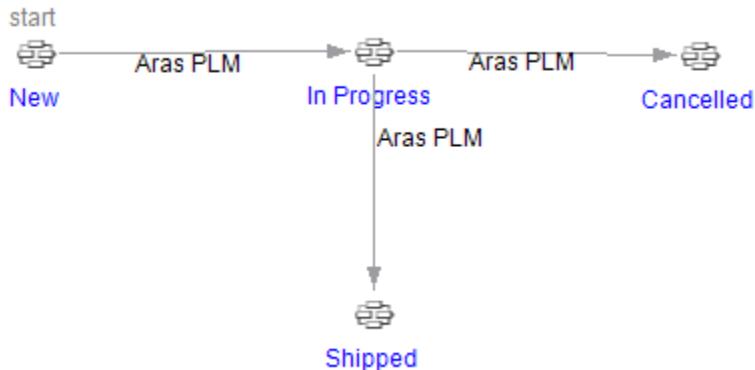
In addition, the Part ItemType is now used to demonstrate Part to Part relationships in the following hierarchy:



Each Sales Order is associated with the following Workflow:



And uses the following lifecycle:



You will provide custom business logic to the solution to validate consistency of data, implement specific business rules and provide data access to external client applications.

A small set of sample data has been provided to that you can begin to access and modify business items. Do not change any of the sample data items as they will be used in subsequent labs in the course.

This page intentionally left blank.



Unit 2 Reviewing AML

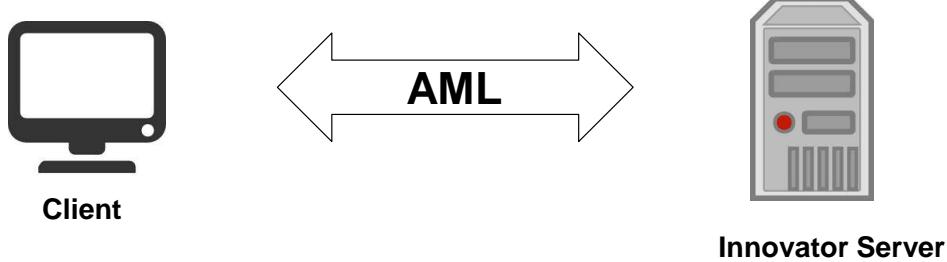
Overview: In this unit, you will learn how to use Adaptive Markup Language (AML) to create custom search queries as well as how to add and modify Items in the Innovator database. AML is the language that drives Aras Innovator and all communication between client and server is done using the AML dialect. Sometimes termed the “secret sauce” of Aras Innovator, understanding this language will also help you to work effectively with the IOM classes and methods.

- Objectives:**
- ✓ Defining the AML Syntax
 - ✓ Exploring AML Tags
 - ✓ Reviewing AML Structure
 - ✓ Building AML Queries
 - ✓ Testing with AML Studio
 - ✓ Reviewing Built In Actions
 - ✓ Adding an Item with AML
 - ✓ Editing an Item with AML
 - ✓ Deleting an Item with AML



Defining AML

- “Everything in Aras Innovator is an Item”
- Every Item is described using Adaptive Markup Language (AML)



Copyright © 2014 Aras All Rights Reserved.

aras.com

As you have already learned, everything in Aras Innovator is considered an Item.

The Adaptive Markup Language (AML) is an extension to XML that allows you to perform operation like search, edit, update and delete of Items without using the Aras Innovator user interface.

Any information that is exchanged between a client and the server is formatted in AML. So every transaction you have performed in this course has generated AML that was sent to the Innovator Server and then returned back to the client with results.

AML is used in integrations to other systems, for building advanced queries against the database as well as for batch loads or updates to the database.

In this unit, you will learn the basics of AML and be able to build queries, as well as edit and delete Items. More information about AML is also available in the *Aras Innovator’s Programmer’s Guide*.

AML Options



- AML Allows Item:
 - Retrieval – complex queries
 - Additions – multi-level items
 - Modifications
 - Purging (of a version)
 - Deletions
 - Locking/Unlocking

Copyright © 2014 Aras

All Rights Reserved.

aras.com

AML provides a set of built-in operations that allow you to get, update and delete Items in the database. You can also extend these actions with logic of your own.

Note

These built-in actions always obey the security permissions that you have defined for your solution.

Reviewing AML Structure



Copyright © 2014 Aras All Rights Reserved.

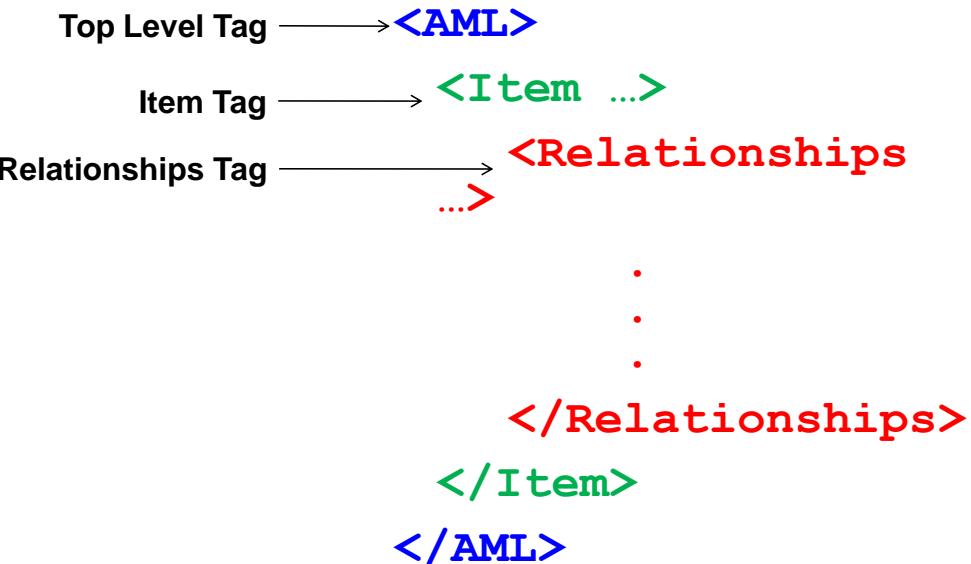
aras.com

As we have discussed earlier, each Item you create in the database can be related to another Item using a relationship Item.

You will recall, that a Relationship Item is created for each connection between two Items and contains the source and related ids of the connected Items. The properties `source_id` and `related_id` are provided for each relationship Item and store the ids of the source Item and the related Item.

This simple structure is important to remember as you begin to work with AML. Many of the AML tags used in the language use these terms as part of the dialect.

Exploring AML Tags



Copyright © 2014 Aras All Rights Reserved.

aras.com

<AML> Tag

This top level tag must always surround the encompassed Items. Like XML, a matching closing tag (</AML>) is also required to define the document.

<Item> Tag

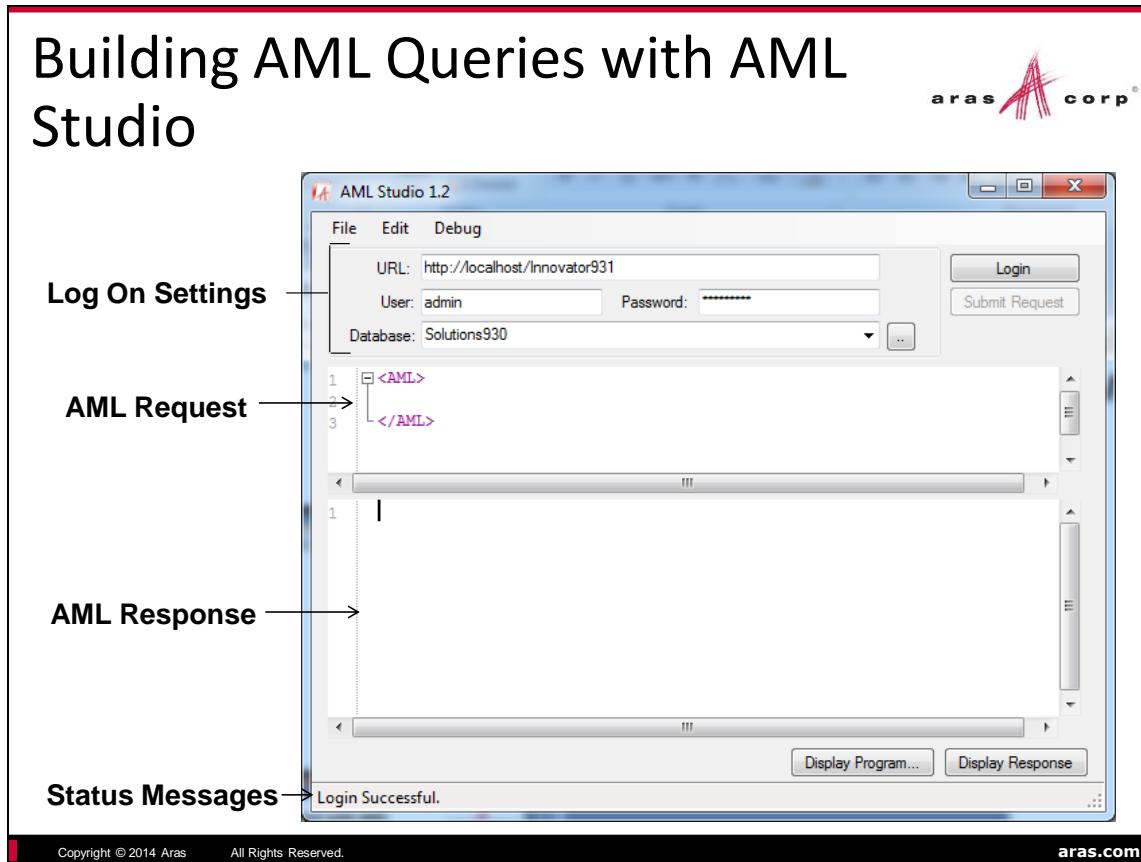
Contained within the AML tag are one or more Item tags that define each Item being expressed in the AML document. Item tags also contain nested property tags that you will learn about shortly.

<Relationships> Tag

Based on our previous discussions, remember that each Item can be related to another Item using a relationship. The Relationship tag defines the connection between the source Item (the parent tag of this relationship) and the related Item.

Note

AML is case-sensitive. Much like XML, each tag must have a matching close tag. Unlike XML, there is no XML schema defined for AML. Because Aras Innovator uses a self-describing data model, a formal named workspace is not used.



AML Studio

To assist with creation of AML queries a tool is available for download from Aras.com on the Community Projects page. This utility can be used to help you build queries faster and includes a “type ahead” feature as well as a syntax check that steps you through the language as you enter an AML request.

The AML Studio window is divided into 4 main areas:

Log On Settings

Provide the URL to your Aras Innovator server as well as the user id, password and desired database. After entering the URL in the field provided you can press the database lookup button to show you the currently configured database names in the drop list. Press the Login button to connect the server indicated. The message *Login Successful* will appear in the Status Message area.

AML Request

Enter a valid AML request in the field provided. Press Control + t (or choose Edit -> Tidy XML) from the menu to check the syntax for any errors and indent the request for easier viewing. Press the Submit Request button to send the message to the server.

You can resize the proportions between the request and response panes by dragging the scroll bar up or down on the screen. You can also increase the size of the request text font by pressing the Control key while moving the mouse wheel up or down.

AML Response

The AML response message from the server will appear in the AML Response pane when a message is submitted to the server. This will either contain the requested results or an error message.

To see the response message in an alternate viewer. Click the Display Program... button and choose an XML editor/viewer installed on your machine. When you click the Display Response button the response text will appear using your installed viewer.

Status Messages

The message area will indicate when you are connected to the server as well as the number of items found in a query when a search is successful.

Note

Aras Innovator also supplies a standard AML tool (NASH) when you install the Aras Innovator server. To access this tool provide the following URL in the web browser:

`http://[servername]/[InnovatorServerAlias]/Client/Scripts/Nash.aspx`

The NASH tool was created by and is used by developers creating solutions for Aras Innovator. The tool provides advanced features not available in the AML Studio. For more information see the *Aras Innovator Programmer's Guide*.



Primary Item Tag Attributes

- id – unique identifier for an Item
- type – ItemType name for the Item
- action – behavior (Method) to apply to the Item

```
<Item type="Part"
      id=[32 character identifier]
      action="get" />
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

Item Tag Attributes

Every Item tag can contain attributes which further describe the kind of Item being expressed.

type – is used to specify the ItemType of the Item. In the example above we are describing a Part Item.

id – is used to specify which Part Item in the database we are describing. One way to locate an Item in the database is to provide the unique identifier.

action – is used to indicate what action to take when this AML is sent to the server. Aras Innovator has a set of built-in actions you will learn about later in the unit. You can also specify a Method that you have written to perform some custom operation.

Note

You can obtain the id of an Item by clicking the right mouse button on an Item in search grid and choosing the Properties menu item. The properties dialog will display the id (which can be copied).



Item Property Tags



- Nested inside of an Item tag
- Property name is the tag name

```
<Item type="Part" action="get">  
  <item_number>8120-1378</item_number>  
</Item>
```



8120-1378

Copyright © 2014 Aras All Rights Reserved.

aras.com

Item Property Tags

Within each Item tag you can specify one or more property tags. The property tag name is the name of the property that has been created for this Item in the database. Like Item properties, property tags cannot contain spaces and are always lower case.

In the example above, the *item_number* property is being used with a Part Item.

The “get” action will attempt to retrieve the Part Item with an item number of 8120-1378.



Multiple Property Tags

- Implied "AND" condition

```
<Item type="Part" action="get">
    <generation>2</generation>
    <classification>Assembly</classification>
</Item>
```

Multiple Property Tags

When you supply more than one property tag in a get operation the implied operation is the AND operator. Only items that meet all conditions will be returned.

Using Logical And/Or Operators



```

<Item type="Part"  action="get"  >
    <or>
        <and>
            <cost>35</cost>
            <weight>10</weight>
            <make_buy>Make</make_buy>
        </and>
        <and>
            <cost>100</cost>
            <weight>15</weight>
            <make_buy>Buy</make_buy>
        </and>
    </or>
</Item>

```

Copyright © 2014 Aras

All Rights Reserved.

aras.com

Using And/Or Tags

AML supports the ability to provide logical AND, OR and NOT operators as part of a query.

You can group property tags into logical condition sets to locate Items that meet those conditions.

In this example, any Part Items that have a cost equal to 35 AND a weight equal to 10 AND a make/buy equal to Make – OR – any Part Items that have a cost equal to 100 AND a weight equal to 15 AND a make/buy equal to Buy will be returned.

Note

To supply a NOT condition wrap the enclosed tags with the `<not> </not>` tag.



Property Attributes

- Using Conditions:

```
<Item type="Part" action="get">
  <cost condition="gt">35</cost>
</Item>
```

	8121-0811 Cost 55
	8121-0868 Cost 70
	8121-1036 Cost 36

Condition Property Attribute

When retrieving Items, it is useful to be able to provide some conditional logic to indicate what Item (or Items) will be returned from the database. The condition attribute supports the following values:

Condition	Description	Example
eq	Property is equal to another value	<name condition="eq">Peter</name>
ne	Property is not equal to another value	<name condition="ne">Peter</name>
ge	Property is greater than or equal to another value	<cost condition="ge">200</cost>
le	Property is less than or equal to another value	<cost condition="le">200</cost>
gt	Property is greater than another value	<cost condition="gt">200</cost>
lt	Property is less than another value	<cost condition="lt">200</cost>
like not like	Include % or * wild card symbols	<name condition="like">Admin%</name>

Condition	Description	Example
between not between	Use <i>and</i> keyword for range	<cost condition="between">10 and 90</cost>
in	Value is in comma delimited set	<name condition="in"> 'Admin','Peter','Bob'</name>
is null is not null	Value is null or not null	<name condition="is null" /> <name condition="is not null" />

Searching by Date and Time

Date and Time is stored in the Aras Innovator database using a neutral format:

yyyy-mm-ddThh:mm:ss

To search for Items by date and time you can use the *condition* property attribute to find items before and after a specific date (and time).

Examples

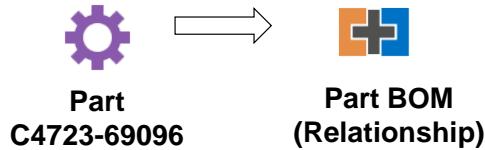
```
<created_on condition="ge">2013-02-01</created_on>
<modified_on condition = "lt">2012-03-01T12:00:00</modified_on>
<schedule_date condition="between">2013-03-01 and 2013-04-01</schedule_date>
```



Relationships Tag

- “Container” for Related Items
- Has no attributes
- Nested ItemType Refers to the Relationship Item

```
<Item type="Part" action="get">  
  <item_number>C4723-69096</item_number>  
  <Relationships>  
    <Item type="Part BOM" action="get">  
    </Item>  
  </Relationships>  
</Item>
```



Copyright © 2014 Aras

All Rights Reserved.

aras.com

Describing Relationships

Items can have relationships to other Items. The Relationships tag describes a relationship between the source Item and a related Item that has been defined with a RelationshipType. Remember that each relationship connection is defined by a Relationship Item in the database.

In the example above, we are attempting to get a Part Item with Item Number C4723-69096 that has a Relationship Item named Part BOM. If successful, the AML returned will describe the Part as well as the Relationship Item.

Related Id Tag



```

<Item type="Part" action="get">
  <item_number>C4723-69096</item_number>
  <Relationships>
    <Item type="Part BOM" action="get">
      <related_id>
        <Item type="Part" action ="get" >
        </Item>
      </related_id>
    </Item>
  </Relationships>
</Item>

```

Part
C4723-69096
 Part BOM
(Relationship)
 Parts...

Copyright © 2014 Aras All Rights Reserved.

aras.com

Describing Related Items

A Relationship Item contains a source_id and a related_id property which define the connection between the two Items. To describe the related Item you can use the `<related_id>` tag as shown above.

In this example, we are retrieving the same Part, getting the Relationship Item named Part BOM – and from that retrieving the related Part.



Searching on Related Items

①

```
<Item type="Part" action="get">
    <Relationships>          ②
        <Item type="Part BOM" action="get">
            <related_id>           ③
                <Item type="Part" action ="get" >
                    <item_number>
                        C4704-60117
                    </item_number>
                </Item>
            </related_id> ①
        </Item>
    </Relationships>
</Item>
```

① Part ② Part BOM (Relationship) ③ Part C4704-60117

Copyright © 2014 Aras

All Rights Reserved.

aras.com

Searching on Related Items

Using the related_id tag, you can describe the related Item and any properties that should be queried.

In this example, any Parts ① with a Part BOM ② relationship to a Part ③ with the item number of C4704-60117 will be returned.

Searching on Relationship Items



```
<Item type="Part BOM" action="get">
  <related_id>
    <Item type="Part" action="get">
      <item_number>C4704-60117</item_number>
    </Item>
  </related_id>
  <source_id>
    <Item type="Part" action="get">
    </Item>
  </source_id>
</Item>
```



Copyright © 2014 Aras All Rights Reserved.

aras.com

Searching on Source Items

Because relationships are also Items, you can use the AML "get" action to locate a relationship and the attached source and related Items.

In this example, a search is performed to find any Part BOM relationships with a related Part number C4704-60117. If any are found, the *source* Part Item is also returned. Reverse lookups are very helpful when working with Configurable Grids which will be discussed later in the course.

Searching on Item Type Properties

①

```
<Item type="Part" action="get" >
    <created_by_id>
        <Item type="User" action="get">
            ② <last_name>Admin</last_name>
        </Item>
    </created_by_id>
</Item>
```



①
Part
(Item)



②
User
(Item)

Name	Data Type	Data Source [...]
created_by_id	Item	User

Searching on Item Type Properties

Previously in this course, we discussed how you can create properties of data type Item that can be used to refer to another single Item in the system. The syntax above shows how you can use AML to describe this property and query on properties that belong to the connected Item.

In this example, the Part Items ① will be searched for a property named created_by_id that can contain an Item of type User②. The search will attempt to find a User with the *last_name* of Admin that is associated with this Part. If successful, the Part(s) will be returned.

Additional Item Attributes



- where – used instead of id attribute for searches
- select – choose the properties to return (get only)
- orderBy – sort the returned results
- page – page number for the result set
- pageSize – size of returned page
- maxRecords – size of returned result set
- levels- Item level “depth” to return
- version – allow versioning (update only)

Additional Item Attributes

Attribute	Data Type	Description
where	String	Used instead of the id attribute to specify the WHERE clause for the search criteria. Include the bracketed table name with the column name using dot notation: where="[user].first_name like 'Tom%'"
doGetItem	boolean	Perform “get” operation on item after initial action has been executed. Default = 1.
select	String	A comma delimited list of property names (column names) to return
orderBy	String	A comma delimited list of property names (column names) to order the results.
page	Integer	The page number for the results set.
pageSize	Integer	The page size for the results set.
maxRecords	Integer	This defines the absolute maximum Items to be searched in the database.

Attribute	Data Type	Description
levels	Integer	The Item configuration depth to be returned for items related to the current item. This should be used with caution due to the lack of specificity in the data fetched (all). Use nested Relationships to define queries for better performance.
serverEvents	Boolean	If 0 then disable the server events improving performance. Default is 1.
isCriteria	Boolean	If 0 then include the nested structure for the Item configuration in the response but don't use it as search criteria. Default is 1, which will use the nested structure in the request as search criteria.
related_expand	Boolean	If 0 then do not expand the related_id Property for the relationship Items to include the related Item. Another word just returns its ID.
language	String	A comma-delimited list of language codes, or "*" to return all languages. Multilingual property values will be returned (if present) for all specified languages.
version	Boolean	If 0 then don't version an Item on update. Default is 1, which is version the Item (if it's a versionable Item) on update.
queryType	String	Effectivity Search – possible values are "Latest", "Released" or "Effective".
queryDate	String	Date of effective search based on queryType. Format must be "yyyy-mm-ddThh:mm:ss"

Using Additional Attributes



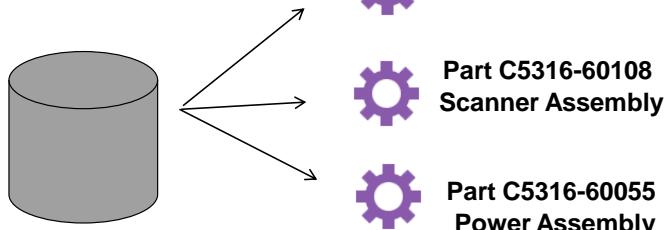
<AML>

```

<Item type="Part" action="get"
① select="item_number,name" ③          ④
② orderBy="item_number" page="1" pagesize="10">

    <is_released>0</is_released>
</Item>
</AML>

```



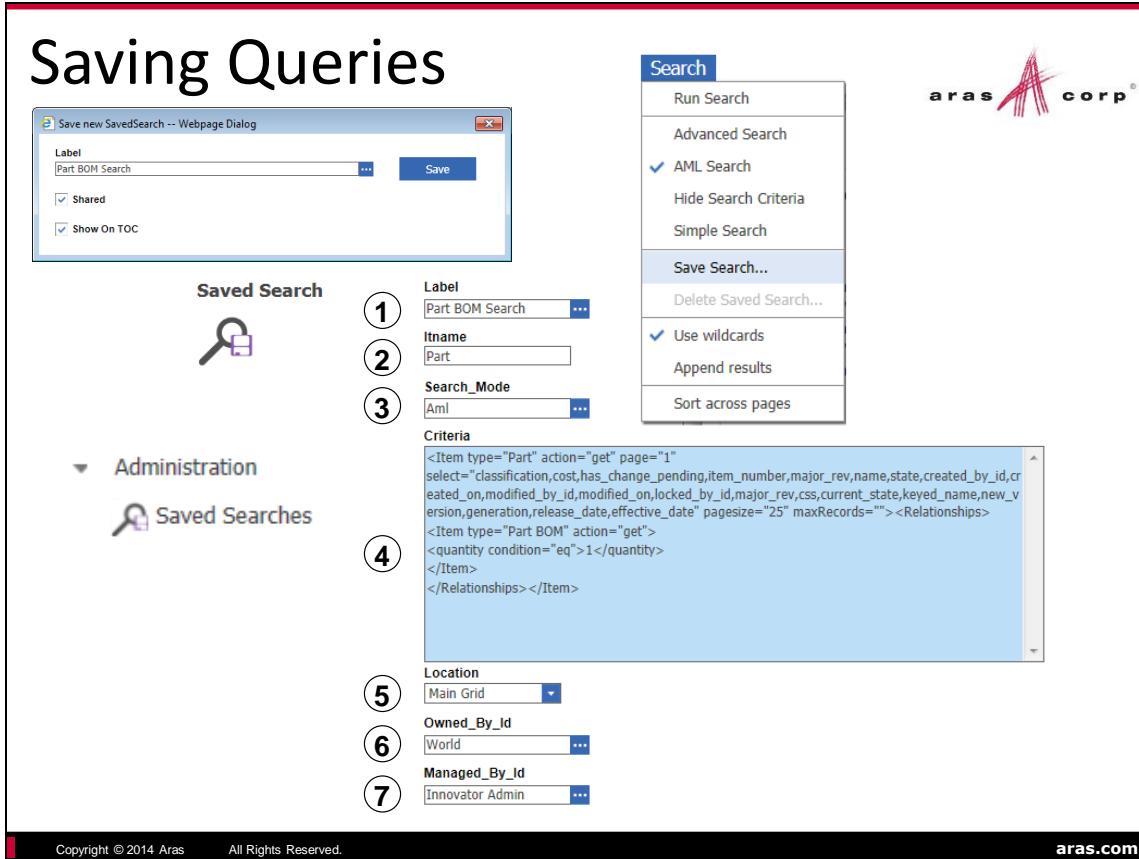
Copyright © 2014 Aras All Rights Reserved.

aras.com

Using Additional Item Attributes

You can reduce the amount of information returned from a query by using additional Item attributes.

In this example, the select attribute ① will retrieve the item_number and name values from the Part Items that have not been released and then order the results ② by the item_number . Only 1 page ③ with a maximum of 10 records ④ will be returned.



Saving Queries

AML Queries can be saved and reused by other users who can take advantage of complex queries only available in AML.

To Save a Query

1. Create the AML query using AML Studio and test to make sure results are correct.
2. Save the query from the Search -> Save Search... menu item.
3. Indicate if the query should be displayed on the TOC and/or shared with other users. You can control who sees the query (described below).

To Edit a Query

1. Locate the saved search by selecting Administration -> Saved Searches from the TOC.
2. Open the search for edit. The following fields are available to further configure the saved search:

① Label

Label that appears in the TOC and MyInnovator -> Search Center page.

② Itname

Itemtype of the items returned from the query.

③ Search Mode

The type of search to be selected in the client when the search is executed by a user. Choices are Simple, Advanced, AML or NO UI(Hidden). For this example AML is selected.

④ **Criteria**

The AML query request.

⑤ **Location**

The grid associated with the query. Choices are Main, Search Dialog or Relationships.

⑥ **Owned_by_id**

Members of this Identity will have access to the shared saved search.

⑦ **Managed_by_id**

Members of this Identity are responsible for any changes to the saved search configuration.



Reviewing Built In Actions

- get – retrieves an Item
- add – creates a new Item
- update – updates a locked Item
- purge – deletes an Item version
- delete – removes the Item (and all versions)
- edit – locks, updates and unlocks an Item
- create – adds the Item if it does not exist
- merge – edits the Item if it exists, otherwise add
- lock/unlock – locks or unlocks an Item

Item Tag Built In Actions

The following actions are built in to the system to perform different kinds of operations against the database. You can also create your own Methods and supply them as an action (described later in the course).

Built in Action Method	Description
add	Add the Item as an instance of an ItemType.
update	Updates the Item. The Item must be locked. If the Item is versionable and is being updated the first time since being locked then update versions the Item (new generation). If the attribute version="0" then versioning is disabled.
purge	Delete the version of the Item.
delete	Delete all versions of the Item. The purge and delete are the same for non-versionable Items.
get	Gets the Item(s) and its configuration based on the AML Item configuration used to query the database.

Built in Action Method	Description
GetItemConfig	<p>This will return the Item configuration as described by the standard AML query. The AML in and out are no different than the standard action="get".</p> <p>The GetItemConfig is optimized by limiting the logic done between the SQL call and the AML result. The performance improvement is gained by limiting the features typically available in the "get" action (no server events).</p>
GetItemRepeatConfig	<p>This will allow deep recursive queries and is useful in multi-Part BOM's with repeating relationships.</p>
edit	<p>This will lock, update, and unlock the Item.</p>
create	<p>This will act as a "get" if the Item exists, otherwise acts as an "add". Note that if an attempt is made to create an item that already exists with the same property values it will not create the item, just return the item(s) that match the existing criteria.</p>
merge	<p>This will act as an "edit" if the Item exists, otherwise acts as an "add".</p>
lock	<p>This will lock the Item and is the same as the Item.lockItem() method. Lock requires the id attribute.</p>
unlock	<p>This will unlock the Item and is the same as the Item.unlockItem() method. Unlock requires the id attribute.</p>

Adding New Items



```
<AML>
<Item type="Part" action="add">
  <item_number>2134-9099</item_number>
  <cost>12</cost>
  <name>Paper Sensor</name>
  <description>Front feed sensor</description>
  <make_buy>Buy</make_buy>
  <classification>Component</classification>
</Item>
</AML>
```



Part
(Item)

Adding New Items

The add action is used to add a new Item to the database (or to add new relationships to existing Items).

In this example, a new Part will be generated with the property values specified by the property tags above.

Adding Multiple Items



```
<AML>

  <Item type="Part"  action="add">
    <item_number>1121-9011</item_number>
    <name>Front Roller</name>
  </Item>

  <Item type="Part"  action="add">
    <item_number>1121-9012</item_number>
    <name>Rear Roller</name>
  </Item>

</AML>
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

Adding Multiple Items

A single AML request may contain multiple Item statements.



Editing a Single Item

- Supply the *id* or *where* attribute to act on an Item

```
<AML>
  <Item type="Part" action="edit"
        where="[part].item_number=
          '2134-9099'">
    <cost>22</cost>
  </Item>
</AML>
```

 Cost
22
2134-9099

Copyright © 2014 Aras

All Rights Reserved.

aras.com

Editing an Item

Much like editing an Item in the Aras Innovator user interface, the edit action will:

- Lock the Item specified (you must supply an Item id or use a where clause to locate the item).
- Make any changes described.
- Save the Item.
- Unlock the Item.

In this example, a Part is being modified so that the cost value is set to 22.

Note

You should specify the table name using the bracket notation shown above to avoid "colliding" with reserved terms in SQL Server. Remember that each ItemType is stored in its own table – the name of the table becomes the name of the ItemType (without spaces).

Editing Multiple Items



- Supply a where clause that returns multiple items

```
<AML>
  <Item type="Part" action="edit"
    where="[part].classification='Assembly' >
    <make_buy>Make</make_buy>
  </Item>
</AML>
```

Copyright © 2014 Aras. All Rights Reserved.

aras.com

Editing Multiple Items

If the where clause returns more than one Item, then all Items in that result set are affected by the edit.

In the example above, the where clause will return Assembly type Part Items and set the make/buy to Buy.



Adding Related Items

```

<Item type="Part" action="edit"
      where="item_number='C5316-60108'>

<Relationships>
    <Item type="Part BOM" action="add">
        <related_id>
            <Item type="Part" action="add">
                <item_number>1121-9011</item_number>
            </Item>
        </related_id>
    </Item>
</Relationships>
</Item>

```

① Part
C5316-60108 ② Part BOM
(Relationship) ③ Part
1121-9011

Copyright © 2014 Aras All Rights Reserved.

aras.com

Adding Related Items

As mentioned previously, the add action can also be used to add related items to an existing item.

Using syntax we have already discussed when querying for related items, the related_id tag can be used to assist with this operation.

In this example,

- ① – Part C5316-60108 is being edited.
- ② – A new relationship “Part BOM” is added to the Part.
- ③ – A new Part is added as the related Item for the relationship.

Deleting Items



```
<Item type="Part" action="delete"  
      where="[part].state='Preliminary'">  
</Item>
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

Deleting Items

If the where clause affects more than one Item, then all Items are deleted that meet the required criteria.

In this example, all Parts that are in the Preliminary life cycle state will be deleted.

AML SOAP Message



```
<SOAP-ENV:Envelope xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope/
  ">
  <SOAP-ENV:Body>
    <ApplyAML>
      <AML>
        <Item type="Work Order" action="get" />
      </AML>
    </ApplyAML>
  </SOAP-ENV:Body></SOAP-ENV:Envelope>
```

AML Request Messages

AML requests made to the Innovator Server are wrapped in a SOAP request. The ApplyAML tag is used to execute the AML statement provided in the message. You will learn about the apply method later in this course.

AML SOAP Response



```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/
"><SOAP-ENV:Body>
<Result>
<Item type="Work Order"
typeId="FFCB14F8E0B64FBAB54BA9EF136CED23"
id="248E0A0E1035482FB86DB94E32CB2782"><permis-
sion_id keyed_name="Work Order"
type="Permission">1C4130D0C13C43F687ABF8614F11
DCC4</permission_id><config_id keyed_name="WO-
000340" type="Work Order"></Item>
</Result>
</SOAP-ENV:Body></SOAP-ENV:Envelope>
```

AML Response Message

Once the request has been processed it is returned to the requestor as a SOAP message <Result>. In this example, a Work Order Item is returned based on the request on the previous page.



Summary

In this unit you learned how to work with AML to query, add and update Items.

You should now be able to:

- ✓ Describe the AML Syntax
- ✓ Build AML Queries
- ✓ Use AML Studio to Test and Execute AML
- ✓ Use Built In Actions to Add and Edit an Item using AML



Review Questions

What prevents you from deleting an Item in AML?

What tag is useful for working with the Item that is related to the source Item?

Who should be granted access to the NASH or AML Studio tool?

What attributes are required to edit an Item?



Lab Exercise

Goal:

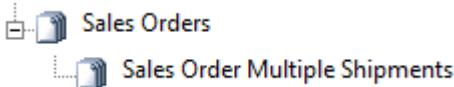
Be able to define and execute custom queries as well as add, edit and delete items using AML commands.

Scenario:

In this exercise, you will create and build several AML queries and that become Saved Searches to be shared with other users. You will also create several AML commands to add, edit and delete Sales Orders in the database. Later we will use these statements to construct Method items using the Innovator Object Model (IOM).

Retrieve Items

1. Create a new AML Search that finds all Sales Orders that allow Multiple Shipments and use the Shipping Method of UPS. Save the Shared search that appears on the TOC with the label "Sales Order Multiple Shipments".



2. Edit the Saved Search Item you created above so that the query finds Sales Orders that allow Multiple Shipments OR uses the Shipping Method UPS. Note: You may need to clear client cache to see the change – select Tools > Admin > Clear Client Metadata Cache from the main menu.
3. Create a new AML Search that finds all Sales Orders where the associated Customer is from the city of New York. Save as a Shared Search that appears on the TOC with the label "New York Customers".



4. Edit the Saved Search Item you created above and restrict the query to only show New York Customers where the Sales Order Ship Date is greater than May 1, 2013. Remember the time format in AML is: yyyy-mm-ddThh:mm:ss.

Add New Items

5. Using AML Studio, create an AML command that will create a new Sales Order and assign the Reviewing Manager (*managed_by_id* Item data type property) to the Identity of Peter Smith and the Shipping Method as UPS.

Reviewing Manager
Peter Smith

Sales Rep

Shipping Info

Shipping Method
UPS

Ship Date

Allow Multiple Shipments?

6. Modify the command you created above so that it creates a new Sales Order and a relationship to the Customer named Brown Industries and the Part with the item_number C4703A with a quantity of 1.

Customer	Parts	Remarks
Brown Industries	213-988-1000	Ken Allen

Customer	Parts	Remarks			
	Quantity	Part Number	Revision	Name	Type
	1	C4703A	A	DesignJet 2000CP Printer	Product

Edit Items

7. Using AML Studio, create an AML command the will change any Sales Orders that allow Multiple Shipments to use the UPS Shipping Method.
8. Modify the command you created above so that is also adds the Sales Order Remark "Modified by AML Administrator" (using the *comment_text* property) to any orders that are changed by this AML command. What type of relationship is Sales Order Remarks

?

Customer	Parts	Remarks
Actions	No Related	
Date [...]	User [...]	
6/19/2013 ...	Innovator Admin	Modified by AML Administrator

This page intentionally left blank.



Unit 3 Creating a Method

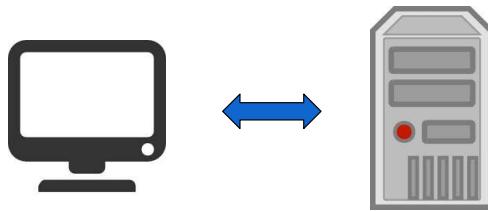
Overview: In this unit you will learn how to create a Client or Server method using the Aras Innovator Solution Studio Editor. You will also learn about the Innovator Object Model and the various methods available to generate AML programmatically.

- Objectives:**
- ✓ Exploring Method Types
 - ✓ Introducing the Innovator Object Model (IOM)
 - ✓ Exploring the API On-Line Reference
 - ✓ Using the Solution Studio Editor
 - ✓ Comparing IOM and AML
 - ✓ Creating a Client and Server Method
 - ✓ Using the Context Item
 - ✓ Using the Item Factory Design Pattern
 - ✓ Returning New Result or Error

Exploring Method Types



- Provide custom business logic to a solution
- Client Methods use JavaScript
- Server Methods use C# or VB.NET
- Innovator Object Model (IOM) can be used to interact with Items



Copyright © 2014 Aras

All Rights Reserved.

aras.com

Exploring Method Types

When you create a Method in Aras Innovator you decide whether Method will be used to execute client code in the browser (JavaScript) or server code in the Innovator Server (C# or VB.NET).

Innovator Methods are special Items that contain source code that gets executed either on the client or server based on which type is selected.

As mentioned earlier, the IOM is available on both the client and server and calls to the IOM methods are basically the same.

Note

Aras Innovator Methods created in the Solution Studio Editor will be referred to as Methods (capital "M") in this course while class methods in the IOM will use lower case "m".

Invoking a Method



- Server Method
 - Item Actions
 - Generic
 - Server Events
- Client Method
 - Item Actions
 - Generic
 - Form, Field, Grid Events
 - Client ItemType Events

Copyright © 2014 Aras All Rights Reserved.

aras.com

Invoking a Method

Methods can be called in several ways depending on their purpose.

Server Methods can be called from an Action (menu driven), from another Method (Generic) or in response to a server event (e.g. OnAdd).

Client Methods can also be called from Actions as well as other client Methods. In addition, there are many events available in the user interface (Form, Field, Grid) that can trigger a client Method. The ItemType also supports several events that can respond to an Item as it is displayed (e.g. OnBeforeNew)

Introducing the Innovator Object Model (IOM)



- API used to generate AML
- Support both Client and Server methods
- Over 200 methods available to interact with Items
- Client and Server implementations share same codebase
 - Namespace: Aras.IOM
 - Contained in: IOM.dll
- On-line API reference available

Copyright © 2014 Aras All Rights Reserved.

aras.com

Common codebase, namespace Aras.IOM in IOM.dll

- Referenced by Method Templates on Server
- Embedded as an Object on the Client
- IOM.dll can be referenced by .Net code
- Alternate compile for use as a COM object in VB6 and VBA etc.

Self documented C# source

- Help>>API Reference>>Aras IOM

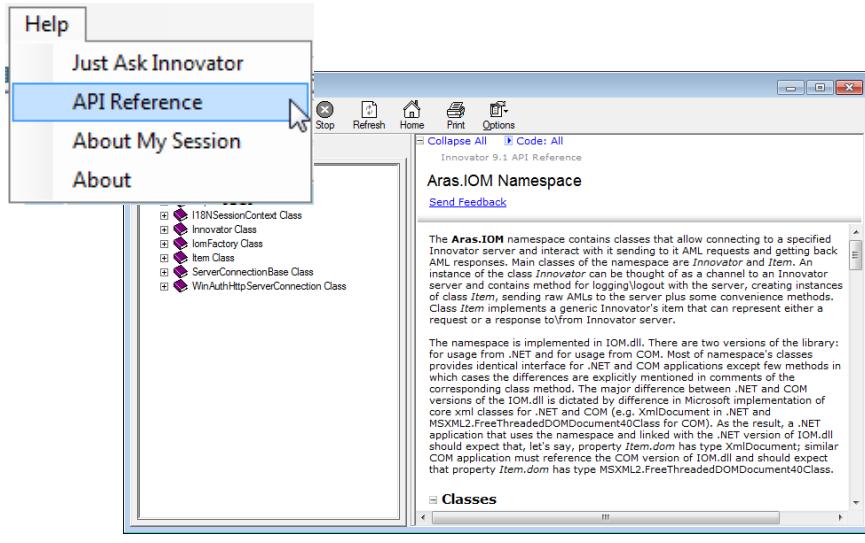
The standard location for the Client IOM.dll is: Innovator\Client\cbin

The standard location for the Server IOM.dll is: Innovator\Server\bin

Introducing the API On-Line Reference



- Over 200 methods available in the IOM



Copyright © 2014 Aras All Rights Reserved.

aras.com

The On-Line Help Reference documents available classes and methods available for your use. This includes .NET controls as well as several classes to interact with the Innovator Server.

Reviewing the Context Item



Methods are executed on the instance of an Item



TKT-0029-2010

Use the keyword *this* to reference the Item (*Me* in VB.NET)

Contains an authenticated connection to the server



```
string value = this.getProperty("priority");
```

Methods represent the behavior of Items. Methods execute in the context of an Item and can refer to it using the keyword *this* (or *Me* using VB.NET).

The context Item may be a single Item, a collection of Items, an Error, just a Result (a string) or be empty. Robust code will check for all of these possibilities.

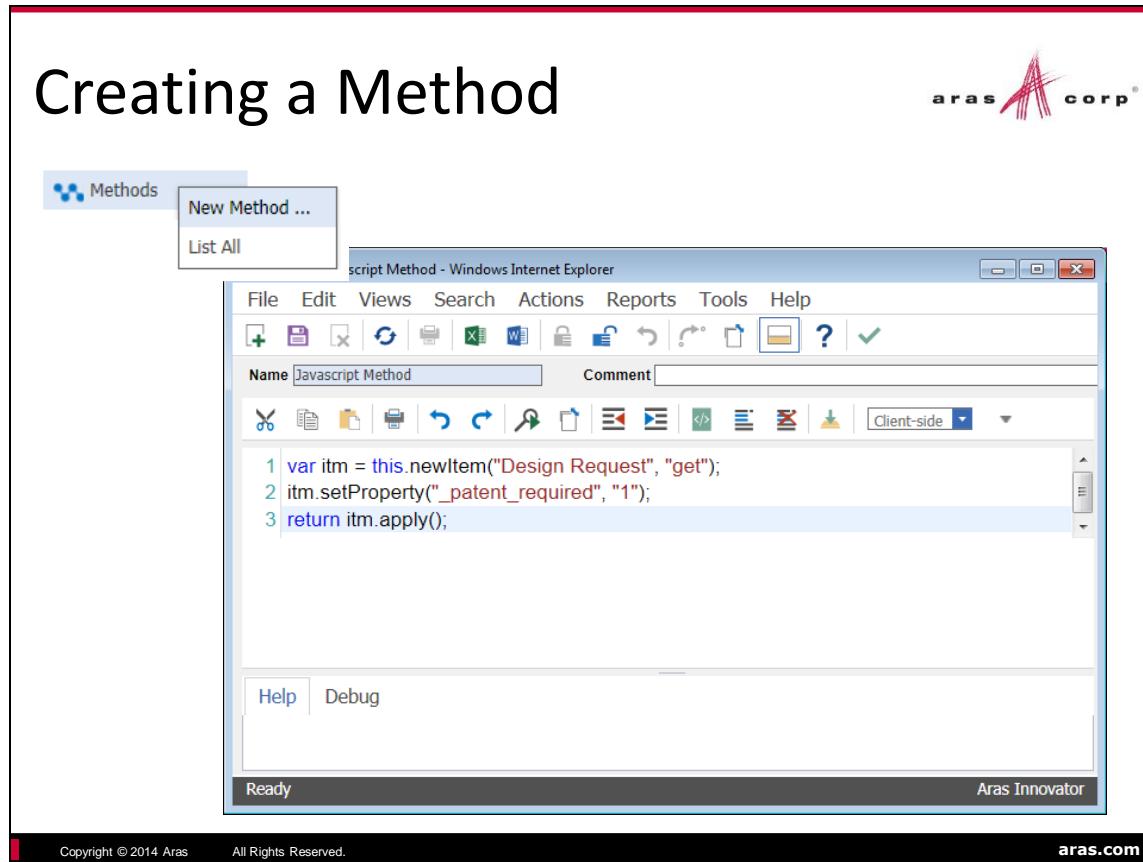
In general, it is a best practice not to alter the context Item, to avoid side effects. An exception to this rule is in certain Server Events where the purpose of the Method is to change the context Item.

Reviewing the Item Factory Design Pattern



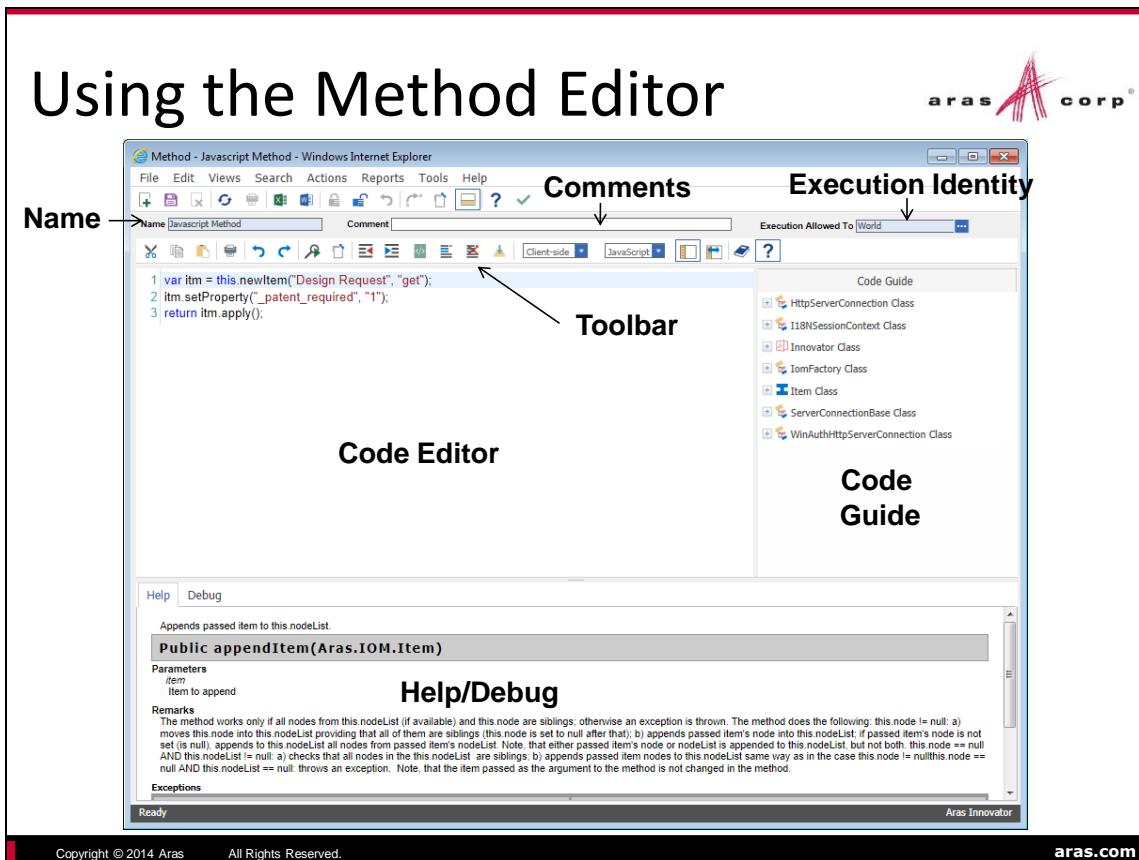
- Methods typically follow an Item Factory Design Pattern
- Method executes on Context Item
- Method should return an Item
 - Recommended for Client Methods
 - Required for Server Methods

A basic rule in the Aras Innovator Methodology is that a Method should always return back an Item. This is optional (but recommended) in client Methods. It is required for a server Method. If your Server Method does not return an Item a compilation error will be raised when you syntax check or attempt to run the Method.



To Create a Method

Choose Administration > Method from the TOC to create a new Method. A Method Item is created in the database to store the source code of the procedure you will invoke from Aras Innovator.



Using the Solution Studio

When you create a Method in Aras Innovator, the Solution Studio Editor is presented to allow you to enter source code and configure some basic settings.

Method Name

Name of the Method Item to be saved in the database. Note this name is case sensitive.

Method Comments

Text description of the Method's purpose.

Toolbar

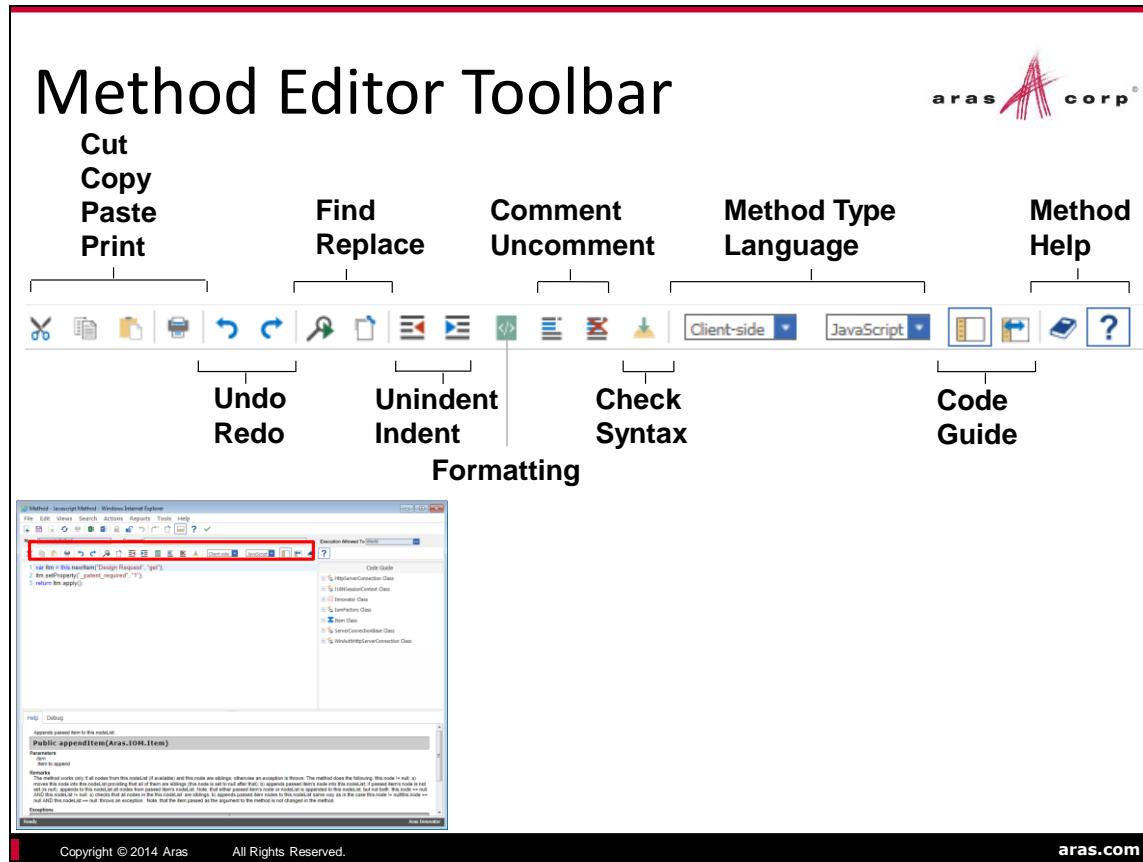
A series of buttons for working with the editor – shown on the next page.

Code Guide

An abbreviated guide to the IOM – API Reference is more complete under Help main menu.

Help/Debug

Two tabs – Debug shows any compilation errors and Help has an abbreviated display of IOM method parameters

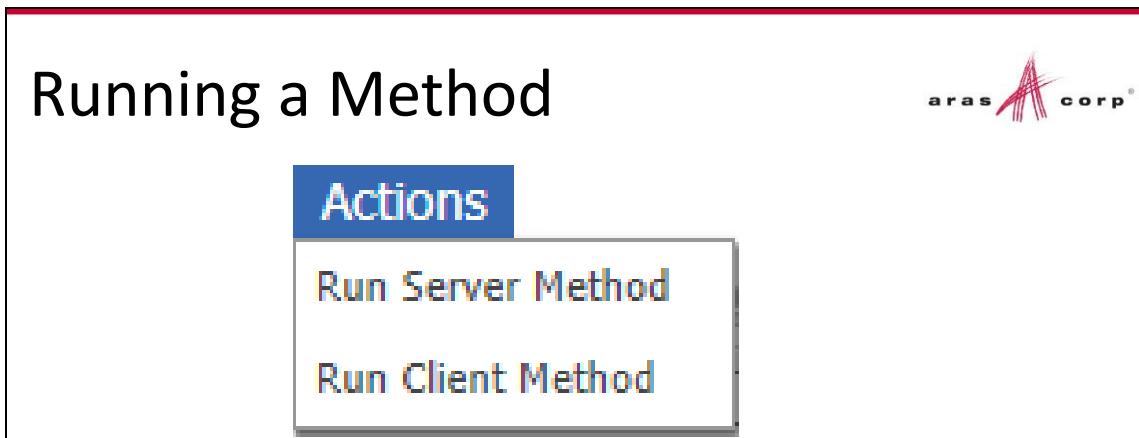


Solution Studio Toolbar

Standard editing functions are available while working in the code editor. The Check Syntax button is a helpful resource that will do a compilation check on the method to look for obvious errors in your code.

Note

Server method syntax checking is more robust as the method is compiled on the server and any compile errors are returned as a message in the Debug window. Client syntax checking relies on the IE browser and is not as thorough (i.e. will not check for IOM method signature errors).



The screenshot shows a software interface for creating methods. At the top, there's a blue header bar with the word "Actions". Below it is a white panel with two main buttons: "Run Server Method" and "Run Client Method". In the bottom left corner of the main workspace, there's a red rectangular box highlighting the "Actions" button.

Method - JavaScript Method

File Edit Insert View Options Import Help

Actions

Run Server Method

Run Client Method

Copyright © 2014 Aras All Rights Reserved. aras.com

Running a Method (Test)

The Run Server Method Action will execute your server Method and display a response window with the output of the response in a separate browser pop-up window. This is helpful for testing purposes.

Run Client Method is an Action that has been provided for training purposes and is not available in the standard product. Use this Action to execute a client Method and review the results. This action uses an aras object JavaScript method that you will learn about later in this course.

Note

When a Method is executed the SAVED copy in the database is the current source of the Method. Remember to always save the Method first before attempting to execute it.



Comparing IOM and AML

■ AML Example:

```
<Item type="Design Request" action="get">
  <_cost>100</_cost>
  <_patent_required>1</_patent_required>
</Item>
```

■ Using the IOM in a Server Method

```
01 Item itm = this newItem("Design Request", "get");
02 itm.setProperty("_cost", "100");
03 itm.setProperty("_patent_required", "1");
04 return itm.apply();
```

Comparing IOM and AML

Remember – everything in Aras Innovator is an Item and every Item is described using Adaptive Markup Language(AML). When you use the IOM on the client or server, you are generating AML that will be sent to the server in the form of a SOAP message. That request, will in turn send a response back in the form of AML.

In the example above, an AML request is made to return back any Items of type Design Request that have a priority Property set to “10” and response_team Property set to “Alpha”.

To create this request using the IOM you use appropriate IOM class methods to generate the AML.

In this example, a new object of type Item is created in the Method (line 1). Two properties are then set on the object using setProperty. Finally, the object is sent to the server for process which will return back an AML response containing any Items found.

Note

Understanding AML is key to working with the IOM effectively.

VB.NET Example

```
Dim itm As Item = Me.newItem("Design Request", "get")
itm.setProperty("_cost", "100")
itm.setProperty("_patent_required", "1")
Return itm.apply()
```

Comparing IOM and AML



■ AML Example:

```
<Item type="Design Request" action="get" select="id">
  <_cost>100</_cost>
  <_patent_required>1</_patent_required>
</Item>
```

■ Using the IOM in a Client Method

```
01 var itm = this newItem("Design Request", "get");
02 itm.setProperty("_cost", "100");
03 itm.setProperty("_patent_required", "1");
04 itm.setAttribute("select", "id");
05 return itm.apply();
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

In this example an attribute is set in the AML request to select the id property from the Help Tickets found. Using the *select* attribute can greatly reduce the amount of data that needs to be returned from the server and can increase performance of a solution.

The *setAttribute* IOM method can be used to set the same attribute and generate the AML displayed in this example.



Previewing the Item and Innovator Classes

- **Innovator Class**
 - Contains authenticated connection to Innovator Server
 - Sends applyAML requests to the Innovator Server
 - Can create new Items
 - Support other non-related Item operations
- **Item Class**
 - Represents the Item instance
 - Perform Item related operations
 - Contains a property field (dom) representing an Item DOM object

The Innovator and Item classes are the two main classes that contain many methods to generate and send AML requests to the Innovator Server. You will learn about many of these methods in this class.

Supported IOM Item Types



- Single
 - Single Instance of an ItemType
- Collection
 - A set of Items (e.g. "get" action that returns multiple Items)
- Error
 - Request fails against the Innovator Server
- Result
 - Arbitrary text wrapped by <Result> tags
- Logical
 - Set of property wrapped in logical statements (query)

An Item object can represent several types of data structures.

The IOM is intended to be a generic and compact API for modeling the Item structure of the AML as abstract Objects. The majority of the methods for the IOM deal with memory management of the AML document for the Item Object. The AML is a script sent as a message to the Aras Innovator Server; the IOM is an Object API to build the AML messages, submit them to the Innovator Server and parse an AML document that is returned by the Server.



Creating an Item Object

■ Client (JavaScript)

```
01 var myItem = this newItem("Design Request", "add");  
02 myItem.setProperty("_title", "New PC Type");  
03 myItem.setProperty("_cost", "500");  
04 return myItem.apply();
```

■ Server (C#)

```
01 Item myItem = this newItem(" Design Request", "add");  
02 myItem.setProperty("_title", "New Mobile Phone");  
03 return myItem.apply();
```

Because everything is an Item in Aras Innovator the Item object is key to any Methods that are created to interact with the Innovator Server.

The newItem method allows you to create an Item object in a program, set various properties and attributes, and apply them against the Innovator Server. You will learn about more options for this and other methods later in the course.

VB.NET

```
Dim myItem As Item = Me newItem("Design Request", "add")  
myItem.setProperty("_title", "New Mobile Phone")  
Return myItem.apply()
```

Editing an Item Object



■ Client (JavaScript)

```
var myItem = this newItem("Design Request", "edit");
myItem.setAttribute("where",
    "[design_request]._item_number='DR-000010'";
myItem.setProperty("_cost", "250");
return myItem.apply();
```

■ Server (C#)

```
Item myItem = this newItem(" Design Request ", "edit");
myItem.setAttribute("where",
    "[design_request]._item_number='DR-000010'";
myItem.setProperty("_cost", "250");
return myItem.apply();
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

As you learned in a previous unit of the course, to change an existing Item you use the edit action which requires a “where” clause to provide the required criteria.

The Item class setAttribute method can be used to assign the where clause to the edit request, and setProperty can be used to set the changed property value.

VB.NET

```
Dim myItem As Item = Me newItem("Design Request", "edit")
myItem.setAttribute("where",
    "[design_request]._item_number='DR-000010'"
)
myItem.setProperty("_cost", "250")
Return myItem.apply()
```



Using the Innovator Object

■ Client (JavaScript)

```
01 var inn = this.getInnovator();  
02 var myUserID = inn.getUserID();  
03 top.aras.AlertSuccess(myUserID);  
04 return this;
```

■ Server (C#)

```
01 Innovator inn = this.getInnovator();  
02 string myUserID = inn.getUserID();  
03 return inn.newResult(myUserID);
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

The Innovator class provides fundamental methods for working in the IOM. You will learn about these methods later in this course.

The Innovator object is easily obtained from the Method context Item by using the getInnovator() method.

This example returns the GUID of the currently logged in User.

VB.NET

```
Dim inn As Innovator = Me.getInnovator()  
Dim myUserID As String = inn.getUserID()  
return inn.newResult(myUserID)
```

Returning A New Result Item



■ Client (JavaScript)

```
01 var myItem = this newItem("Design Request", "get");
02 myItem.setProperty("_item_number", "DR-000010");
03 myItem = myItem.apply();
04 var cost = myItem.getProperty("_cost");
05 return this.getInnovator().newResult(cost);
```

■ Server (C#)

```
01 Item myItem = this newItem("Design Request", "get");
02 myItem.setProperty("_item_number", "DR-000010");
03 myItem = myItem.apply();
04 string cost= myItem.getProperty("_cost");
05 return this.getInnovator().newResult(cost);
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

The newResult method takes a string as its argument which it then transforms into an Item object.

The string will be enclosed in a <Result> tags as follows:

<Result>10</Result>

VB.NET

```
Dim myItem as Item = Me newItem("Design Request", "get")
myItem.setProperty("_item_number", "DR-000010")
myItem = myItem.apply()
Dim cost As String = myItem.getProperty("_cost")
Return Me.getInnovator().newResult(cost)
```

Returning a New Error Item



■ Server (C#)

```
Innovator inn = this.getInnovator();

Item myItem = this newItem("Design Request", "get");

myItem.setProperty("_item_number", "DR-000010");

myItem = myItem.apply();

string costval = myItem.getProperty("_cost");

if (costval=="0") {

    return inn.newError("Cost cannot be zero.");

}

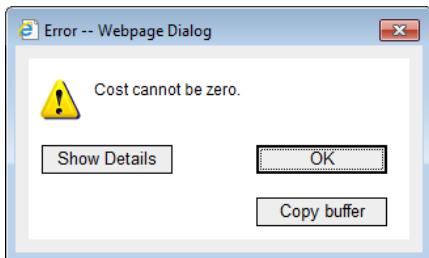
return myItem;
```

Copyright © 2014 Aras

All Rights Reserved.

aras.com

An Error Item will display a dialog on the client and can be configured with additional attributes you will learn about later in this course. The example above will produce an error if the cost is set to zero:



VB.NET

```
Dim inn As Innovator = Me.getInnovator()
Dim myItem As Item = Me newItem("Design Request", "get")
myItem.setProperty("_item_number", "DR-000010")
myItem = myItem.apply()
Dim costval As String = myItem.getProperty("_cost")
If costval = "0" Then
    Return inn.newError("Cost cannot be zero.")
End If
Return myItem
```



Summary

In this unit you learned how to create new client and server Methods and some fundamental IOM methods that are used to generate AML.

You should now be able to:

- ✓ Use the Solution Studio Editor to Create a Method
- ✓ Understand the Basics of the Innovator Object Model (IOM)
- ✓ Compare IOM and AML
- ✓ Execute a simple Client or Server Method
- ✓ Explore the API On-Line Reference
- ✓ Use the Context Item in a Method
- ✓ Use the Item Factory Design Pattern
- ✓ Return a New Result or Error



Review Questions

What are some basic differences between client Method and a server Methods?

What should every Method return? When is it required vs. recommended?

What does every IOM method generate?

What IOM methods do you use to retrieve an Item(s) from the database using the IOM?



Lab Exercise

Goal:

Be able to create basic client and server Methods and test them in the Innovator client.

Scenario:

In this exercise, you will create several Methods that use the IOM to generate AML to the server. You will run these Methods using the Run Server and Run Client Method Actions available from the Solution Studio Editor.

Retrieve Items

1. Create a Server Method named **Server – GetOrders** that retrieves all Sales Orders that allow Multiple Shipments and use the Shipping Method of UPS. Unit test the Method using the Run Server Method Action in the Method editor.
2. Create a Client Method named **Client – GetOrders** that performs the same command as above but only retrieves the Sales Order number. You must use the "select" attribute to accomplish this. What IOM method allows you to assign an attribute? _____
Unit test the Method using the Run Client Method Action in the Method editor.

Add Items

3. Create a Server Method named **Server – AddOrder** that will create a new Sales Order with the Shipping Method of Federal Express. Unit test in the Method editor and log on to Aras Innovator to make sure the item is created.
4. Modify the Method you created above to also assign a Purchase Order number (using today's date and time) using the following .NET function:

C#:

```
string po = DateTime.Now.ToString("yyMMddhhmmss");
```

VB:

```
Dim po As String = DateTime.Now.ToString("yyMMddhhmmss")
```

Edit Items

5. Create a Server Method named **Server – EditOrders** that will modify any Sales Orders that allow Multiple Shipments to use the UPS Shipping Method. You must use the "where" attribute to edit the item.

This page intentionally left blank.



Unit 4 Debugging Client/Server Methods

Overview: In this unit you will learn how to create a Client or Server method using the Aras Innovator Solution Studio Editor. You will also learn about the Innovator Object Model and the various methods available to generate AML programmatically.

- Objectives:**
- ✓ Configuring the Server Method Debugger
 - ✓ Using Visual Studio
 - ✓ Using MS CLR Debugger (.NET SDK)
 - ✓ Configuring the Client Method Debugger
 - ✓ Using Visual Studio
 - ✓ Using Microsoft Script Debugger
 - ✓ Configuring Logging
 - ✓ Clearing Client and Server Cache
 - ✓ Debugging Production Code



Objectives

- Configuring the Client Method Debugger
 - Using Visual Studio
 - Using Microsoft Script Debugger
- Configuring the Server Method Debugger
 - Using Visual Studio
 - Using MS CLR Debugger (.NET SDK)
- Configuring Logging
- Clearing Client and Server Cache
- Debugging Production Code
- Inspecting AML Requests/Responses

Configuring a debugger for client and server Methods is probably one of the most important steps you will take when you begin to create custom solutions (unless you are a perfect coder!).

Because Aras Innovator is a Microsoft .NET server application you can easily configure Visual Studio or the Microsoft Script Debugger to execute at a break point in your Method.

Just-In-Time debugging is a feature that launches the Visual Studio debugger automatically when a program, running outside Visual Studio, encounters a fatal error. Just-In-Time debugging allows you to examine the error before the application is terminated by the operating system.

Configuring the Server Method Debugger



- Using Visual Studio
 - Standard Edition or higher
 - Visual Studio Express does not support JIT debugging
- Using Microsoft CLR Debugger
 - Included with Microsoft .NET SDK
- Invoking Server Method Debugger
 - Include the following at start of Server Method;

```
01 System.Diagnostics.Debugger.Launch();
```

Copyright © 2014 Aras

All Rights Reserved.

aras.com

Note

It is sometimes necessary to attach to a named process in order for the Just In Time debugger to execute successfully.

From the Visual Studio main menu select Tools >> Attach To Process ...

- For Windows XP attach to process: **aspnet_wp.exe**
- For Windows Server 2003, 2008 and Windows 7 attach to process: **w3wp.exe**

The .NET System.Diagnostics.Debugger.Launch() command will start the Visual Studio JIT debugger if the debugger has not been started. To run the same Method again in the same debug session you can use the System.Diagnostics.Debugger.Break() command.

The following code will work for first time and repeat invocations of the same Method:

```
if (System.Diagnostics.Debugger.Launch()) {
    System.Diagnostics.Debugger.Break();
}
```

VB.NET

```
If System.Diagnostics.Debugger.Launch() Then
    System.Diagnostics.Debugger.Break()
End If
```

Invoking Server Debugging

The following tag must be set in the InnovatorServerConfig.xml file located on the server to support server Method debugging:

```
<operating_parameter key="DebugServerMethod" value="true" />
```

The default setting is "false".

Each time a server Method is created and saved successfully, a corresponding dll assembly file as well as a program database file (pdb) can be created containing the compiled program code for debugging purposes.

To further assist in debugging server Methods, a “smart naming” feature is available when the corresponding dll and pdb files are created. To activate this feature, the *DebugServerMethod* key must be set to “true” as described in the previous section. In addition, the following key must be provided in the InnovatorServerConfig.xml file:

```
<operating_parameter key="ServerMethodTempDir" value="[path]" />
```

where path is equal to a directory location to store the resulting dll and pdb files.

If both settings are provided the following corresponding files are created and named using the following notation:

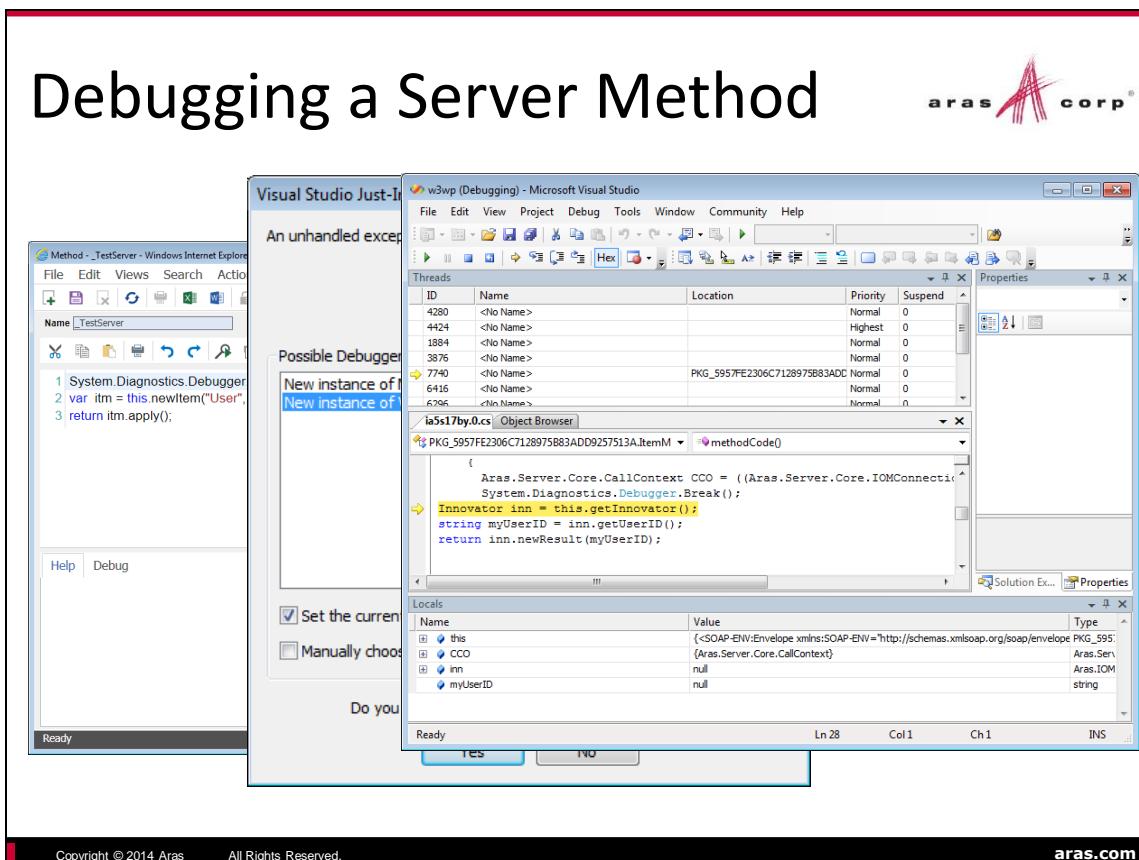
[release and build number].[database_name].[Method_name].[Method_GUID]

For example, if a server Method is created named *SampleMethod* in the InnovatorSolutions database the following files will be generated in the location indicated by the ServerMethodTempDir key:

10.0.0.5846.InnovatorSolutions.SampleMethod.17FBE8EAA7DD41329FE52D6E3CEE135A.dll
10.0.0.5846.InnovatorSolutions.SampleMethod.17FBE8EAA7DD41329FE52D6E3CEE135A.pdb

IIS Settings

The Windows IIS server default the debug timeout on a single Method step to 90 seconds which causes the client to hang when time expires. You can change this behavior by selecting the Aras Application Pool in IIS and accessing the Advanced Setting. From the context menu set Process Model/Ping Enabled to False (default true), or increase the Ping Maximum Response Time.



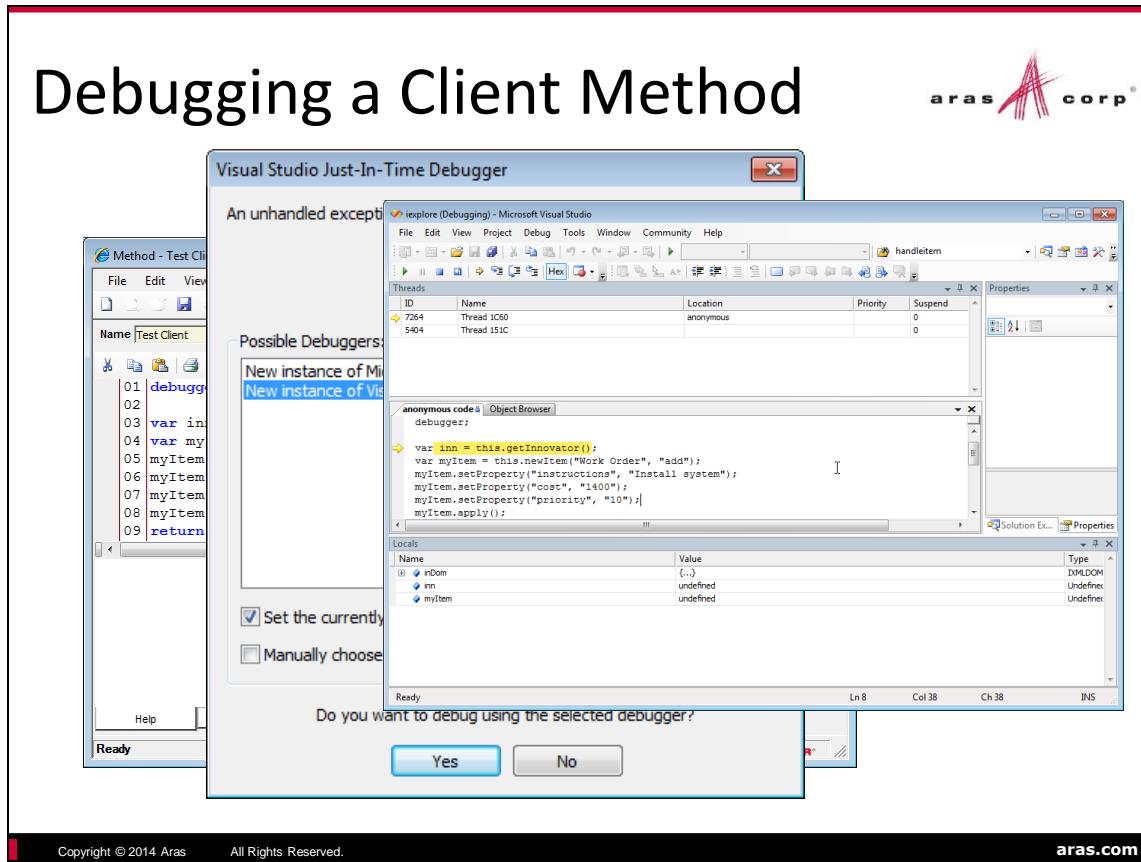
You can use all of the Visual Studio debugging features available (single step, quick watch, etc.) while examining your Method code.

Notes

As discussed earlier, your Method runs inside of the Aras Framework – you can view the framework code while examining your own Method.

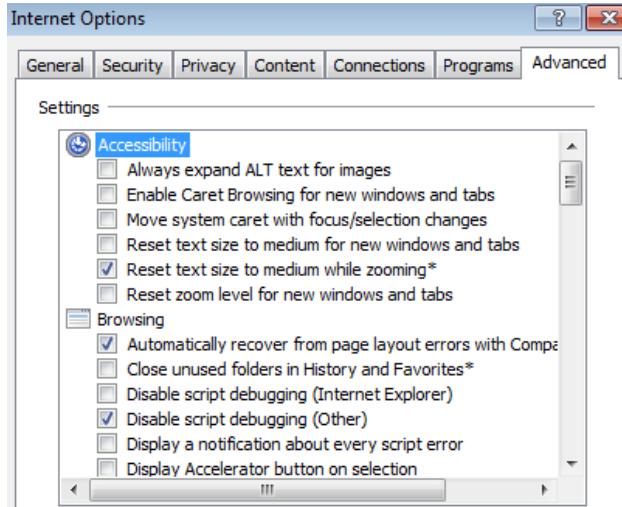
You can also write debug messages to the Visual Studio Output window by using the .NET Write method of the System.Diagnostics.Debug class :

```
System.Diagnostics.Debug.WriteLine("Result is" + var);
System.Diagnostics.Debug.Write(item);
```



To Configure Client Debugger (Internet Explorer):

1. Add the statement `debugger;` at the point you want to set as breakpoint to invoke the debugger.
2. Deselect *Disable Script Debugging* in Advanced Options of Internet Explorer. Select *Display a notification about every script error* in Advanced Options of Internet Explorer



Using debugger Command



- Sets breakpoint in JavaScript client Method

```
01 debugger; ← Debugger Command
02 var itm = this newItem("Design Request", "get");
03 itm.setProperty("_patent_required", "1");
04 return itm.apply();
```

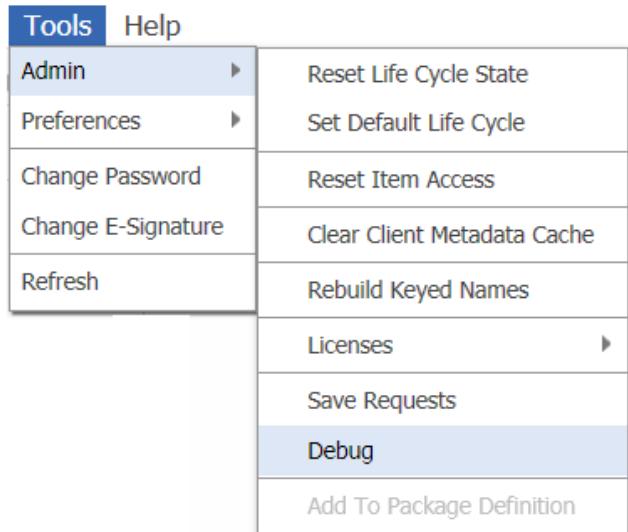
Client Debugging with Visual Studio 2010

Add the *debugger* statement to cause Aras Innovator set a breakpoint in your client Method and call a script debugger.

To debug client JavaScript methods with VS 2010 (or greater) requires additional setup. Use the following steps to debug a client program:

1. Start Aras Innovator in Internet Explorer but do not logon to the system.
2. Start Visual Studio 2010.
3. From the VS 2010 main menu, select "Tools" and "Attach to Process"
4. Locate the "iexplore.exe" process with the "Title" that matches the Aras Innovator instance.
5. Select the process and click "Attach".
6. Return to the login screen for Aras Innovator and press F5 to refresh the page.
7. Log in to Aras Innovator
8. Run the Method and the application will switch to the "Microsoft Visual Studio .NET 2010" debugger and break on the *debugger;* statement in a client JavaScript method.

Setting Client Debug Option



- top.aras.DEBUG is set to TRUE

Copyright © 2014 Aras All Rights Reserved.

aras.com

You can set and use a simple flag to indicate when you want the client debugger to be invoked. If you choose Debug from the Tools menu property named DEBUG is set to true on the client aras object.

Example

If the following code is placed at the top of a client Method, the debugger will be invoked if the Debug menu is set to true (checked):

```
if (top.aras.DEBUG) {  
    Alert("Debug is set");  
    debugger;  
}
```

Inspecting an Item



- Item contains:

- dom – Document Object Model for Item
- node – item node (if a single Item)
- nodelist – item nodes (if more than one Item returned)

	item	
	dom	{<SOAP-ENV:Envelope xmlns:SOAP-ENV:
	node	{Document}
	nodelist	null
		{System.Xml.XPathNodeList}

Copyright © 2014 Aras

All Rights Reserved.

aras.com

Typically, it is useful to examine the Item in question while debugging your Method code. Remember that Document Object Model of the Item is contained in the dom property, while the actual Item node (or nodes) is displayed in the node or nodelist property respectively.

If a single Item is constructed (returned) from the Server, the node property points to a single instance. If a collection is returned, the nodelist property points to this collection.

The dom property represents an instance of the .NET System.Xml.XmlDocument class.

The node property represents an instance of the System.Xml.XmlElement class.

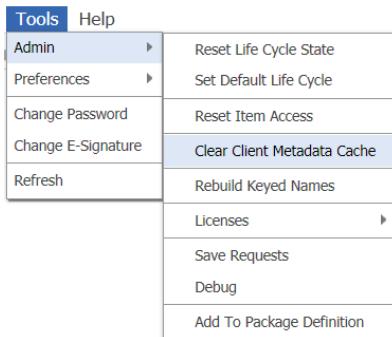
The nodelist property represents an instance of the System.Xml.XmlNodeList class.

Clearing Client Cache



■ Client Cache

- Close all Internet Explorer Windows
- Close any IEXPLORE.EXE processes
- Delete all temporary Internet Explorer Files



← **Clear Metadata Caches**

Aras Innovator caches the Item DOM in memory along with other browser information. At times, it may be necessary to clear this cache to resolve a problem or prevent the inspection of “stale” data. When a client is started, Aras Innovator will cache metadata about commonly used Item types that a user works with in a session to improve performance. You can clear this cache by selecting *Clear Client Metadata Cache* from the *Tools > Admin* menu.

You should also delete any IE temporary files using the IE browser delete files feature.

Configuring Logging



- Writing to a log file:
 - CCO.Utilities.WriteDebug("logfilename", stringvar)
- Appends *stringvar* text to log file
- Log files written to "temp_folder" location
 - Configured in InnovatorServerConfig.xml file
- Example:

```
01 string msg = "Changed variable value";
02 CCO.Utilities.WriteDebug("DebugMsg", msg);
```

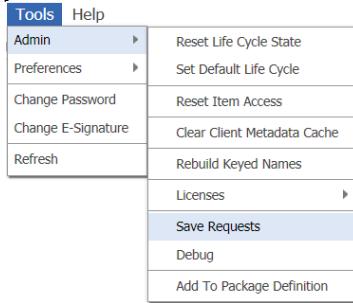
The CCO class (Core Context Object) contains several useful utility methods including the ability to write debug messages to a log file. The WriteDebug method accepts two arguments – the name of the log file (the .log extension is automatically applied) and a string which represents the log text to append to the file. In the example above, the log file DebugMsg.log will be created (if it does not already exist) and the message will be added to the file with a date and time stamp.

The location of the debug log is indicated by the temp_folder setting in the InnovatorServerConfig.xml file.

Debugging Production Code



- Adding Server Logging to Methods
- Logging Requests



- Logging Server Events – InnovatorServerConfig.xml

```
<operating_parameter key="debug_log_flag" value="true"></operating_parameter>
<operating_parameter key="debug_log_prefix"
    value="C:\Program Files (x86)\Aras\Innovator94\Server\Logs\">
</operating_parameter>
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

Debugging production code presents its own set of issues as it may not be possible to set breakpoints or disturb the system as it is running.

You can:

- Add logging to Methods are described previously
- Select the Save Requests option from the Tools menu – this will generate request files on the server containing the AML request and response messages – note that this can affect performance
- Enable the logging of server events by configuring the InnovatorServerConfig.xml file to log server events in the location specified with the debug_log_prefix parameter.



Summary

In this unit you learned how to configure Aras Innovator to work with a Just in TimeDebugger like Microsoft Visual Studio to be able to debug both client and server Methods.

You should now be able to:

- ✓ Debug Client Methods
- ✓ Debug Server Methods using Visual Studio
- ✓ Configure and Review Logs
- ✓ Clear Client and Server Cache
- ✓ Understand the implications of debugging a production problem



Lab Exercise

Goal:

Be able to debug client and server Methods using the Visual Studio debugger.

Scenario:

In this exercise, you will debug a server Method to determine and correct a problem.

Time:

20 minutes

Steps:

1. Locate the Method named **DebugMe**. This server Method was created to return the current purchase order number of a Sales Order as a Result and does not run correctly.
2. Check the Method syntax first to correct any issues before trying to run or debug the Method and then insert the appropriate statement to start the Visual Studio Debugger and run the Method.
3. Use Single Step (F10) to step over each statement in your Method and notice how the AML is generated at each statement for the Item you are working with.
4. Step past the apply() method statement in the code and inspect the Item(s) returned.
5. Compare the property names used in the Method with the Sales Order ItemType properties.
6. Correct the Method so that it operates as intended.



Unit 5 Creating Custom Actions

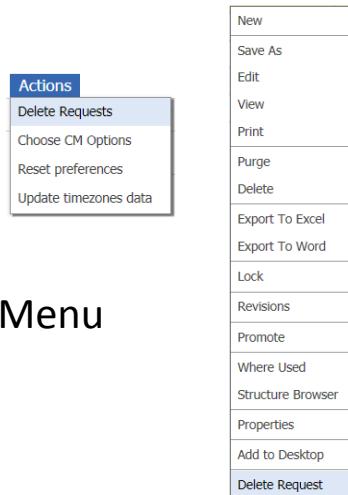
Overview: In this unit you will learn how to build Actions that appear on the client menu to allow a user to perform an operation. You will learn about the three kinds of Actions available and use an Action that displays a custom editable dialog in a solution.

- Objectives:**
- ✓ Exploring Actions
 - ✓ Exploring Actions with Client Methods
 - ✓ Examining Client Cache and Actions
 - ✓ Exploring Actions with Server Methods
 - ✓ Creating Item Actions
 - ✓ Creating Generic Actions
 - ✓ Passing Arguments to a Method from an Action
 - ✓ Creating Custom Interactive Dialogs



Defining Actions

- Action Items bind Methods to Client User Interface
- Action is invoked on client and can execute:
 - Client Method
 - Server Method
- User selects from:
 - Action Menu Bar
 - Mouse Popup Context Menu



An Action Item is used to bind Methods to the client user interface. Actions provide the hooks for invoking Methods from the Action menu bar or from the right mouse context popup menu. An Action is invoked on the client side but can call either a client or server side Method. In both cases the context Item must be defined.

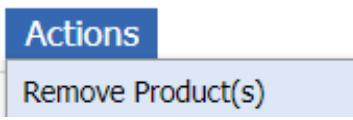
Defining Action Types



- Three Action types are supported:
 - **Generic** – Action applies to any Item(s)



- **ItemType** – Action applies to any Item(s) of a specific ItemType



- **Item** – Action applies to current Item selected by a user

Actions	
Form Name	Type
Design Request	Default View "Form"
Design Request Internal	Default View "Views"

Copy
Lock
RebuildViewAction

Generic Actions appears under the Actions main menu, ItemType and Item Actions appear under the Tear Off window Actions menu, and Item Actions also appear on the search grid context menu (access with right mouse button).

Defining Actions



To Create a New Action

Select Action from Administration in the TOC and create a new Action Item. The following fields are available:

① Name

Name of the Action (used for Menu Item entry text).

② Type

Generic, Item, ItemType (Note: URL is not supported).

③ Location

Location of Method – Client or Server

④ Label

Used in installations that support multiple languages – discussed later in this course.

⑤ Method

Name of the Client or Server Method

⑥ Target

None – no window is displayed when Method executes

Main – Method results appear in the main workspace window frame

Window – Method results appear in a popup window – subsequent calls to the Action display new windows.

One Window – Same as Window, except Method results are appended in the same window.

⑦ On Complete

Optional Method that executes following completion of Action main Method.

⑧ **Body**

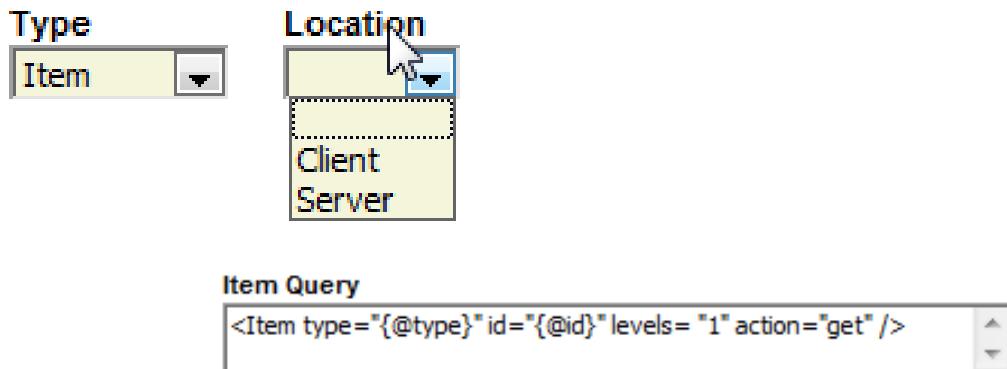
Additional arguments to be passed to Action Method. Discussed in this unit.

⑨ **Query**

Override of standard AML query to define context Item for the Action Method. (Only applies to Item Actions.)

Defining Item Actions

- Are associated with the current Item
- Can be Client or Server based
- Context Item is defined by supplying an Item



An Item Action operates within the context of a single Item. An Item Query is generated by default which returns the current Item the user is working on. This can be modified in the Query field on the Action window.

The Item Query defines the context Item that the associated Method will access when the Action is executed. You can use the keyword "this" (JavaScript/C#) or "Me" (Visual Basic) to access the currently selected Item.

Note

When a user selects an Item to execute an Action from the search grid, the current Item's AML "action" attribute is set to "get". You must change the "action" attribute to "edit" to allow changes to be made to an existing Item – and apply the changes to the server.

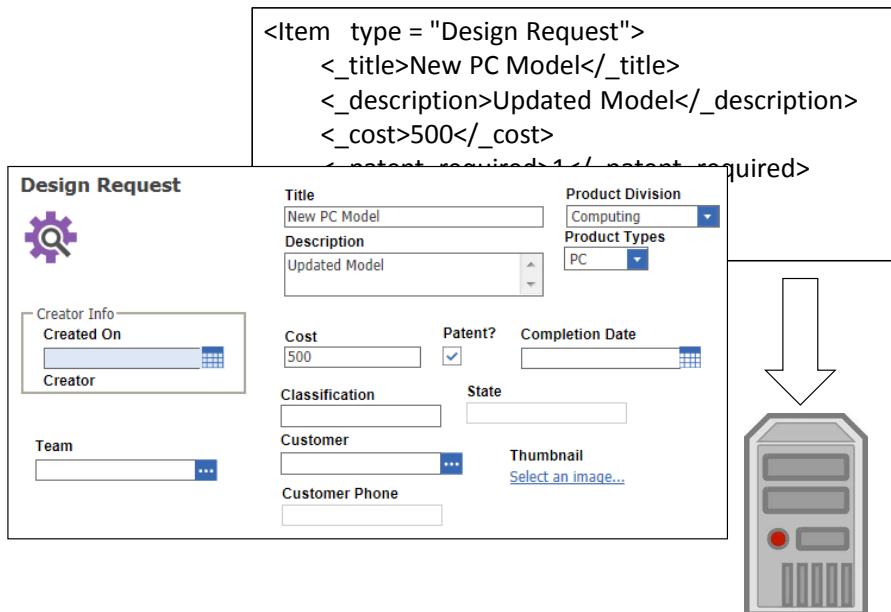
C# Example:

```
this.setAttribute("action", "edit");
this.setProperty("priority", "1");
return this.apply();
```

VB Example:

```
Me.setAttribute("action", "edit");
Me.setProperty("priority", "1");
return Me.apply();
```

Reviewing Client Cache



Slide 7 Copyright © 2014 Aras

All Rights Reserved.

aras.com

Determining Client Side Method Context Item

The following rules are used to determine the current context Item for a client Action method:

If the Item is cached and is “dirty” (has been modified before Action is called) then it is considered the context Item regardless of the Item query in the Action form, otherwise the Action query is applied to obtain the current context Item.

If no query has been supplied in the Action form then the cache Item is considered the context Item.

If no query has been supplied and there is no Item in cache then a “temporary” Item is used to represent the context Item with just the Item type and ID attributes supplied.



Defining the Action Method Context Item

■ Client Item Method:

Item Query

```
<Item type="{@type}" id="{@id}" levels= "1" action="get" />
```

■ Server Item Method

Item Query

```
<Item type="{@type}" action="get"><id><xsl:value-of  
select="@id" /></id></Item>
```

A default query is supplied in the Action form when you choose type Item. Two different queries are defined based on whether a client or server Method will be used with the Action.

This query is for Items with no changes (cache is not dirty) –to determine the DOM for the context Item

If cache is dirty then changes are included in the DOM so they can be saved to server on both client and server methods

The Query syntax is defaulted automatically but can be changed – note however it will return to default if the Action is edited and saved again.

The Syntax is different for client vs. server to maintain backward compatibility.

Defining ItemType Actions



- Associated with an ItemType
- Action Menu item only appears when user selects ItemType from TOC
- Server Method Context Item is the executing Method
 - *this* keyword (or Me in VB.NET)

ItemType Actions define some logic for a class of Items of the same type.

The "Actions" menu item will appear when a user selects the assigned ItemType from the TOC.

Note that the “this” keyword (Me in VB.NET) in a server event refers to the current Method that is executing. To obtain access to a different Item requires the use of the newItem method discussed earlier in the course.

Relating Item and ItemType Action to ItemType



	Name	Type	Location	Method [...]	Target	Body [...]	Sort Order
	PE: rollup all parts in...	ItemType	Server	PE_RollupAllPartsInDB	None		768
	PE_ManualRelease	Item	Server	PE_ManualRelease	None		896
	PE_CreateNewRevisi...	Item	Server	PE_CreateNewRevisi...	None		1024
	PE_AddToChange	Item	Client	PE_GetSelectedItems	None		1152

The screenshot shows the Aras Innovator web interface for managing ItemTypes. The top navigation bar includes File, Edit, Views, Search, Actions, Reports, Tools, and Help. The main content area is titled 'ItemType - Part - Windows Internet Explorer'. It displays the 'Actions' tab of the ItemType configuration page. The Actions table lists the following items:

	Name	Type	Location	Method [...]	Target	Body [...]	Sort Order
	PE: rollup all parts in...	ItemType	Server	PE_RollupAllPartsInDB	None		768
	PE_ManualRelease	Item	Server	PE_ManualRelease	None		896
	PE_CreateNewRevisi...	Item	Server	PE_CreateNewRevisi...	None		1024
	PE_AddToChange	Item	Client	PE_GetSelectedItems	None		1152

Slide 10

© 2014 Aras. All Rights Reserved.

aras.com

To Relate an Action to an ItemType

1. Edit an ItemType and Select the Actions tab.
2. Create a new relationship to the Action.

Defining Generic Actions



- Appear on top Actions Menu
- Useful for working with any ItemType
- Client Method accesses Context Item using
 - `top.aras.newIOMInnovator();`
- Server Method accessed Context Item using
 - `this` keyword (or `Me` in VB.NET)

Generic Actions are used for general activities that are not associated with a specific Item or ItemType. You defined access to the context Item in the same way as an ItemType Action.



Passing Arguments to an Action Method

- Arguments passed in Action Body:

Body

```
<Discount>.10</Discount>
```

- Can be retrieved in Client or Server Method:

```
01 string disc = this.getProperty("Discount", "0");  
02 decimal val = System.Convert.ToDecimal(disc);
```

To pass an argument to a Method in an ItemType or Generic Action use the AML property tag to define the argument name. The value of the argument can then be retrieved using the getProperty method discussed earlier in this course.

Note

Arguments are used for Generic and Itemtype actions only- they are not used in Item actions. A more complete discussion of argument passing with Generic Methods is discussed later in this course.

VB.NET

```
Dim disc as String = Me.getProperty("Discount", "0");  
Dim val as Decimal = System.Convert.ToDecimal(disc);
```



Summary

In this unit you learned how to build custom Actions that are available from the Innovator client to perform custom operations or show an interactive Action dialog box.

You should now be able to:

- ✓ Create a Client or Server Actions
- ✓ Associate the Action with an Item
- ✓ Create Generic Actions
- ✓ Pass Arguments to a Method from an Action
- ✓ Create an Interactive Dialog using a Form



Lab Exercise

Goal:

Be able to create an Action that calls a Method to perform a custom user function.

Scenario:

In this exercise, you will use Server and Client Methods to create custom Actions that user can execute.

Generic Action

1. Create a Server Method named **Server-GetMachineName** that will retrieve the name of your current server. You can obtain this using the .NET instruction:

C#

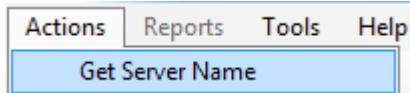
```
string machine = System.Environment.MachineName;
```

VB:

```
Dim machine As String = System.Environment.MachineName
```

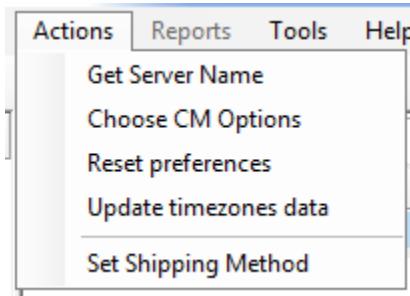
Return the machine name as a new Result item. See Unit 3 to review how to return a Result item.

Attach this method to a Generic action named Get Machine Name. Make sure the Action Target is Window.



ItemType Action

2. Create an ItemType Action named **Set Shipping Method** that uses the **Server-EditOrders** Method you created in an earlier exercise. Set the Action Target to None. Attach the Action to the Sales Order ItemType.



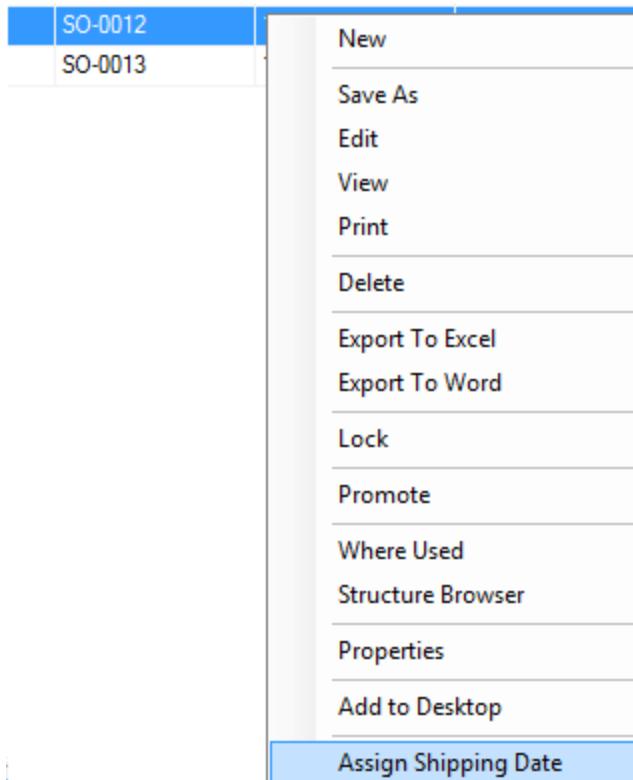
Item Action

3. Create a client Method that will assign today's date to the Shipping Date of an existing selected Sales Order. Assign the Method to an Item Action named **Assign Shipping Date**.

A date property is always set using a "neutral" date string and must be formatted as "yyyy-mm-dd". Example: "2010-06-10"

The following JavaScript will obtain the date in the correct format:

```
var today = new Date();
var dd = today.getDate();
var mm = today.getMonth()+1; //January is 0!
var yyyy = today.getFullYear();
if(dd<10) {dd = '0'+ dd;}
if(mm<10) {mm = '0'+ mm;}
today = yyyy+'-'+mm+'-'+dd;
```



This page left blank intentionally.



Unit 6 Exploring the Innovator Class

Overview: In this unit you will learn how to use the various methods available in the Innovator Class. These foundation methods allow you to execute AML requests, process SQL statements, create new Items and return a new Result or Error.

Objectives:

- ✓ Classifying IOM Innovator method types:
- ✓ Creating Generic Methods
- ✓ Creating AML Action Methods



Base methods

- **applyAML**
 - Submit AML string to server and returns an Item response
- **applyMethod**
 - Execute server Method
- **applySQL**
 - Submit SQL string to server and returns an Item response

Copyright © 2014 Aras All Rights Reserved.

aras.com

Base methods Examples:

applyAML

```
Item applyAML(string AML)
```

```
Innovator inn = this.getInnovator();
string AML = "<AML><Item type='Customer' action='get' /></AML>";
Item result = inn.applyAML(AML);
```

VB.NET

```
Dim inn as Innovator = Me.getInnovator
Dim AML as String =
    "<AML><Item type='Customer' action='get' /></AML>"
Dim result as Item = inn.applyAML(AML)
```

applyMethod

```
Item applyMethod( string methodName, string body)
```

```
Innovator inn = this.getInnovator();
string body ="<itm_type>Customer</itm_type>
```

```

<name>IBM</name>";
Item res =
    inn.applyMethod ("GetItem", body);
return res;

```

VB.NET

```

Dim inn as Innovator = Me.getInnovator
Dim body as String =
    "<itm_type>Customer</itm_type>" & _
    "<name>IBM</name>"
Dim res as Item =
    inn.applyMethod("GetItem", body)

```

applySQL

```

Item applySQL(string SQL)

Innovator inn = this.getInnovator();
Item qry = inn.applySQL("select login_name from
    innovator.[user]");
return qry;

```

VB.NET

```

Dim inn As Innovator = Me.getInnovator()
Dim qry As Item =
    inn.applySQL("select login_name from innovator.[user]")
Return qry

```

Returns:

```

<Result>
    <Item><login_name>admin</login_name> </Item>
    <Item><login_name>root</login_name></Item>
    <Item><login_name>psmith</login_name></Item>
    <Item> <login_name>vadmin</login_name></Item>
</Result>

```

NOTE

The `applySQL` method is intended for execution in *server* Methods only. Attempting to execute SQL that modifies data in a client Method (by a user without administrative credentials) will result in an error.



Item methods

- Retrieve an Item
 - getItemById
 - getItemByKeyName
 - getItemInDom
- Get a New ID
 - getNewID
- Get next Sequence
 - getNextSequence
- Create new Item
 - newItem
- Create new Error Item
 - newError
- Create new Result Item
 - newResult

Copyright © 2014 Aras All Rights Reserved.

aras.com

Item method Examples:

getItemById

```
Item getItemById(string itemTypeName, string id )
```

```
Innovator inn = this.getInnovator();
string id = "02898329898912891289918299129d";
Item myItem = inn.getItemId("Customer", id);
```

VB.NET

```
Dim inn as Innovator = Me.getInnovator()
Dim id as String = "02898329898912891289918299129d"
Dim myItem as Item = inn.getItemId("Customer", id)
```

getItemByKeyName

```
Item getItemByKeyName(string itemTypeName, string keyedName)
```

```
Innovator inn = this.getInnovator();
Item myItem = inn.getItemByKeyName("Document",
    "Attached Document");
```

VB.NET

```
Dim inn as Innovator = Me.getInnovator()  
Dim myItem as Item =  
    inn.getItemByKeyedName("Document", "Attached Document")
```

getItemInDom**Item getItemInDom(XmlDocument DOM)**

```
Innovator inn = this.getInnovator();  
Item myItem = inn.getItemInDom(item.dom);
```

VB.NET

```
Dim inn as Innovator = Me.getInnovator()  
Dim myItem as Item = inn.getItemInDom(item.dom)
```

getNewID**string getNewId()**

```
string id = this.getNewID();  
return this.getInnovator().newResult(id);
```

VB.NET

```
Dim id as String = Me.getNewID()  
return Me.getInnovator().newResult(id)
```

getNextSequence**string getNextSequence(string sequence_name)**

```
Innovator inn = this.getInnovator();  
string nextseq = inn.getNextSequence("Design Request");
```

VB.NET

```
Dim inn as Innovator = Me.getInnovator()  
Dim myItem as Item = inn.getNextSequence("Design Request")
```



Utility methods

- **getI18NSessionContext**
- **newXMLDocument**
- **ScalcMD5**
- **getUserID**

Copyright © 2014 Aras All Rights Reserved.

aras.com

Utility Method Examples:

getI18NSessionContext

```
I18NSessionContext getI18NSessionContext()  
I18NSessionContext i =  
this.getInnovator().getI18NSessionContext();
```

ScalcMD5

```
static string ScalcMD5(string val)  
var md5_string = this.getInnovator().ScalcMD5("mypassword");
```

newXMLDocument

```
XMLDocument newXMLDocument()  
XmlDocument doc = this.newXMLDocument();
```

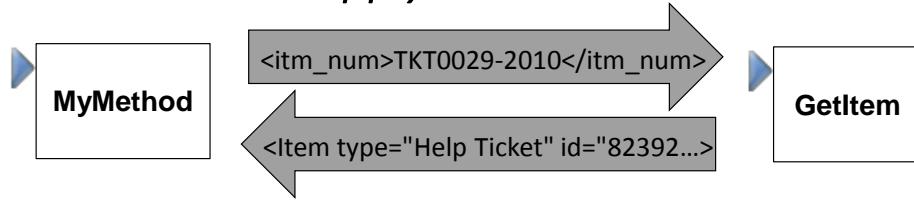
getUserID()

```
string getUserId()  
string myid = this.getInnovator().getUserID();
```

Creating Generic Methods



- Represent a piece of business logic
- Must return an Item, NewResult or NewError
- Can accept formatted XML as arguments
- Can be called from other Methods using the Innovator Class *applyMethod* method



Copyright © 2014 Aras All Rights Reserved.

aras.com

Generic Methods are useful for creating reusable modules. They can accept input and return an Item. The *applyMethod* method allows you to call one Method from another and pass parameters as shown above.

The XML argument list is parsed and the arguments and values become temporary properties of the Method being passed to a called Method. The starting Method becomes the context item in the called Method.

This allows you to access the value of the argument properties in the called Method and use them for processing.

The called Method then returns an Item back to the starting Method for inspection.



Defining Generic Methods

■ MyMethod (Calling Method)

```

01 Innovator inn = this.getInnovator();
02 string body ="<itm_type>Design Request</itm_type>
03     <itm_num>DR-000010</itm_num>";
04 Item res =
05 inn.applyMethod ("GetItem", body);
06 return res;

```

Define arguments

Call Method

■ GetItem (Called Method)

```

01 string item_number = this.getProperty("itm_num");
02 string item_type = this.getProperty("itm_type");
03 Item myItem = this newItem(item_type, "get");
04 myItem.setProperty("_item_number", item_number);
05 myItem = myItem.apply();
06 return myItem;

```

Passed Context Item

Return Item to Calling Method

Calling Method

The calling method creates a parameter string that contains two arguments named *itm_type* and *itm_num*(line 2). The method then calls the second Method named GetItem using the applyMethod method.

VB.NET

```

Dim inn As Innovator = Me.getInnovator()
Dim body As String ="<itm_type>Design Request</itm_type>" & _
    "<itm_num>DR-000010</itm_num>"
Dim res As Item = inn.applyMethod ("GetItem", body)
Return res

```

Called Method

The context Item (ME or this) is MyMethod (calling Method) in this example. In addition, the called method accesses the two parameters using the getProperty method and uses them to return an Item from the server.

VB.NET

```

Dim item_number As String = Me.getProperty("itm_num")
Dim item_type As String = Me.getProperty("itm_type")
Dim myItem As Item = Me.newItem(item_type, "get")
myItem.setProperty("item_number", item_number)
myItem = myItem.apply()
Return myItem

```

Defining Generic Methods



■ Calculated Ship Date – Server Method

```

string sonum = this.getProperty("order");
string days = this.getProperty("days");
//.NET Functions
int daynum = Convert.ToInt16(days);
DateTime date = DateTime.Now.AddDays(daynum);
string strdate = date.ToString("yyyy-MM-dd");

Item itm = this newItem("Sales Order", "edit");
itm.setAttribute("where",
    "[sales_order].so_number='"
    + sonum + "'");
itm.setProperty("ship_date", strdate);
return itm.apply();

```

Retrieve arguments

Return new result

Copyright © 2014 Aras All Rights Reserved.

aras.com

This example shows how to create a server Method named **CalculateShipDate** that will return a Sales Order with an assigned ship date that has been calculated by the Method. This method can then be called from a client Method (next page). Note that the Method expects the parameters named "days" and "order".

VB.NET

```

Dim sonum As String = Me.getProperty("order")
Dim days As String = Me.getProperty("days")

'.NET Functions
Dim daynum As Integer = Convert.ToInt16(days)
Dim [date] As DateTime = DateTime.Now.AddDays(daynum)
Dim strdate As String = [date].ToString("yyyy-MM-dd")

Dim itm As Item = Me.newItem("Sales Order", "edit")
itm.setAttribute("where", "[sales_order].so_number='"
    & sonum & "'")
itm.setProperty("ship_date", strdate)
Return itm.apply()

```



Defining Generic Methods

■ Call Server Method from Client

```
01 var inn = this.getInnovator();  
02 var body = "<days>10</days><order>SO-0005</order>";  
03 var result =  
04 inn.applyMethod("CalculateShipDate", body);  
05 return result;
```

A client (or server) Method can then be used to call the *CalculateShipDate* server Method to access server data.

Using Generic Methods as Actions



- Server Method is also available as AML Action

```
<AML>
<Item type="Sales Order" action="CalculateShipDate">
    <order>SO-0005</order>
    <days>10</days>
</Item>
</AML>
```

A diagram illustrating the AML code. A callout box labeled "Server Method" points to the "action" attribute of the "Item" tag. Another callout box labeled "Property arguments" points to the "days" element within the "Item" tag.

Copyright © 2014 Aras All Rights Reserved.

aras.com

As discussed earlier in the course – an AML statement contains an "action" attribute that instructs the server on what operation should take place (add, edit, delete, etc.). There are a collection of built in actions available. You can also create your own custom action by creating a server Method.

This allows you to easily extend the AML actions available to other developers or users.

Using Generic Methods as Actions

To call the *CalculateShipDate* server method the action attribute is used to supply the server Method name. The property tags become arguments to the server Method and can be obtained by using the *getProperty* method of the Item class.



Passing an Attribute From AML

- Additional attributes added to the calling AML expression can be retrieved in the called Method
- AML Alternative:

```
<AML>  
  <Item type="Sales Order" action="CalculateShipDate"  
        order="SO-0005" days="10">  
  </Item>  
</AML>
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

You can also pass additional parameters to a custom AML action by adding one or more attributes that then capturing the values using the `getAttribute` method `Item` class method.

To accept the attributes displayed in the AML example above two lines of the server Method would be changed to:

C#:

```
string sonum = this.getAttribute("order");  
string days = this.getAttribute("days");
```

VB.NET

```
Dim sonum As String = Me.getAttribute("order")  
Dim days As String = Me.getAttribute("days")
```



Summary

In this unit you learned about the various IOM methods available in the Innovator class.

You should now be able to:

- ✓ Identify and Classify IOM Innovator Class methods
- ✓ Create Generic Methods
- ✓ Create AML Action Methods



Lab Exercise

Goal:

Be able to use and locate the basic IOM methods available in the Innovator class.

Scenario:

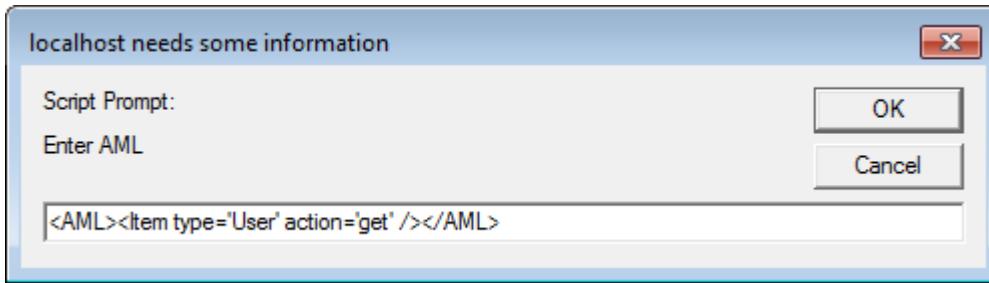
In this exercise, you will create Methods that use the IOM Innovator Class.

Steps:

1. Create a client Method that allows a user to enter an AML statement into a prompt dialog box - using the JavaScript prompt method:

```
var aml = prompt("message", "defaultvalue");
```

Use the applyAML Innovator class method to process the AML request and return the AML response back to the client. Attach the Method to a Generic Client Action where the output Target is "Window" to allow users to run the utility and view the results.



Optional:

To make the output of the Action easier to read you can apply a default XML style sheet to the AML result variable before returning it using the following example. (Note: you will only be able to test the Method by running it from the Generic Client Action menu you created above. The *Run Client Method* action in the Method editor cannot be used to test this code.)

```
var ss="http://localhost/Innovator10/Client/styles/default.xsl";
return your_AMLresult.applyStylesheet(ss, "URL");
```

2. Create a Server Method named **Server-Calculator** that calculates the total cost of all Parts in the system using a SQL statement. You can quickly get the sum of a column in a database select statement using:

```
select SUM(cost) as value from innovator.part
```

Unit test the Method using the *Run Server Method* Action in the Method editor. Your result should resemble the following (the actual value may differ):

```
<Item>
    <value>5544.00</value>
</Item>
```

- Now make the program more generic by allowing someone to call your Method and pass the column_name, table_name and SQL aggregate function name as property parameters to the SQL statement.

First, use the getProperty Item class method to get the value of "column_name", "table_name" and "function" and assign them to string variables.

Then create a SQL select string that uses the value of these property arguments:

C#

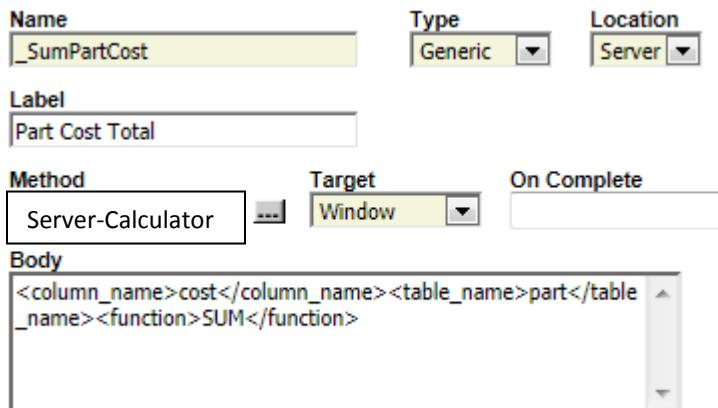
```
string SQL = "select " + function + "(" + column + ")" as " +
function + " from innovator." + table;
```

VB .NET

```
Dim SQL As String = "select " & function & "(" & column & ")" _ 
as " & function & " from innovator." & table
```

Apply the SQL using the Innovator class applySQL method.

Attach this generic Method to an Action and supply the parameter values in the Body field of the Action as shown below. Each tag value supplied in the Body field is passed as a property value to the generic Method.



Create one action to SUM the Part cost and another Action to AVG the cost. Attach the Actions to the Part ItemType.

Notice how you can create one Method that performs different operations based on the *Body* expression.

- Try calling the **Server-Calculator** Method as an AML request using AML Studio. You will need to set the appropriate property tags in the AML expression to calculate a result.

This page intentionally left blank.



Unit 7 Exploring the Item Class

Overview: In this unit you will learn how to use the various methods available in the Item Class. These foundation methods allow you to set and get property and attribute values, define relationships, iterate through Item collections and test for various Item conditions.

Objectives: ✓ Classifying IOM Item method types:

- ✓ Base
- ✓ Boolean
- ✓ Attribute
- ✓ Property
- ✓ File
- ✓ Relationship
- ✓ Collection
- ✓ Logical
- ✓ New
- ✓ Error



Base methods

- **apply**
 - Submit AML request to server using Context Item and returns new Item
- **loadAML**
 - Load AML into Item dom property – does not return a new Item
- **clone**
 - Creates a copy of Item (with new id)
 - Can optionally clone relationships

Base method Examples:

apply

```
Item apply()  
Item apply(string actionPerformed)
```

The apply method can accept a string argument which defines a custom AML action to run on the item in context as it is submitted to the server. You will use this method signature later in the course.

```
Item resultItm = this.apply("MyCustom Action");
```

VB.NET

```
Dim resultItm as Item – Me.apply("MyCustom Action")
```

loadAML

```
void loadAML(string AML)
```

The loadAML method sets the AML string to the item DOM but does not submit it to the server. You must use the apply() method to submit the item after the AML is loaded.

```
Item itm = this newItem();
String AML =
    @"<Item type='Customer' action='get'></Item>";
try {itm.loadAML(AML);}
catch (System.Xml.XmlException ex) {
    return this;
}
itm = itm.apply();
return itm;
```

VB.NET

```
Dim itm As Item = Me newItem()
Dim AML As String = "<Item type='Customer' action='get'></Item>"
Try
    itm.loadAML(AML)
Catch ex As System.Xml.XmlException
    Return Me
End Try
itm = itm.apply()
Return itm
```

clone

```
Item clone( bool cloneRelationships)
```

If cloneRelationships is true the relationships must be fetched in the item before it is copied.

```
Item itm = this newItem("Customer" , "get");
itm.setProperty("name", "EDA");
itm = itm.apply();

Item i_copy = itm.clone(false);
i_copy.setProperty("name", "EDA - Copy");
return i_copy.apply();
```

VB.NET

```
Dim itm as Item = Me newItem("Customer", "get")
itm. setProperty("name", "EDA")
itm = itm.apply()

Dim i_copy as Item = itm.clone(false)
i_copy.setProperty("name", "EDA - Copy")
return i_copy.apply()
```



Boolean methods

- isError
- isCollection
- isEmpty
- isLogical
- isNew
- isRoot

Copyright © 2014 Aras All Rights Reserved.

aras.com

Boolean method Examples:

```
string sResult = "The following is true about this item: ";
Item itm = this newItem("User", "get");
itm = itm.apply();
if (itm.isError()) {sResult += "Error Item";}
if (itm.isCollection()) {sResult += "Item Collection";}
if (itm.isEmpty()) {sResult += "Empty Item";}
if (itm.isNew()) {sResult += "New Item";}
if (itm.isRoot()) {sResult += "Root Item";}
return this.getInnovator().newResult(sResult);
```

VB.NET

```
Dim itm As Item = Me newItem("User", "get")
itm = itm.apply()
If itm.isError() Then sResult += "Error Item" End If
If itm.isCollection() Then sResult += "Item Collection" End If
If itm.isEmpty() Then sResult += "Empty Item" End If
If itm.isNew() Then sResult += "New Item" End If
If itm.isRoot() Then sResult += "Root Item" End If
Return Me.getInnovator().newResult(sResult)
```

Attribute methods



- `getAttribute`
- `setAttribute`

- `getID`
- `setID`
- `setNewID`

- `getType`
- `setType`
- `getAction`
- `setAction`

- `removeAttribute`

Copyright © 2014 Aras All Rights Reserved.

aras.com

Attribute method Example

```
Item itm = this newItem();
itm.setType("Customer");
itm.setAction("get");
itm.setAttribute("select", "name");
itm.setProperty("name", "EDA");
itm = itm.apply();
string id = itm.getID();
string type = itm.getType();
return this.getInnovator().newResult("ID: " + id + " - " + "Type: "
+ type);
```

VB.NET

```
Dim itm as Item = Me newItem()
itm.setType("Customer")
itm.setAction("get")
itm.setAttribute("select", "name")
itm.setProperty("name", "EDA")
itm = itm.apply()
Dim id as String = itm.getID()
Dim type as String = itm.getType()
Return _
Me.getInnovator().newResult("ID: " + id + " - " + "Type: " + type)
```



Property methods

- `getProperty`
- `setProperty`
- `removeProperty`

- `getPropertyAttribute`
- `setPropertyAttribute`
- `removePropertyAttribute`

- `getProperties`
- `setProperties`
- `createProperties`

- `getPropertyItem`
- `setPropertyItem`
- `createPropertyItem`

- `setPropertyCondition`
- `getPropertyCondition`

- `fetchDefaultPropertyValue`

Copyright © 2014 Aras All Rights Reserved.

aras.com

Property methods

The `getProperty` method offers two argument signatures:

```
getProperty(property_name);
getProperty(property_name, default_value);
```

The second argument signature allows you to specify a default value to be returned from the method if the property is not defined in the Item.

Examples:

```
string s = this.getProperty("details", "");
Dim s as String = Me.getProperty("details", "")
```

The value of *s* would be equal to "" if the `details` property was not provided in the current Item. Using the default value makes it easier for testing purposes in a condition statement.

```
if (s == "") {property not defined logic}
If (s=="") property not defined logic End If
```

Property method Examples:

setProperty

```
Item item = this newItem("Design Request", "get");
//Set Cost Value to 50
item.setProperty("_cost", "50");
```

setPropertyAttribute/setPropertyCondition

```
//Add condition greater than (in this example > 50)
item.setPropertyCondition("_cost", "gt");

//Set a property value to NULL
item.setPropertyAttribute("locked_by_id", "is_null", "1");

item = item.apply();
```

getPropertyAttribute

```
String state =
    item.getPropertyAttribute("current_state", "keyed_name");
```

getPropertyItem

```
Item customer = item.getPropertyItem("_customer");
```

getProperty

```
String id = customer.getProperty("id", "");
return this.getInnovator().newResult("ID: " + id);
```

fetchDefaultPropertyValues

Sets the default values of all properties on the item and returns the updated item. To overwrite current property values set the first argument to true.

```
item = item.fetchDefaultPropertyValues(true);
```

VB.NET

```
Dim item as Item = Me newItem("Design Request", "get")
item.setProperty("_cost", "50")
item.setPropertyAttribute("_cost", "condition", "gt")
item.setPropertyAttribute("locked_by_id", "is_null", "1")

item = item.apply();

Dim state as String =
    item.getPropertyAttribute("current_state", "keyed_name")

Dim customer as Item = item.getPropertyItem("_customer")
Dim id as String = customer.getProperty("id", "")

Return Me.getInnovator().newResult("ID: " + id)
```



File methods

- checkout
- attachPhysicalFile

File method Example:

Only used for ItemType "File". The checkout method will copy a file from the vault to a targeted directory. To prevent changes by other users, you should also lock the file using the lockItem method discussed later in this unit. The specified destination directory will be created if it does not exist. If the file already exists at the destination, it is overwritten without warning.

Checkout

```
Innovator inn = this.getInnovator();
Item qry = inn newItem("File", "get");
qry.setProperty("filename", "Test.txt");

Item file = qry.apply();
if (file.getItemCount() != 1) {
    return inn.newError("File not found");
}
Item result = file.checkout( @"c:\temp" );
if( result.isError() ) {
    return inn.newError("Cannot checkout file");
}
string filename = result.getProperty("checkedout_path");
file.lockItem();
return inn.newResult("File checked out to: " +
    filename);
```

VB.NET

```

Dim inn As Innovator = Me.getInnovator()
Dim qry As Item = inn newItem("File", "get")
qry.setProperty("filename", "Test.txt")

Dim file As Item = qry.apply()
If (file.isError()) Then
    Return inn.newError("File not found")
End If
Dim result As Item = file.checkout( "c:\temp" )
If( result.isError() )
    Return inn.newError("Cannot checkout file")
End If
Dim filename As String = result.getProperty("checkedout_path")
file.lockItem()
Return
    inn.newResult("File checked out to: " + checkout_path)

```

Checkin

```

var itm = this newItem("File", "get");
itm.setProperty("filename", "Test.txt");
itm = itm.apply();
//item needs to be locked to attach file
itm = itm.lockItem();
itm.attachPhysicalFile("c:\\temp\\Test.txt");
//now save the changes to File Item which will also unlock item
itm.setAction("update");
itm = itm.apply();
return itm;

```



Relationship methods

- addRelationship
- createRelationship
- removeRelationship
- createRelatedItem
- setRelatedItem
- getRelationships
- getRelatedItem
- getRelatedItemID
- getParentItem
- fetchRelationships

Relationship methods allow you define a Relationship as part of a query and also return Relationships or related items.

Remember, that these methods are generating AML that will be submitted to the server for processing.

Note

The “fetch” methods in the IOM are designed to access the database directly rather than construct an AML request in memory prior to processing. The fetchRelationships method will access the database directly to assign the Relationships to the current context Item. The next page demonstrates this method.

Using fetchRelationships



- Retrieves current relationships from server

```

01 Item itm = this newItem("Design Request", "get");
02 itm.setProperty("item_number", "DR-000010");
03 itm = itm.apply();

//Go to server to fetch relationships
04 itm.fetchRelationships("Design Request Document");

05 return itm;

```

Copyright © 2014 Aras All Rights Reserved.

aras.com

Note

In this example the “fetch” has been done after the apply – fetch retrieves from data from the server while the other Relationships methods work with the current DOM. In the initial apply no relationship information will be included in the returned AML for the Design Request Item.

```

<Result>
  <Item type="Design Request" ... >
  ...
  </Item>
<Result>

```

Once the fetchRelationships method is invoked, the Design Request Document Relationships will then include the relationship information.

```

<Result>
  <Item type="Design Request" ... >
  ...
    <Relationships>
      <Item type=" Design Request Document" ... />
    </Relationships>
  </Item>
<Result>

```

VB.NET

```
Dim itm as Item = Me.newItem("Design Request", "get")
itm.setProperty("item_number", "DR-000010")
itm = itm.apply()
//Go to server to fetch relationships
itm.fetchRelationships("Design Request Document")
Return itm
```

Note

The *fetchRelationships* method changes the current item (and automatically does an apply to the server). In the example above the variable *itm* is changed by the *fetchRelationships* method. This is not true for the *apply* method which requires that you return an *itm* to inspect changes.

Retrieving Items using Relationships



```

01 Item itm = this newItem("Design Request", "get");
02 Item relationship =
03     this newItem("Design Request Document", "get");
04 Item relitm = this newItem("Document", "get");
05 relitm.setProperty("item_number", "SPEC-00001");
06 relationship.setRelatedItem(relitm);
07 itm.addRelationship(relationship);
08 itm = itm.apply();
09 return itm;

```

Copyright © 2014 Aras

All Rights Reserved.

aras.com

The method above will generate the AML shown below. It is important to understand how the IOM constructs AML requests that need to use Relationships:

```

<Item isNew = "1" isTemp = "1" type = "Design Request" action = "get">
    <Relationships>
        <Item isNew = "1" isTemp = "1" type = "Design Request Document" action = "get">
            <related_id>
                <Item isNew = "1" isTemp = "1" type = "Document" action = "get">
                    <item_number>SPEC-00001</item_number>
                </Item>
            </related_id>
        </Item>
    </Relationships>
</Item>

```

VB.NET

```

Dim itm as Item = Me newItem("Design Request", "get")
Item relationship = Me newItem("Design Request Document", "get")
Item relitm = Me newItem("Document", "get")
relitm.setProperty("item_number", " SPEC-00001")
relationship.setRelatedItem(relitm)
itm.addRelationship(relationship)
itm = itm.apply()
Return itm;

```



Adding New Relationships

```

01 Item itm = this newItem("Design Request", "edit");
02 itm.setAttribute("where",
03     "design_request._item_number='DR-000020'");
04 Item relationship =
05     this newItem("Design Request Document", "add");
06 Item relitm = this newItem("Document", "get");
07 relitm.setProperty("item_number",
08     "SPEC-00001");
09 relationship.setRelatedItem(relitm);
10 itm.addRelationship(relationship);
11 itm = itm.apply();
12 return itm;

```

Copyright © 2014 Aras All Rights Reserved.

aras.com

This example demonstrates how to add a new Relationship to an Item.

In this example, a new Document is related to the Design Request using the Design Request Document relationship.

Can you write down what the generated AML would look like that is applied to the server?

VB.NET

```

Dim itm As Item = Me newItem("Design Request", "edit")
itm.setAttribute("where",
    "design_request._item_number='DR-000020'")
Dim relationship As Item =
    Me newItem("Design Request Document", "add")
Dim relitm As Item = Me newItem("Document", "get")
relitm.setProperty("item_number", "SPEC-00001")
relationship.setRelatedItem(relitm)
itm.addRelationship(relationship)
itm = itm.apply()
Return itm

```

Retrieving Related Items



```

01 Item itm = this newItem("Design Request", "get");
02 itm.setProperty("_item_number", "DR-000010");
03 itm = itm.apply();
04
05 itm.fetchRelationships("Design Request Document");
06 Item relationships =
07     itm.getRelationships("Design Request Document");
08 int count = relationships.getItemCount();
09 for (int i=0; i<count; i++) {
10     Item doc_relationship =
11         relationships.getItemByIndex(i);
12     Item doc_itm =
13         doc_relationship.getRelatedItem();
14 }
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

This example demonstrates how to retrieve related items from a source item.

In the example, after the “fetchRelationships” has been applied to the Design Request item the getRelationships method will return the Relationships on the Design Request (Design Request Document).

A for loop is then used to cycle through each Relationship and the getRelatedItem method is used to get the current Item and assign it to rel_result.

Further processing on each Item could then take place as required.

The next page describes how to use Collection methods to process a list of Items.

VB.NET

```

Dim relationships As Item =
    itm.getRelationships("Design Request Document")
Dim count As Integer = relationships.getItemCount()
Dim i As Integer
For i = 0 To count - 1
    Dim doc_relationship As Item =
        relationships.getItemByIndex(i)
    Dim doc_itm As Item =
        doc_relationship.getRelatedItem()
Next i
```



Collection methods

- **getItemCount**
- **getItemByIndex**
- **getItemsByXPath**
- **appendItem**
- **removeItem**

Collection methods Example:

Collection methods allow you to process a collection of Items. This example builds a comma delimited list of Users:

```
Innovator inn = this.getInnovator();
Item itm = this newItem("User", "get");
Item results = itm.apply();

int count = results.getItemCount();
string ids_str = "";

for (int i=0; i<count; i++) {
    Item itm_idx = results.getItemByIndex(i);
    string id = itm_idx.getID();
    ids_str += id + ",";
}
return inn.newResult(ids_str);
```

VB.NET

```
Dim inn as Innovator = this.getInnovator()
Dim itm as Item = this newItem("User", "get")
Dim results as Item = itm.apply()
```

```

Dim count as Integer = results.getItemCount()
Dim ids_str as String = ""
Dim i as Integer

For i = 0 to count
    Dim itm_idx as Item = results.getItemByIndex(i)
    Dim id as String = itm_idx.getID()
    ids_str += id + ","
Next i

Return inn.newResult(ids_str);

```

Using XPath

XPath can be used to search for values in the DOM and return specific data. In this example, a list of Design Requests is requested from the server. XPath is then used to locate a specific Item and assign it to a variable

```

Item qry = this newItem("Design Request", "get");
qry = qry.apply();
Item itm_find =
qry.getItemsByXPath("//Item[item_number='DR-000010']");
return this.getInnovator().newResult(itm_find.getID());

```

VB.NET

```

Dim qry as Item = Me newItem("Design Request", "get")
qry = qry.apply()
Item itm_find =
qry.getItemsByXPath("//Item[item_number='DR-000010']")
Return Me.getInnovator().newResult(itm_find.getID())

```

You will use XPath later in the course to set columns in a Configurable Grid. We will discuss the basics of the language in that unit if you are not familiar with the syntax.

Append/Remove Item

Append Item can be used to build a collection in a Method. Remove Item will remove an Item from the collection.

```

Item itm1 = this newItem("Customer", "get");
itm1.setProperty("name", "IBM");
Item itm2 = this newItem("Customer", "get");
itm2.setProperty("name", "GHC");
itm1 = itm1.apply();
itm2 = itm2.apply();
itm1.appendItem(itm2); //Appends Item to form collection
itm1.removeItem(itm2); //Removes the Item from the collection
return itm1;

```

VB.NET

```
Dim itm1 as Item = Me.newItem("Customer", "get")
itm1.setProperty("name", "IBM");
Dim itm2 as Item = Me.newItem("Customer", "get")
itm2.setProperty("name", "GHC")
itm1 = itm1.apply()
itm2 = itm2.apply()
itm1.appendItem(itm2)  'Appends Item to form collection
itm1.removeItem(itm2)  'Removes the Item from the collection
return itm1
```

Logical methods



- newOR
- newAND
- newNOT
- removeLogical
- getLogicalChildren

Copyright © 2014 Aras All Rights Reserved.

aras.com

Logical methods allow you to create Boolean queries using AND, OR or NOT. The appropriate AML tags are generated based on how the methods are used.

In this example, a query is built to get all Design Requests that have a status of “Start” AND patent required = true— OR – have a status of “In Review” and a patent required of false.

```
Item qry = this newItem("Design Request", "get");
```

```
Item logicItm = qry.newOR();
```

```
Item logicAND1 = logicItm.newAND();
logicAND1.setProperty("state", "Start");
logicAND1.setProperty("_patent_required", "1");
```

```
Item logicAND2 = logicItm.newAND();
logicAND2.setProperty("state", "In Review");
logicAND2.setProperty("_patent_required", "0");
```

```
Item retItm = qry.apply();
```

VB.NET

```
Dim qry as Item = Me newItem("Design Request", "get")
```

```
Dim logicItm As Item = qry.newOR()
```

Developing Solutions

```
Dim logicAND1 As Item = logicItm.newAND()
logicAND1.setProperty("state", "Start")
logicAND1.setProperty("_patent_required ", "1")

Dim logicAND2 As Item = logicItm.newAND()
logicAND2.setProperty("state", "In Review")
logicAND2.setProperty("_patent_required ", "0")

Dim retItm as Item = qry.apply();
```

Which creates the following AML request:

```
<Item type="Design Request" action="get">
    <or>
        <and>
            <state>Start</state>
            <_patent_required>1</priority>
        </and>
        <and>
            <state>In Review</state>
            <_patent_required>0</priority>
        </and>
    </or>
</Item>
```

To modify the request above you can access the logical children of the Design Request query and remove the “and” logic:

```
Item lchildren = qry.getLogicalChildren();

for( int i = 0; i < lchildren.getItemCount(); i++ )
{
    Item lchild = lchildren.getItemByIndex(i);
    qry.removeLogical(lchild);
}
```

VB.NET

```
Dim lchildren As Item = qry.getLogicalChildren()

For i As Integer = 0 To lchildren.getItemCount() - 1
    Dim lchild As Item = lchildren.getItemByIndex(i)
    qry.removeLogical(lchild)
```

Next

New methods



- newItem
- newXMLDocument

Copyright © 2014 Aras All Rights Reserved.

aras.com

The newItem method is one of the most common Item methods used as you have seen in previous examples. You can create an “empty” Item (no parameters), an Item of a specified type with no action assigned (one parameter) or an Item of a specific type with the action attribute set.

```
Item item = this.newItem("Design Request", "delete");
```

VB.NET

```
Dim item as Item = Me.newItem("Design Request", "delete")
```

The newXMLDocument method can be used to create a new DOM which can then be parsed using standard XML .NET methods.

```
 XmlDocument doc = this.newXMLDocument();  
XmlElement root = doc.CreateElement("AML");  
XmlElement item = doc.CreateElement("Item");  
item.SetAttribute("type", "Design Request");  
item.SetAttribute("action", "get");  
doc.AppendChild(root);  
root.AppendChild(item);  
Item item = this.newItem();  
itm.loadAML(doc.OuterXml);  
itm = itm.apply();
```



Error methods

- `getErrorString`
- `setErrorString`

- `getErrorCode`
- `setErrorCode`

- `getErrorDetail`
- `setErrorDetail`



Copyright © 2014 Aras

All Rights Reserved.

aras.com

A series of methods are available to configure a custom error message. You must first define a new `Error` object to set the appropriate attributes:

```
Innovator inn = this.getInnovator();
Item err = inn.newError("");

err.setErrorCode("-1");
err.setErrorDetail("My custom message");
err.setErrorSource("My program");
err.setErrorString("Custom Error Message");

return err;
```

VB.NET

```
Dim inn as Innovator = this.getInnovator()

Dim err as Item = inn.newError ""

err.setErrorCode("-1")
err.setErrorDetail("My custom message")
err.setErrorSource("My program")
err.setErrorString("Custom Error Message")

Return err
```

Extended methods



- Locking
 - lockItem
 - unlockItem
 - fetchLockStatus
- Lifecycle
 - promote
- Workflow
 - instantiateWorkflow
- Email
 - email
- Item to String
 - ToString

Copyright © 2014 Aras All Rights Reserved.

aras.com

Extended methods

These methods are described as “extended” as they do not fit in any other “category” described previously.

Workflow Example

The example below shows how a workflow process can be created for a controlled item using the instantiateWorkflow method. Here it is assumed that the Method is being invoked from the controlled Item.

```
Item item = this newItem("Workflow Map", "get");
item.setAttribute("select", "id");
item.setProperty("name", "Design Request");
item = item.apply();
string wf_id = item.getID();
Item wf_process = this.instantiateWorkflow(wf_id);
return wf_process.apply("startWorkflow");
```

VB.NET

```
Dim item as Item = this newItem("Workflow Map", "get")
item.setAttribute("select", "id")
item.setProperty("name", "Design Request")
item = item.apply()
Dim wf_id as String = item.getID()
Dim wf_process as Item = this.instantiateWorkflow(wf_id)
```

```
Return Me.apply("startWorkflow")
```

Note

Once a workflow has been instantiated, it must be started using the action *startWorkflow*. To invoke an action, use the Item *apply* method discussed earlier in this unit. You can also cancel a running workflow using the *cancelWorkflow* or close a workflow using the *closeWorkflow* action.

E-mail Messages

To send an e-mail message, you must first create the message and insure that the From User and To User identities have been specified and that all users involved have valid e-mail addresses in their User Profiles.

```
Innovator inn = this.getInnovator();
Item email_msg = this newItem("Email Message", "get");
email_msg.setProperty("name", "Service Reminder");
email_msg = email_msg.apply();
//Get admin Identity
Item iden = this newItem("Identity", "get");
iden.setProperty("name", "Innovator Admin");
iden = iden.apply();
try {
    bool result = this.email(email_msg, iden);
}
catch (Exception e) {
    return inn.newError(e.Message);
}
return this;
```

VB.NET

```
Dim inn as Innovator = Me.getInnovator()
Dim email_msg as Item = Me newItem("Email Message", "get")
email_msg.setProperty("name", "Service Reminder")
email_msg = email_msg.apply()
//Get admin Identity
Dim iden as Item = Me newItem("Identity", "get")
iden.setProperty("name", "Innovator Admin")
iden = iden.apply()
Try
    Dim result as Boolean = Me.email(email_msg, iden);
Catch (Exception e)
    return inn.newError(e.Message);
End Try
return Me;
```



Summary

In this unit you learned about the IOM Item class methods available to create, edit and remove Items from the database.

You should now be able to:

- ✓ Identify and Classify IOM Item Class methods



Lab Exercise

Goal:

Be able use IOM Item class methods to create and manipulate Items in a solution.

Scenario:

In this exercise, you will use Item class methods to create a Part cost calculator that can be called from an existing Sales Order to show the total of all Part line items on an order.

Create Server Calculate Method

Create a Server Method that will total up the cost of all of the Part line items in a Sales Order. The Method will need to establish a line item cost first (quantity * part unit cost) and then sum all of the line items for a total cost. Attach the Method to an Item Action to allow a user to see the result in a target Window.

TIPS

The steps for this Method follow the same pattern discussed in the "Retrieving Related Items" example discussed in this unit. The value of quantity is stored in a property on the relationship item (between the Sales Order and the related Part item).

1. Create a Server Method named **Server-CalculateCost**. This server Method will be used as an Item Action so what does the context Item (this or Me) represent? _____
2. Retrieve an Innovator instance using the `getInnovator` IOM method and store it in a variable named `inn`. You will use this instance later in the program to return a new Result item.
3. Declare the following variables that will be used to calculate cost and set them to 0:

Data Type C#	Data Type VB	Variable Name
decimal	Decimal	total_cost
decimal	Decimal	unit_cost
decimal	Decimal	line_cost
int	Integer	quantity
int	Integer	count

4. Use the `fetchRelationships` IOM method to retrieve the "Sales Order Part" relationships from the current Sales Order item (`this`) .
5. Now use the `getRelationships` IOM method to retrieve the Relationship collection from the current item (`this`) and store this information in an Item variable named `relationships`.
6. Get the count of relationships using the `getItemCount` method and store it in `count`.
7. Create a for-loop to iterate through each item in `relationships`. (See the example shown in Retrieving Related Items in this unit if you need help with the syntax).

8. Within the for-loop, access and store the currently indexed relationships item in an Item variable named partRel using the getItemByIndex method.

9. Get the value of the *quantity* property from the current partRel relationship item using getProperty, convert the returned string to an integer and store in quantity:

```
quantity= Convert.ToInt16(partRel.getProperty("quantity", "0"));
```

10. Access the related Item from the current partRel Relationship using the getRelatedItem method and store it in an item variable named part.

11. Get the value of the *cost* property from the current related part item using getProperty, convert it to a decimal variable and store in unit_cost:

```
unit_cost = Convert.ToDecimal(part.getProperty("cost", "0"))
```

12. Multiply the quantity * unit_cost and store in line_cost.

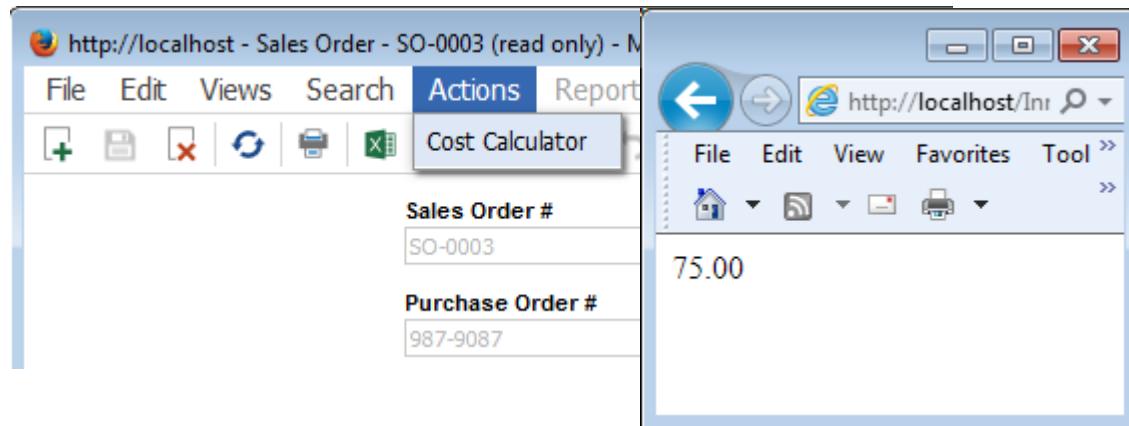
13. Add the line_cost to total_cost.

14. When the for-loop completes, convert total_cost to a string and return it using the Innovator newResult method:

```
inn.newResult(totalcost.ToString())
```

Create Action

15. Create a new Item Action named **Calculate Cost** that calls the **Server-CalculateCost** Method and Targets a Window. Attach the Action to the Sales Order ItemType.



This page intentionally left blank.



Unit 8 Creating Server Event Methods

Overview: In this unit you will learn how to build and assign server methods to customize and extend the standard behavior of the system. This includes Item validation, setting Item property values as well as workflow dynamic routing and assignment. In addition, you will learn how to capture login and logout information using a system event.

- Objectives:**
- ✓ Reviewing Server Events
 - ✓ Developing ItemType Validation Routines
 - ✓ Developing Workflow Validation Routines
 - ✓ Creating Dynamic Workflow Routes
 - ✓ Creating Workflow Dynamic Assignments



Reviewing Server Events

- Allow custom logic in a Server Method to be applied:
 - Before standard logic (OnBefore*****)
 - After standard logic (OnAfter*****)
 - Instead of standard logic (On*****)
- Server Event Methods access a Context Item
 - Before Events have access to AML Request
 - After Events have access to AML Response
 - On Events must create and return a valid AML Response

Reviewing Server Events

Server Events enable you to assign a server Method that prevents, extends or replaces the standard behavior in an Aras Innovator solution.

"OnBefore" events are typically used for validation to prevent invalid data from being saved to the database.

"OnAfter" events can be used to extend the standard behavior and perform additional operations once the AML response has been sent back to the client.

"On" events are used to completely replace the standard behavior and require that you create an AML response Item that is acceptable to the client. On event Methods typically take the most work to employ in a solution. Note that when you assign a Method to an On event that behavior is disabled by Aras Innovator and you are responsible for replacing out the task behavior.

ItemType Server Events



- Assigned to an ItemType
- Allow prevention, validation and replacement of Item events including:

- Add
- Update
- Delete
- Get
- Copy
- Lock
- Unlock
- Version
- Method

Name	Method Type	Ver	execution_allow...	Comments	Event	Sort Order
onBeforeDeleteRequest	CSharp	1	World		onBeforeDelete	128

Copyright © 2014 Aras

All Rights Reserved.

aras.com

ItemType Server Events

The following events have been defined to prevent, extend or override standard Item behaviors:

ItemType Events

Before	After	Instead Of
OnBeforeAdd	OnAfterAdd	OnAdd
OnBeforeUpdate	OnAfterUpdate	OnUpdate
OnBefore Delete	OnAfter Delete	OnDelete
OnBeforeGet	OnAfterGet	OnGet
OnBeforeCopy	OnAfterCopy	OnCopy
OnBeforeLock	OnAfterLock	OnLock
OnBeforeUnlock	OnAfterUnlock	OnUnlock
OnBeforeVersion	OnAfterVersion	OnVersion
OnBeforeMethod	OnAfterMethod	OnMethod

The GetKeyName Event is also available for each item.

LifeCycle Transition Events



- Assigned to LifeCycle Map

A screenshot of a software interface titled "LifeCycle Transition Events". At the top, there are tabs for "State" and "Transition", with "Transition" being the active tab. Below the tabs, there is a section labeled "Role" containing a dropdown menu with the option "Support". Underneath this is a section labeled "Server Methods" with two rows: "Pre" and "Post". The "Pre" row contains a dropdown menu with the option "cfg_Server_CheckProble". There are also three small square icons with arrows pointing outwards in the bottom right corner of the interface area.

Copyright © 2014 Aras All Rights Reserved.

aras.com

LifeCycle Transitions

A Server Method can be used before and after a lifecycle state is changed to prevent a promotion (Pre) or extend the behaviors (Post) of a completed promotion.

Workflow Events



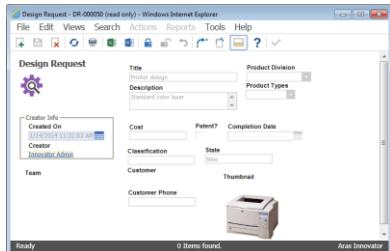
- Workflow Activity:
 - OnActivate
 - OnAssign
 - OnRefuse
 - OnDelegate
 - OnVote
 - OnRemind
 - OnDue
 - OnEscalate
 - OnClose
- Workflow Path
 - Pre Method
 - Post Method

Workflow Events

Workflow Events are enabled for each Workflow Activity as well as each Path.

Workflow Events are useful for performing Item validation before an Activity can proceed in a workflow. They are also used to perform Workflow Dynamic Routing (decided which path to take at runtime) and Workflow Dynamic Assignments (creating Activity assignments at runtime).

Developing ItemType Validation



Design Request

OnBeforeAdd
OnBeforeUpdate



Innovator Server

Copyright © 2014 Aras All Rights Reserved.

aras.com

A common use of server Methods is in Item validation to prevent invalid data from being inserted into the database based on business rules.

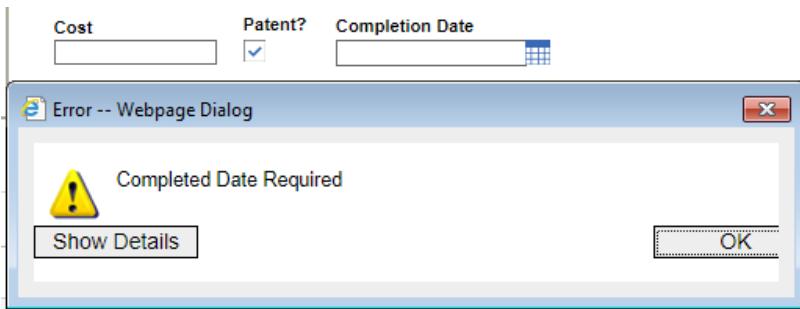
The onBeforeAdd and onBeforeUpdate events are fired each time an “add”, “create”, “merge” or “edit” action is received by the server. You can then review the data being sent to the server and prevent acceptance by returning a new Error Item.

Defining Server Validation Method



Note Default Parameters

```
Innovator inn = this.getInnovator();
string p = this.getProperty("_patent_required", "0");
string c = this.getProperty("_completion_date", "");
if (p=="1" && c=="") {
    return inn.newError("Completed Date Required");
}
return this;
```



Copyright © 2014 Aras All Rights Reserved.

aras.com

In this example the priority and support_contract properties are assigned to variables from the context of the current Design Request Item and then tested using a condition statement.

A new Error Item is returned if the condition is true and a dialog is displayed to the user.

This Method can then be attached to an onBeforeAdd or onBeforeUpdate method to prevent the data from being added or saved to the database.

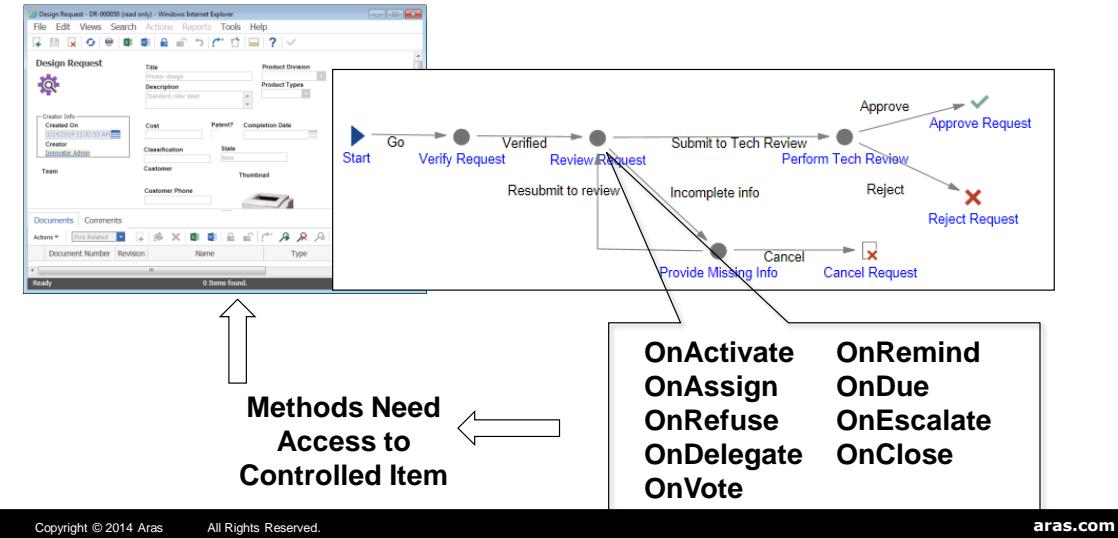
VB.NET

```
Dim inn as Innovator = Me.getInnovator()
Dim p as String = Me.getProperty("priority", "0")
Dim c as String = Me.getProperty("support_contract")
If p="1" And c="0" then
    Return inn.newError("Contract customer must be P1")
End If
Return Me
```

Developing Workflow Validation



- Context Item for Workflow events is current Workflow Activity



Testing for Item conditions in a Workflow also may require validation.

If you attach a Method to an Activity event the context Item is the actual Activity Item which is typically not what you are attempting to validate.

Remember that every Workflow Process is related to a Controlled Item which typically contains the properties to be validated.

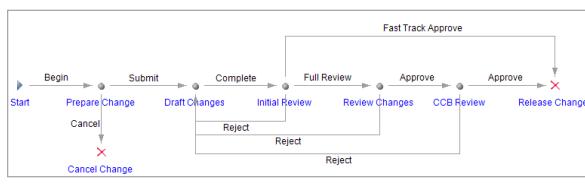
Defining Workflow Validation Method



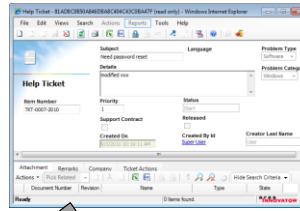
Call custom action
on this Item

```
Innovator inn = this.getInnovator();
Item controlledItem = this.apply("Get Controlled Item");
string subj = controlledItem.getProperty("subject", "");
if (subj=="") {
    return inn.newError("Subject required");
}
return this;
```

Workflow Process



Controlled Item



Copyright © 2014 Aras

All Rights Reserved.

aras.com

Accessing the controlled item from a Workflow Activity requires an additional step.

An AML Action Method named **Get Controlled Item** is available on the Aras.com Community Projects site that retrieves the controlled item from a Workflow Activity. Search for the package named *Workflow Automation Examples*.

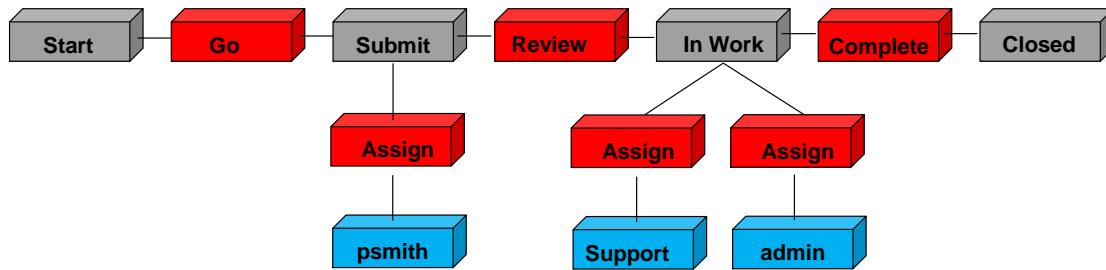
You can use the Item class apply method discussed earlier in this course to run a custom action.. The apply method will pass the context of the current activity to the Get Controlled Item Method.

Once you have a handle to the controlled item you can then proceed to validate any properties on the that item.

VB.NET

```
Dim inn as Innovator = Me.getInnovator()
Dim controlledItem as Item =
Me.apply("Get Controlled Item")
Dim subj as String = controlledItem.getProperty("subject", "")
If subj==""
    return inn.newError("Subject required")
Else
    Return Me
End if
```

Understanding Workflow Definition Items

**Workflow Map****Corresponding Items**

Copyright © 2014 Aras All Rights Reserved.

aras.com

Understanding Definition Workflow Items

When you create and save a Workflow Map, Aras Innovator generates corresponding Items and Relationships in the database to define the workflow process.

The following ItemTypes are defined by the Workflow Map:

Activity

Defines the Activity created on the Workflow Map.

Workflow Process Path

Relationship between Activities – represented as the path on the Workflow Map.

Activity Assignment

Relationship from an Activity to an Identity.

Activity Email

Relationship from an Activity to an E-mail message.

Activity Method

Relationship from an Activity to a Server Event Method.

Activity Task

Null relationship from an Activity which defines each Task on the Activity.

Activity Transitions

Relationship from the Activity to a Life Cycle Transition for promotion.

Activity Variable

Null relationship defining an Activity variable that can appear on the Workflow Completion dialog.

When you define Server Events on a Workflow Activity, you have access to the items described above to make additions, modifications or deletions while the workflow is in process on a controlled item.

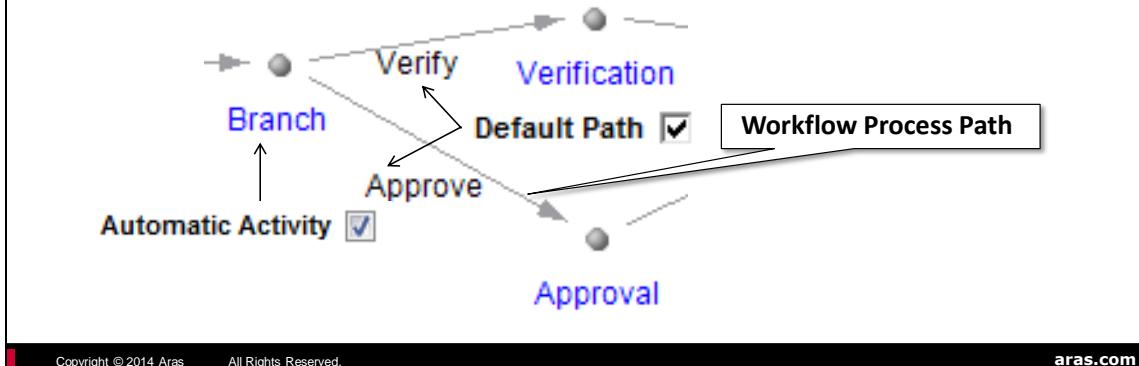
Two of the common techniques used in customizing a workflow are to create Assignments dynamically on an Activity as well as change to a default process route defined on the Workflow Map.

The following sections describe these options.

Creating Dynamic Workflow Routes



- Change Path Route Based on Conditions
- All Paths from Automatic Activity should be "Default"
- Method will "remove" a Default path



Another common problem when creating a solution with Workflows is dynamically assigning a route based on custom business logic.

Remember that when the Default Path setting is used on a path following an Automatic Activity the workflow will follow that path regardless of any other settings.

You can use this feature to dynamically “switch” paths so that only one path becomes the Default when the Workflow Activity becomes active.

In the example above, the workflow is originally set so that both paths following the Branch activity are Default. If an activity is created to change the Default setting on one of the paths, then only one path will be followed.

Note

A workflow path is represented by the ItemType "Workflow Process Path".

Defining Dynamic Workflow Route Method

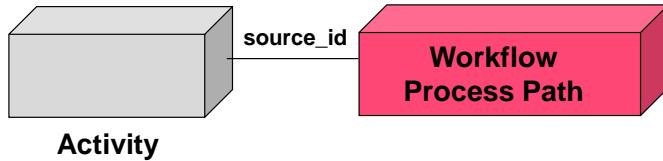


```
//Remove a "Default" path
string path_name="Approve";
Item qry = this.newItem("Workflow Process Path", "get");
qry.setProperty("source_id", this.getID());
qry.setProperty("name", path_name);
Item path = qry.apply();
path.setAction("edit");
path.setProperty("is_default", "0");
path = path.apply();
return this;
```

Get path

Deselect default

Default Path



Copyright © 2014 Aras All Rights Reserved.

aras.com

This example shows how the Default path can be turned “off” by setting the `is_default` property on the Workflow Process Path Item to false (“0”).

If you assign this Method to an Automatic Activity as discussed on the previous page you can indicate the “direction” the workflow should take based on some business rule.

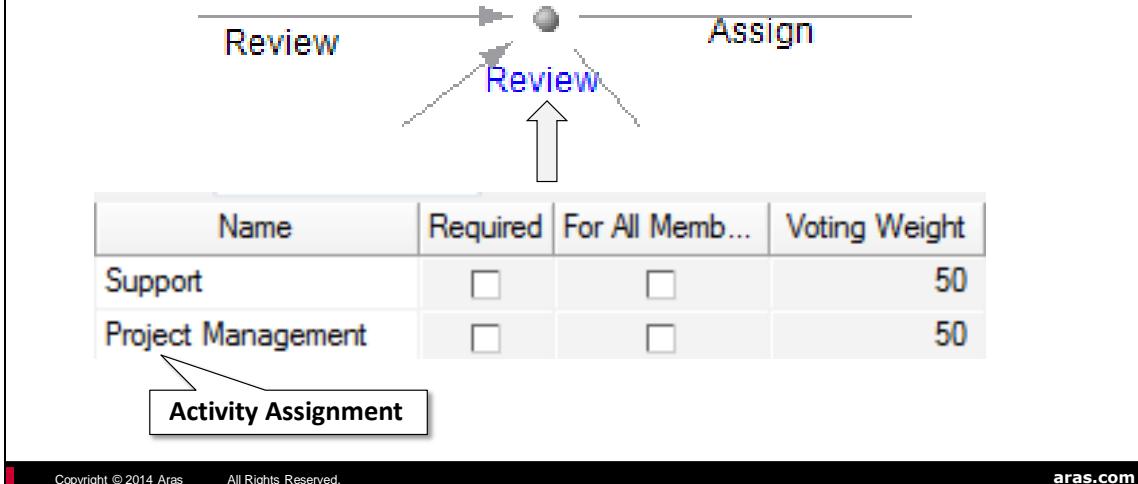
VB.NET

```
Dim path_name as String ="Approve"
Dim qry as Item = Me.newItem("Workflow Process Path", "get")
qry.setProperty("source_id", Me.getID())
qry.setProperty("name", path_name);
Dim path as Item = qry.apply()
path.setAction("edit")
path.setProperty("is_default", "0")
path = path.apply()
Return Me
```

Developing Workflow Dynamic Assignments



- Add, modify or remove assignments based on business rules



There may be times when the recipients of an Activity Assignment are unknown until the workflow is in progress.

You can create a dynamic assignment on an activity that indicates who should receive the assignment and what voting weight they should receive.

Note

Assignments are represented by the ItemType "Activity Assignment"

Defining Dynamic Assignments Method

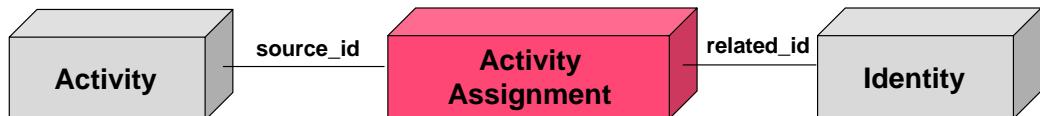


```
//Add assignment to an activity
Item asgn = this newItem("Activity Assignment", "add");
asgn.setProperty("source_id", this.getID());
Item iden = this newItem("Identity", "get");
iden.setProperty("keyed_name", "Peter Smith");
iden = iden.apply();
asgn.setProperty("related_id", iden.getID());
asgn.setProperty("voting_weight", "100");
Item res = asgn.apply();
```

Add Assignment

Set Assignment
to this Activity

Relate to
Identity



Copyright © 2014 Aras All Rights Reserved.

aras.com

In this example, a new Activity Assignment relationship is created between the current Activity and an existing Identity (Peter Smith).

Note how both the source and related id's are set on the Relationship. The Relationship also stores the voting weight of the Identity in a property named "voting weight".

VB.NET

```
Dim asgn as Item= Me newItem("Activity Assignment", "add")
asgn.setProperty("source_id", this.getID())
Dim iden as Item = Me newItem("Identity", "get")
iden.setProperty("keyed_name", "Peter Smith")
iden = iden.apply()
asgn.setProperty("related_id", iden.getID())
asgn.setProperty("voting_weight", "100")
Dim res as Item = asgn.apply()
```



Summary

In this unit you learned how to apply server Methods to customize a solution.

You should now be able to:

- ✓ Develop ItemType Validation Routines
- ✓ Develop Workflow Validation Routines
- ✓ Create Dynamic Assignment Routes in a Workflow
- ✓ Create Dynamic Assignments in a Workflow



Lab Exercise

Goal:

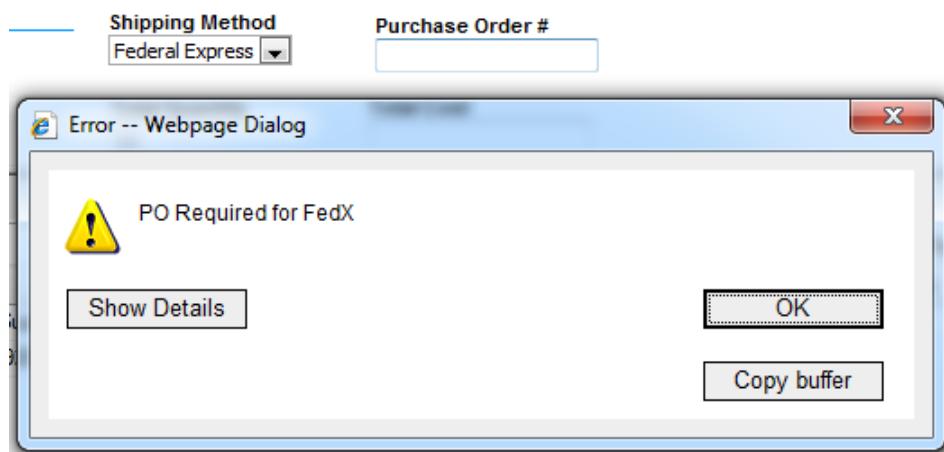
Be able to use server Methods and associate them with various events to customize a solution.

Scenario:

In this exercise, you will create several Methods to provide custom business logic to a solution. You will then associate each Method with the appropriate event to extend the standard behavior.

Before Saving a Sales Order

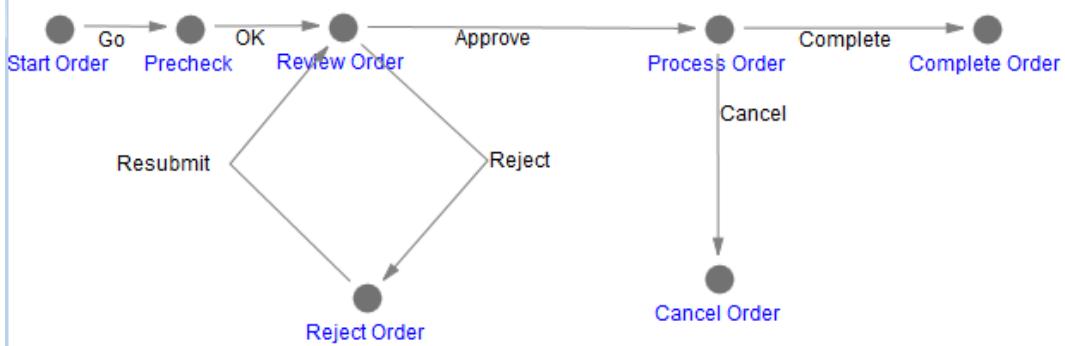
1. A Sales Order with Federal Express Shipping Method requires a P.O. number be provided before the order can be saved.

**Before Promoting Sales Order**

2. A Sales Order requires a Shipping Date before it can be promoted from the **New** to **In Progress** lifecycle state. Create a Method to stop the promotion if this is not true and attach it to the Sales Order lifecycle transition between New and In Progress.



Workflow OnActivate Activity



3. When the Sales Order **Reject Order** Workflow Activity is activated (onActivate), the following Remark should automatically be added to the Sales Order Remarks tab:

Customer	Parts	Remarks
Actions ▾ No Related <input type="button" value="+"/>		
Date [...]	User [...]	Comments
10/22/2014 12:1...	Innovator Admin	Sales Order Rejected

Create a server Method and obtain the controlled item from the workflow activity as shown in this unit:

C#

```
Item order = this.apply("Get Controlled Item");
```

VB.Net

```
Dim order as Item = Me.apply("Get Controlled Item")
```

Use the `createRelationship` Item class method to add a new "Sales Order Remarks" relationship on the `order`.

Set the `comment_text` property on the relationship item to "Sales Order Cancelled" and apply the change to the `order`.

Attach the Method to the Server Event of the **Reject Order** activity (onActivate).

To test the behavior, create a NEW Sales Order and then go the MyInnovator > InBasket to Reject it.

Vote	
Vote:	<input checked="" type="button" value="Approve"/> <input type="button" value="Reject"/> <input type="button" value="Delegate"/> <input type="button" value="Refuse"/>
Comments:	

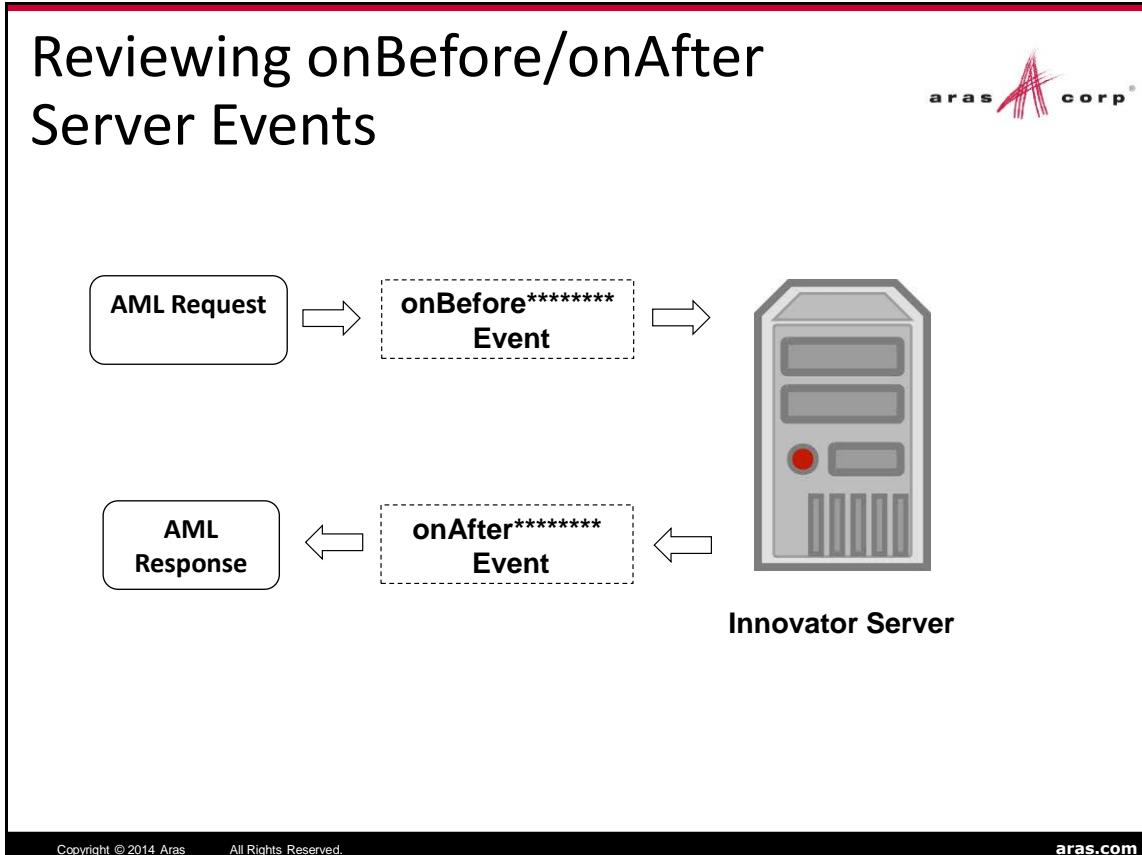


Unit 9 Passing Event Parameters

Overview: All server Methods (only) have access to a reserved class named RequestState to allow information to be passed between Methods when they are invoked from an event. Server Methods are typically associated with system events that execute before and after an interaction with the Aras Innovator server. With this facility, you can pass information from onBefore***** to onAfter*** server events on the same (or different) Items, check for the existence of event parameters and remove or clear parameters as necessary.

- Objectives:**
- ✓ Reviewing onBefore and onAfter Server Events
 - ✓ Implementing the RequestState Class
 - ✓ Adding RequestState Object Keys
 - ✓ Retrieving RequestState Object Keys
 - ✓ Clearing RequestState
 - ✓ Examining Example Use Cases

Reviewing onBefore/onAfter Server Events



Reviewing onBefore and onAfter Server Events

Server Methods are associated with one or more server events to prevent, replace or extend the standard behavior of Aras Innovator.

onBefore** Event**

This collection of events execute before the AML request from the client is passed to the Innovator server and are typically used to validate data based on custom business logic (e.g. onBeforeAdd, onBeforeUpdate, onBeforeDelete, etc.).

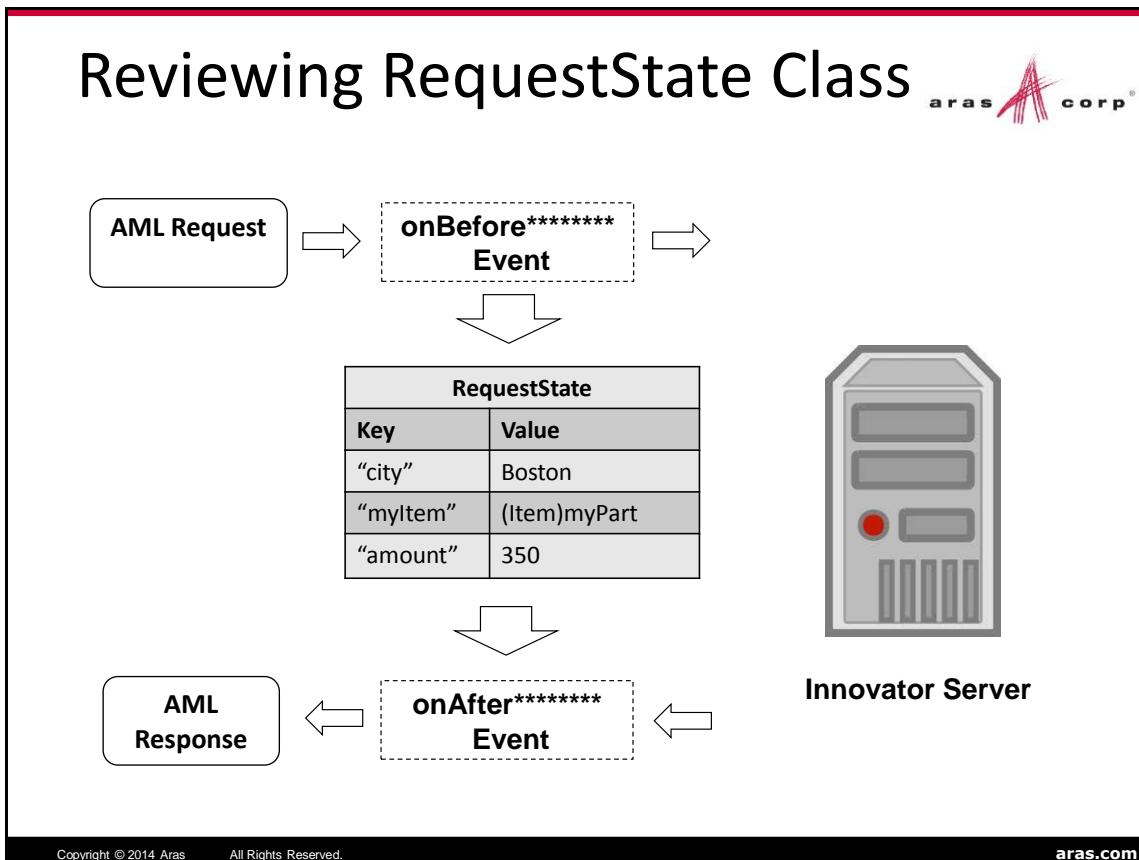
onAfter** Event**

This collection of events execute after the server interaction has executed and have access to the information returned from the Aras Innovator server (e.g. onAfterAdd, onAfterUpdate, onAfterDelete).

on*** Event**

This collection of events replaces the standard behavior defined by Aras Innovator completely. It is the responsibility of the Method developer to create the appropriate request and format the response.

In this unit, you will learn how to assign values to a reserved class named RequestState that is available to store and retrieve values between Methods called from different events.



Reviewing the RequestState Class

The RequestState class is an implementation of the Aras.Server.Core.IContextState interface and is available to any server Method created in Aras Innovator.

The RequestState class contains an Object hash table that can be assigned key/values using a series of class methods. A typical example is to assign one or more values in a Method associated with an onBefore**** event. Values can then be retrieved using a server Method associated with an onAfter**** event.

For example, when an Item is deleted it may be useful to store information about the Item before it is removed from the database and then use that information in a later call to the server after the Item is removed (and no longer available).

Note

The RequestState values are session-based and should be removed or cleared when they are no longer necessary.



Adding RequestState Keys

- `RequestState.Add(Object key, Object value)`
- `RequestState[Object value] = Object value`

```
RequestState.Add("city", "Boston");
RequestState.Add("itemKey", myItem);
RequestState.Add("amount", 350);
```

"city"	"Boston"
"itemKey"	(Item)myItem
"amount"	350

Copyright © 2014 Aras All Rights Reserved.

aras.com

Adding RequestState Keys

To add a session key use the `Add` method and supply the key name and key value or assign a key to a value. Note that the keys and corresponding values support any Object type.

In the example above, three key/value pairs have been assigned to the `RequestState`. The first key named “city” is set to the string of “Boston”. The second key named “itemKey” is assigned to an Aras Innovator Item that has been declared in the program. The third key refers to an integer value.

Notes

These examples all use a string as the key name. You can use any Object type to define the key.

An exception is thrown if you attempt to add a key and it already exists in the collection using the `Add` method. Assigning a value to an existing key directly (as in the “amount” example shown above) overwrites the current value.

Key assignments respect the Method sort order on a configured event. If two Methods have been assigned to the same event and both assign a value to the same key, the last Method that executes in the sort order will replace the value (assuming it already exists).

VB.NET

```
RequestState.Add("city", "Boston")
RequestState.Add("itemkey", myItem)
RequestState.Add("amount", 350)
```

Retrieving RequestState Values



- Object value = RequestState[Object key];

```
string city = (string)RequestState["city"];
Item itm = (Item)RequestState["itemKey"];
int total = (int)RequestState["amount"];
```

“city”	“Boston”
“itemKey”	(Item)myItem
“amount”	350

Copyright © 2014 Aras All Rights Reserved.

aras.com

Retrieving RequestState Values

In a subsequent Method you retrieve the values using the appropriate key.

In the example above, the values set on the previous page are retrieved into their appropriate variable types. Note the use of casting to ensure the appropriate type is returned from the RequestState.

Note

Null (or *Nothing* in VB) is returned if you attempt to retrieve a key that does not exist in the collection.

VB.NET

```
Dim city As String = DirectCast(RequestState("city"), String)
Dim itm As Item = DirectCast(RequestState("itemKey"), Item)
Dim total As Integer = CInt(RequestState("amount"))
```



Clearing RequestState

- ❖ RequestState.Remove(Object key);
- ❖ RequestState.Clear();

```
RequestState.Remove("city");  
RequestState.Clear();
```

"city"	
"itemKey"	
"amount"	

Copyright © 2014 Aras All Rights Reserved.

aras.com

Clearing RequestState Values

The RequestState class uses the session cache to allow the key/value pairs to persist between server events. It is a best practice to remove the key/values as soon as they are no longer needed.

To remove each key individually you can use the Remove method and supply the appropriate key name.

To clear all key/value pairs use the Clear method which will remove all values from the RequestState.

VB.NET

```
RequestState.Remove("city")  
RequestState.Clear()
```

Checking for Keys



- boolean value = RequestState.Contains(Object key);
- integer value = RequestState.Count;

```
if (RequestState.Contains("city")) ...
if (RequestState.Count < 1) ...
```

“city”	“Boston”
“itemKey”	(Item)myItem
“amount”	350

3

Checking for Keys

You can check to see if a key exists in a collection as well as determine how many key/value pairs have been assigned.

Use the Contains method to determine if a key exists in the collection. A true or false is returned based on the search.

Use the Count property to determine how many key/value pairs have been assigned.

The following C# example adds a new key if it does not already exist. Otherwise the key is reassigned:

```
string key = "amountKey";
string value = 350;
if (RequestState.Contains(key))
{
    RequestState[key] = value;
}
else
{
    RequestState.Add(key, value);
}
return null;
```

VB.NET

```
Dim key As String = "amountKey"  
Dim value As String = 350  
If RequestState.Contains(key) Then  
    RequestState(key) = value  
Else  
    RequestState.Add(key, value)  
End If  
Return Nothing
```

Use Case: Item Versioning



- **onBeforeVersion**
 - Obtain current property values before item is versioned
 - Set RequestState keys to original values

- **onAfterVersion**
 - Retrieve RequestState property values
 - Compare values and prevent versioning if certain values have changed

Use Case Example: Item Versioning

This example demonstrates how to use the RequestState class with a versioned Item. The following Method has been created for the onBeforeVersion event of a Document Item. The current document number as well as the name is stored in the RequestState collection.

```
string doc_number = this.getProperty("item_number");
string name = this.getProperty("name");

RequestState.Add("docnum", doc_number);
RequestState.Add("docname", name);

return null;
```

VB.NET

```
Dim doc_number As String = Me.getProperty("item_number")
Dim name As String = Me.getProperty("name")

RequestState.Add("docnum", doc_number)
RequestState.Add("docname", name)

Return Nothing
```

A second server Method is created to compare the original values of the Document with any changes. In this example, an error is returned if the user attempts to change the name or document number in the new version:

```
string previous_number = (string)RequestState["docnum"];
string previous_name = (string)RequestState["docname"];

if (previous_number != this.getProperty("item_number") ||
    previous_name != this.getProperty("name")) {
    return this.getInnovator().newError("Cannot change doc number or
name in versions");
}
RequestState.Clear();
return this;
```

VB.NET

```
Dim previous_number As String =
    DirectCast(RequestState("docnum"), String)
Dim previous_name As String =
    DirectCast(RequestState("docname"), String)

If previous_number <> Me.getProperty("item_number")
    OrElse previous_name <> Me.getProperty("name") Then
    Return _
        Me.getInnovator().newError("Cannot change doc number or
name in versions")
End If
RequestState.Clear()
Return Me
```

The server Methods are then configured on the Document ItemType with the appropriate server events:

Server Events	Actions	Life Cycles	Workflows	TOC Access	TOC View	Client Events	Can Add	Perm
Actions	Pick Related					Hide Search Criteria	<input checked="" type="checkbox"/>	Page Size:
Name								
_Event After Version	CSharp	2	World			onAfterVersion		1664
_Event Before Version	CSharp	2	World			onBeforeVersion		1536

Use Case: Item Deletion



- **onBeforeDelete**
 - Obtain property values of items before it is deleted
 - Set RequestState key equal to item and its values

- **onAfterDelete**
 - Retrieve RequestState item after item is deleted
 - Send e-mail with message text confirming item is deleted (with property data)

Use Case Example: Item Deletion

In this example, an email message is generated when an Item is deleted from the system. The first Method obtains the information about the Product to be deleted and caches it in the RequestState collection before the Item is deleted. Note that the value stored is an Object of type Item.

```
//Retrieve the desired property values before deleting item
Item itm = this newItem(this.getType(), "get");
itm.setID(this.getID());
itm.setAttribute("select", "item_number,name");
itm = itm.apply();

RequestState.Add("itmKey", itm);
return this;
```

VB.NET

```
Dim itm As Item = Me.newItem(Me.[getType](), "get")
itm.setID(Me.getID())
itm.setAttribute("select", "item_number,name")
itm = itm.apply()

RequestState.Add("itmKey", itm)
Return Me
```

The second Method checks to make sure the itemKey has been set and then retrieves the cached information. An email message is then generated using the email method of the Aras IOM Item class.

```
if (!RequestState.Contains("itmKey")) {  
    return null;  
}  
  
Item deleted_item = (Item)RequestState["itmKey"];  
  
Item identityItem = this newItem("Identity", "get");  
identityItem.setProperty("name", "Innovator Admin");  
identityItem = identityItem.apply();  
  
Item emailItem = this newItem("EMail Message", "get");  
emailItem.setProperty("name", "EventDeleteMessage");  
emailItem = emailItem.apply();  
  
bool sent = deleted_item.email(emailItem, identityItem);  
  
RequestState.Remove("itemKey");  
return this;
```

VB.NET

```
If Not RequestState.Contains("itmKey") Then  
    Return Nothing  
End If  
  
Dim deleted_item As Item =  
    DirectCast(RequestState("itmKey"), Item)  
  
Dim identityItem As Item = Me newItem("Identity", "get")  
identityItem.setProperty("name", "Innovator Admin")  
identityItem = identityItem.apply()  
  
Dim emailItem As Item = Me newItem("EMail Message", "get")  
emailItem.setProperty("name", "EventDeleteMessage")  
emailItem = emailItem.apply()  
  
Dim sent As Boolean = deleted_item.email(emailItem, identityItem)  
  
RequestState.Remove("itemKey")  
Return Me
```

The Product ItemType is configured to use both Methods.

Name	Method Type	Ver	execution_allowe...	Comments	Event	Sort Order
_Event After Delete	CSharp	1	World		onAfterDelete	128
_Event Before Delete	CSharp	3	World		onBeforeDelete	256

The following EMail Message is used to send the email:

EMail Message

Name EventDeleteMessage	Body Plain Product Number \${Item[@type="Product"]/item_number} was deleted. Product Number \${Item[@type="Product"]/name} is no longer available.	
From User Innovator Admin	Subject Product \${Item[@type="Product"]/item}	Body Html



Use Case: Related Items

- **onBeforeUpdate (Parent Item)**
 - Obtain property values of parent item
 - Set RequestState keys equal to property values

- **onBeforeAdd (Related Item)**
 - Retrieve RequestState property values
 - Set property values of related item equal to property values of parent item

Use Case Example: Related Items

In this example two different Items are used with the RequestState collection. When a user creates a new Model from a Product Item the Model number is automatically prefixed with the Product Number.

The following Method is created to obtain the Product number and cache it in the RequestState before the Product is updated.

```
string prodnum = this.getProperty("item_number");
RequestState.Add("prodnum", prodnum);
return this;
```

VB.NET

```
Dim part As String = Me.getProperty("item_number")
RequestState.Add("prodnum", prodnum)
Return Me
```

The second Method retrieves the new Model number and the current Product before the new Model is added to the system and prefixes the Model number with the Product number:

```

if (!RequestState.Contains("prodnum")) return this;
string prodnumber = (string)RequestState["prodnum"];
string modelnum = this.getProperty("item_number", "");
this.setProperty("item_number", prodnumber + "-" + modelnum);
RequestState.Remove("prodnum");
return this;

```

VB.NET

```

If Not RequestState.Contains("prodnum") Then
    Return Me
End If
Dim prodnumber As String = DirectCast(RequestState("prodnum"), 
String)
Dim modelnum As String = Me.getProperty("item_number", "")
Me.setProperty("item_number", prodnumber & "-" & modelnum)
RequestState.Remove("prodnum")
Return Me

```

The first Method is configured on the Product ItemType:

A screenshot of a software interface showing the 'Server Events' tab selected. The table below lists two entries:

Name	Method Type	Ver	execution_allo...	Comments	Event	Sort Order
_ProductModelAdd	CSharp	1	World		onBeforeAdd	128
_ProductModelAdd	CSharp	1	World		onBeforeUpdate	256

The second Method is configured on the Model ItemType:

A screenshot of a software interface showing the 'Server Events' tab selected. The table below lists one entry:

Name	Method Type	Ver	execution_allo...	Comments	Event	Sort Order
_ModelAdd	CSharp	1	World		onBeforeAdd	128

Result

Product



Product Number
SN10005

Name
Smart Phone (Standard)

Description

Models

Actions ▾ No Related |        

Model Number	Name	Release Number
SN10005-100A	16GB Plasma Display	



Summary

In this unit you learned how to pass parameters between server event Methods.

You should now be able to:

- ✓ Implement the RequestState Class with Server Events
- ✓ Add RequestState Object Keys
- ✓ Retrieve RequestState Object Keys
- ✓ Clear the RequestState of key/value



Review Questions

A RequestState key/value can only be set in what kind of Aras Method?

What variable data types are supported for the key or value of a RequestState?

How long does a RequestState key/value persist?

Why is it important to remove a RequestState variable once it is no longer used?



Lab Exercise

Goal:

Be able to save and restore a RequestState value using a server events.

Scenario:

A user should not be able to change the Part Type once it has been established for a Part item. You need to add a business rule that prevents a user from versioning a Part if the Type (classification) is changed.

Part	Part Number	Revision	State
	0515-4700	A	Preliminary
	Name	Screw (Plastic) - M3x12.30	
Created By: Created On: Modified By: Modified On: Locked By: Major Rev: Release Date: Effective Date: Generation: State:	Type Assembly	Unit EA	Make / Buy Make
	Cost		

Long Description

Type cannot be changed between versions.

[Show Details](#) [OK](#)

[Copy buffer](#)

Create A Server Method for the onBeforeVersion event of the Part ItemType

1. Create a Server Method named **PartBeforeVersion**.
2. Get the value of the classification property of the current Part item (this or Me).
3. Add a RequestState key named "part_type" with the value of the classification property.
4. Return the current item (this or Me).

Create A Server Method for the onAfterVersion event of the Part ItemType

5. Create a Server Method named **PartAfterVersion**.
6. Retrieve the value of the "part_type" from the RequestState. You will need to cast the return values to a string. See an example of string casting in this unit.
7. Get the value of the *classification* property of the current Part item.
8. Use an "if" statement to determine if the previous part_type and the current classification are equal. If they are not return an error.

9. Clear the RequestState (best practice).

Associate Methods with the Part ItemType

10. Associate the Methods above with the onBeforeVersion and onAfterVersion server events of the Part ItemType.



Unit 10 Exploring Aras Object Functions

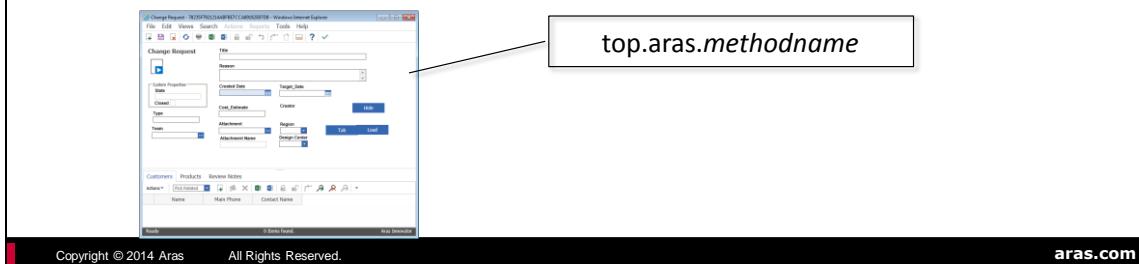
Overview: In this unit you will learn how to use the various client JavaScript methods available in the aras Object. Every window in Aras Innovator has access to the aras Object which contains over 200 utility functions that can be used in client Methods.

Objectives: ✓ Classifying Aras Object Functions

Reviewing Aras Object methods



- Every Aras Innovator window has a reference to "aras" object
- Object contains 200+ JavaScript methods (in addition to IOM)
- Source file: aras_object.js
- Contains useful client utilities for Client Methods only



The aras_object.js file is located in the /Innovator/Client/javascript folder

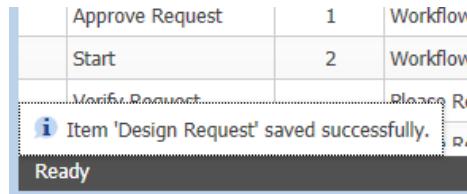
You access the aras Object from any JavaScript client Method using the syntax:

```
top.aras.functionname()
```

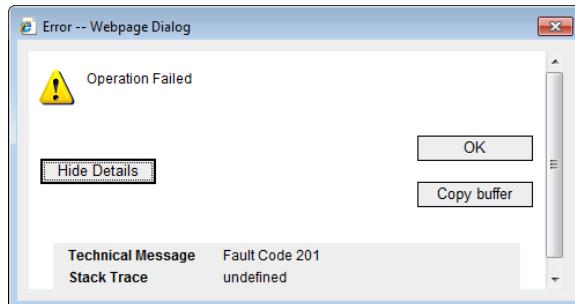
Displaying Prompts and Errors



- **top.aras.AlertSuccess**



- **top.aras.AlertError**



Copyright © 2014 Aras All Rights Reserved.

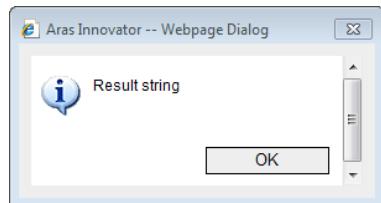
aras.com

Examples

```
top.aras.AlertSuccess("Operation Completed Successfully");
top.aras.AlertError("Operation Failed", "Fault Code 201",
"undefined");
```

The AlertSuccess method has also been designed to handle a Result Item generated by the Innovator class newResult method. It will parse the Result Item and only display the inner element of the Item (string) to the user.

```
var inn = this.getInnovator();
var itm = inn.newResult("Result string");
top.aras.AlertSuccess(itm);
```

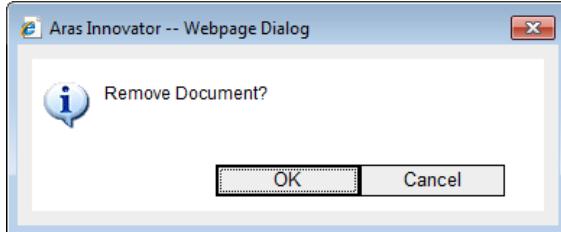


The AlertError method supports the display of a custom message which can include a fault code and stack trace as optional arguments.

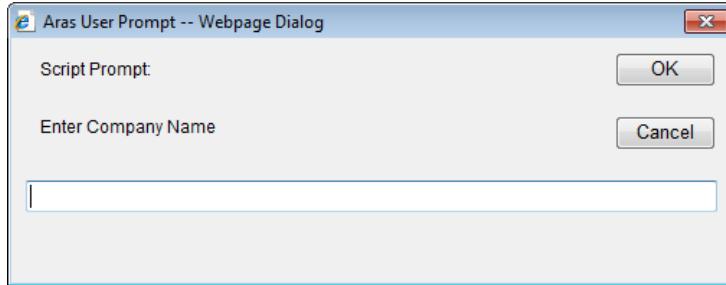
Displaying Confirmation Dialog



■ top.aras.confirm



■ top.aras.prompt



Copyright © 2014 Aras

All Rights Reserved.

aras.com

Confirm and Prompt

Displays a confirmation dialog box which contains a message and OK and Cancel buttons.

Parameters:

- message - string. Message to display in a dialog.
- win - parent window for the dialog.

Returns:

- true - if a user clicked the OK button.
- false - if a user clicked Cancel button.

Examples

```
var result = top.aras.confirm("Are you sure?", "");  
  
var result = top.aras.prompt("Enter Company Name", "");
```

Executing a JavaScript Client Method



- **evalMethod**
 - Executes Client JavaScript Method
- **evalItemMethod**
 - Executes Client JavaScript Method on Item node and can accept arguments

```
01 //Execute Client Methods
02 var results = top.aras.evalMethod("get_help_tix");
03 var result =
04     top.aras.evalMethodItem("total_cost", this.node,
05         args);
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

Executing a Client Method from another Method

The Run Client Method Action that you have used in this course to test a Method uses these two functions. You can examine the *Run Lab* method to see how they are being used.

Examples

```
var result=top.aras.evalMethod("My Client Method")

//Pass arguments in body string
var result = top.aras.evalMethodItem("SetPriority", itm.node,
"<priority>2</priority>");
```



Getting Paths

- **getServerBaseURL**
 - Get URL of Innovator Server
- **getBaseUrl**
 - Get URL of Innovator Client
- **getWorkingDir**
 - Get (and check) User's Working File Checkout Directory

Examples

```
var directory = top.aras.getWorkingDir();  
var url = top.aras.getServerBaseURL();  
var base = top.aras.getBaseUrl();
```

Getting User Information



- **getLoginName**
 - Returns current User's login name
- **getUserType**
 - Returns user or admin
- **isAdminUser**
- **getUserID**
- **getDatabase**
- **getIdentityList**
 - Returns list of Identity ID's that User belongs to

Copyright © 2014 Aras All Rights Reserved.

aras.com

Examples

```
var r0 = top.aras.getLoginName();
var r1 = top.aras.getUserType();
var r2 = top.aras.isAdminUser();
var r3 = top.aras.getUserID();
var r4 = top.aras.getDatabase();
var r5 = top.aras.getIdentityList();

top.aras.AlertSuccess("Login: " + r0 + " Database: " + r4);
```



Validate User

■ ValidateVote

- Validate current user's password or e-signature
 - Requires MD5 encrypted string and mode ("password" or "esignature")
 - Returns "pass" or "fail"

```
var md5_str = this.getInnovator().ScalcMD5("innovator");
var result = top.aras.ValidateVote(md5_str, "esignature");
if (result=="fail") {
    top.aras.AlertError("Invalid Signature");
    return false;
}
```

You can check a user's password or signature in a client Method to ensure the user is valid to proceed. The password or signature must be MD5 encrypted to perform the comparison as shown in the example.

Innovator Utility Functions



■ newIOMInnovator

- Obtains the innovator Object

```
var innovator = top.aras.newIOMInnovator();
```

■ applyAML(AML string)

```
var AML =
    "<AML><Item type='User' action='get' /></AML>";
var result = top.aras.applyAML(AML);
```

■ applyItem(AML Item string)

```
var AML =
    "<Item type='User' action='get' />;
var result = top.aras.applyItem(AML);
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

Innovator Utility Functions

Several utility functions are available to allow access to the Innovator object and to execute AML statements against the server using a client Method as shown above.

newIOMInnovator

Returns an Innovator object without supplying a Context Item.

applyAML

Sends an AML request to the server and returns a string containing the result. Note that the return is not an Item but a string variable

applyItem

Sends an AML request to the server with the <AML> wrapper tags. Returns a string containing the result. Note that the return is not an Item but a string variable



Summary

In this unit you learned about the various IOM methods available in the Innovator class.

You should now be able to:

- ✓ Identify and Classify Aras Object Functions



Lab Exercise

Goal:

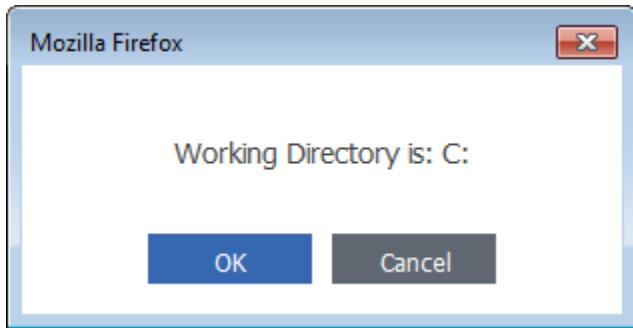
Be able use locate and use the Aras Object Functions in a client Method.

Scenario:

In this exercise, you will create a Method that use the functions discussed in this unit.

Steps:

1. Create a client Method that displays the current user's working directory in a confirm window.
Associate the Method with a Generic Action.



This page intentionally left blank.



Unit 11 Creating Client Event Methods

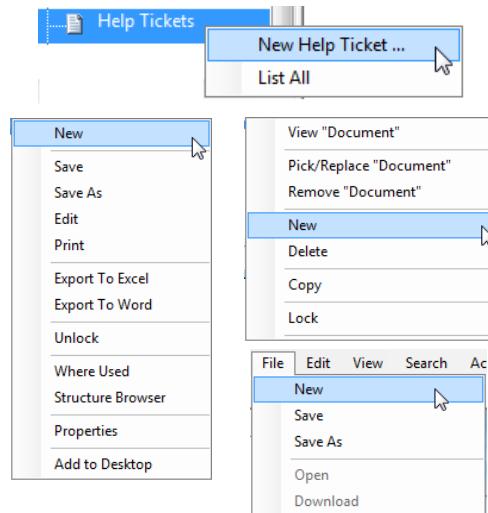
Overview: In this unit you will learn how to build and user client methods to customize and extend the standard client behavior of the system. This includes client validation and customizing the user interface.

- Objectives:**
- ✓ Customizing New Item Functionality
 - ✓ Examining the Aras Innovator Main Window Layout
 - ✓ Examining the Tear-off Window Layout
 - ✓ Developing Form and Field Client Methods
 - ✓ Predefine Search Criteria

Customizing New Item Functionality



- Client events triggered by creating a new Item in the user interface from:
 - Main Menu
 - TOC
 - Search Grid
 - Relationship Grid



Copyright © 2014 Aras All Rights Reserved.

aras.com

Customizing New Item

Three client events are available on each Item and are triggered when a user attempts to create an Item for the first time (add).

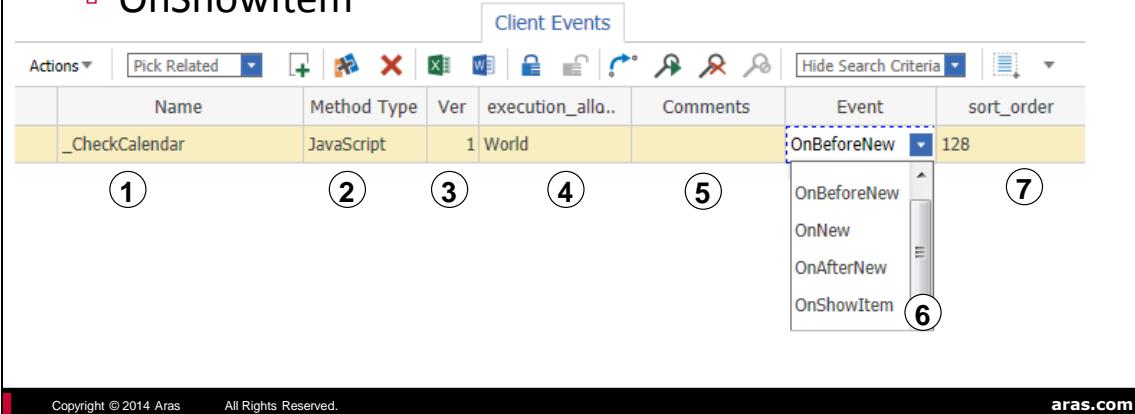
You can create logic to prevent the Item from being created or alter the server response before the user sees the form appear on the client.

Configuring ItemType Client Events



- Four client events supported:

- OnBeforeNew
- OnNew
- OnAfterNew
- OnShowItem



	Name	Method Type	Ver	execution_alla..	Comments	Event	sort_order
①	_CheckCalendar	JavaScript	1	World		② OnBeforeNew	③ 128
④	⑤	⑥	⑦			OnNew	
						OnAfterNew	
						OnShowItem	

Copyright © 2014 Aras All Rights Reserved. aras.com

Four events are available that are configured on the ItemType:

- **OnBeforeNew** – runs before a new Item is created and can be used to cancel a new Item request as well prevent display of the new Item Form. This is useful for preventing new Item creation if certain conditions have not been met.
- **OnNew** – replaces the standard new Item behavior. A client Method must be provided that provides custom logic to replace the standard new operation.
- **OnAfterNew** – runs after the Item has been created. Useful for populating a new Item with data or opening custom dialogs, etc. The new request has been processed by the server and the context Item contains the current values that will be displayed on the form. If you change (or add properties) to the AML response they will be displayed instead (or in addition to the existing properties) on the form.
- **OnShowItem** – replaces the standard logic for displaying a tear off window. A client Method must provide the custom logic to display an appropriate window.

Defining ItemType Client Method – OnBeforeNew



```
var itm = this newItem("Business Calendar Year", "get");
itm.setProperty("year", "2014");
itm = itm.apply();

if (itm.isError()) {
    top.aras.AlertError("Calendar Required to Create Item");
    return false;
}
return true;
```

In this example, a Calendar must be configured for the current year by the administrator or the Item in question cannot be created.

The program attempts to retrieve a Calendar with the year 2014. If not found the program displays an error and returns false. A false return will block the ability for the Aras client to create a new Item on the server.

Defining ItemType Client Method – OnAfterNew



```
var d = new Date();
var nowdate = d.getMonth()+1 + "/" + d.getDate() + "/" +
d.getFullYear();
var msg = "Item Added " + nowdate;

var rel = this newItem("Design Request Comments", "add");
rel.setProperty("remarks", msg);
this.addRelationship(rel);
```

Context Item
Altered

The screenshot shows a user interface for managing comments on a document. At the top, there are tabs for 'Documents', 'Comments' (which is selected), and 'Files'. Below the tabs is a toolbar with various icons. The main area displays a list of comments, with the first comment being 'Item added 01/04/2014'. A tooltip box with the text 'Context Item Altered' has an arrow pointing to the line of code where the 'remarks' property is set.

Copyright © 2014 Aras All Rights Reserved.

aras.com

This example demonstrates modification of the context Item returned from the server (response). In this case, we want to add more information to the Item before it processed by the user interface so the OnAfterNew event is used with the Method above.

Note

Take care when modifying a response context Item. If you provide invalid data the response may cause unpredictable results on the client.

Exploring User Interface Events

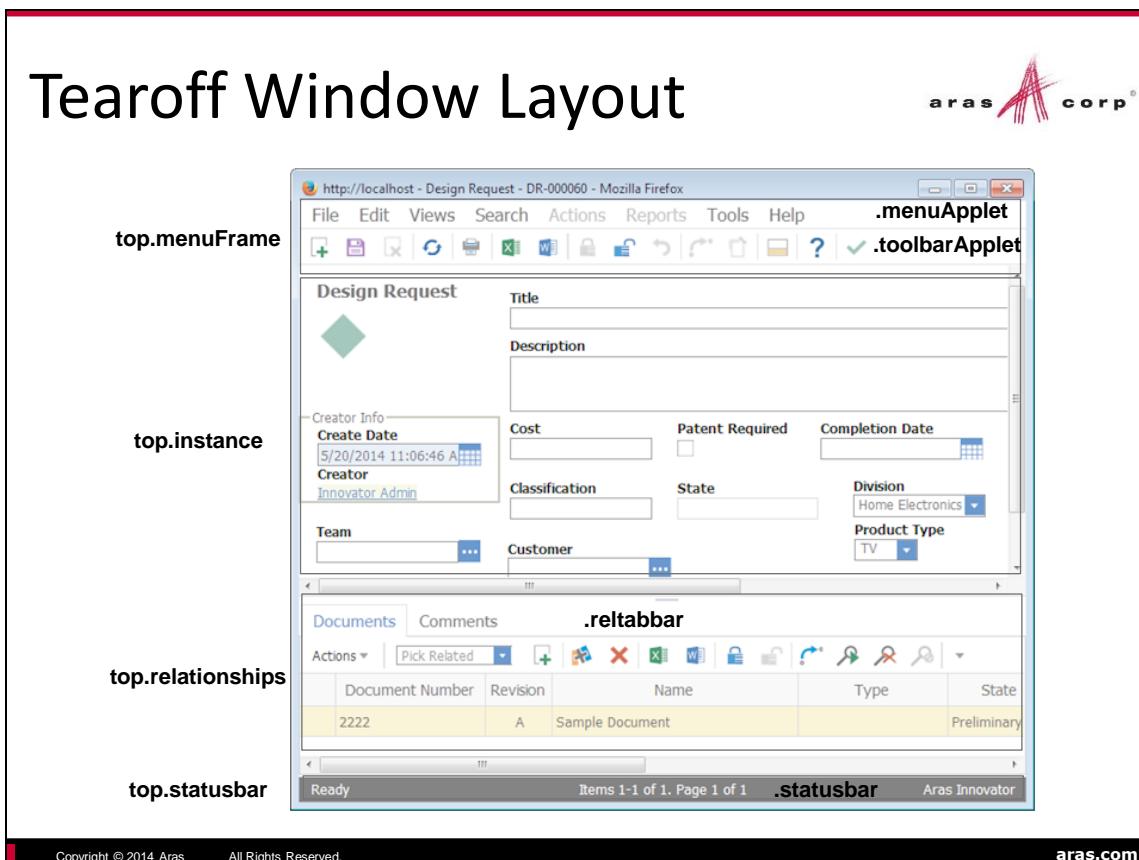


- Used to provide custom business logic for:
 - Form
 - Field
 - Relationship Grid

Exploring User Interface Client Events

The Aras Innovator client displays forms to the user to allow them to view and edit data relevant to an Item.

You can customize the interaction between the user and the form with a series of client events on the form, each field of the form as well as the rows and columns of the relationship grid (tab).



Tearoff Window Layout

The Aras Tearoff window contains a collection of iframes with controls that allow a user to select from a menu or toolbar, work within a form, choose items from a relationships grid and view status messages in a status bar.

You can provide custom logic by associating client Methods to elements on the form as well as the relationship grid rows and columns.

For more information on the relationship grid and creating custom call back methods see the appendices at the end of this student guide.

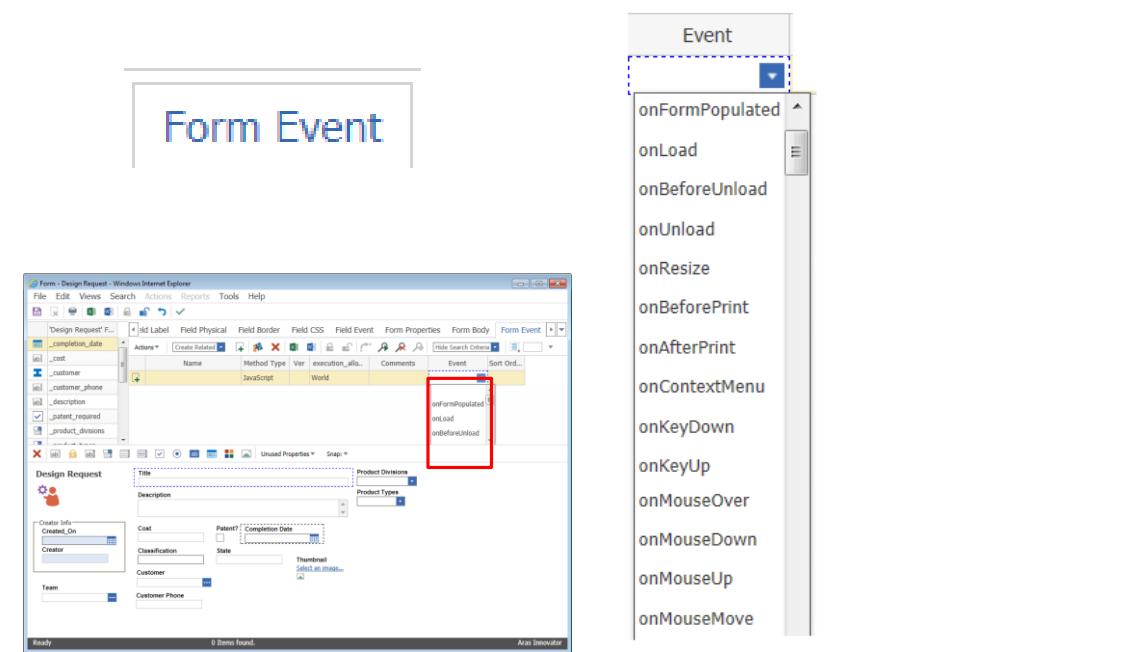
Developing Form and Field Client Event Methods



- Binding Method to Form and Field Events
- Useful for:
 - Populating/copying data
 - Validating client input
 - Providing additional feedback

Form and Field client event Methods are useful for populating data on a form as well as providing cross field validation before a user attempts to save Item information to the database.

Reviewing Form Client Events



The screenshot shows the Aras Innovator interface for managing form events. On the left, there's a preview window titled "Form Event". Below it is a search interface for "Design Request F...". A red box highlights the "Event" column in a table where several rows are listed, including "onFormPopulated", "onLoad", and "onBeforeUnload". To the right is a vertical list of all available events:

- onFormPopulated
- onLoad
- onBeforeUnload
- onUnload
- onResize
- onBeforePrint
- onAfterPrint
- onContextMenu
- onKeyDown
- onKeyUp
- onMouseOver
- onMouseDown
- onMouseUp
- onMouseMove

Copyright © 2014 Aras All Rights Reserved. aras.com

Many of the Form events are familiar to developers who have worked with HTML forms previously. In addition Aras Innovator has added some additional events that you can configure for your solution.

Form Loading Events

- **onFormPopulated** – runs when all Fields have been populated with data from the server.
- **onLoad** – runs when the HTML form has been loaded into the window.
- **onBeforeUnload** – runs just before the form is removed from view.
- **onUnLoad** – runs when the form is removed from view (on a window close).

Form Resize Event

- **onResize** – runs when window height or width is modified.

Form Print Events

- **onBeforePrint** – runs before the form is sent to the printer – allows for changes or modification before sending a form to the printer.
- **onAfterPrint** – runs after the form has been sent to the printer – allows form to be reverted back to previous view once form is sent to printer.

Form Menu Event

- **onContextMenu** – runs when user clicks right mouse button to display context menu item.

Form Keyboard Events

- **onKeyUp** – runs when key is released.
- **onKeyDown** – runs when key is pressed.

Form Mouse Events

- **onMouseOver** – runs when mouse pointer is moved over form body.
- **onMouseUp** – runs when left mouse button is released.
- **onMouseDown** – runs when left mouse button is pressed.
- **onMouseMove** – runs when mouse pointer is moved.

Reviewing Field Client Events



The screenshot shows the Aras Innovator interface. On the left, there is a 'Form - Design Request - Windows Internet Explorer' window titled 'Field Event'. It displays a table with columns for Name, Method Type, Ver, execution_all., and Comments. A row is selected with the values: Name = 'product_divisions', Method Type = 'Javascript', Ver = 'World', and Comments = 'onBlur onFocus onChange'. A red box highlights this row. Below the table, there is a 'Design Request' section with various fields like 'Description', 'Cost', 'Parent', 'Completion Date', 'Classification', 'State', 'Customer', 'Thumbnail', and 'Customer Phone'. On the right, a vertical list of events is shown under the heading 'Event': 'onBlur', 'onFocus', 'onChange', 'onClick', 'OnDoubleClick', and 'onSelect'. The 'onChange' event is also highlighted with a red box.

Field Events

Each field on a form can be configured to respond to the following events:

- **onBlur** – runs when a user leaves field.
- **onFocus** – runs when the cursor is moved into the field.
- **onChange** – runs when the value of the field is changed.
- **onClick** – runs when the user mouse clicks in the field.
- **OnDoubleClick** – runs when the user double clicks the mouse in the field.
- **onSelect** – runs when a user selects text in a field.



Examining the Context Item

- Each Client Method has access to the Innovator Context Item
- Context Item for Form and Field Event Methods is:
document.thisItem
- Item DOM Object is also available from:
document.item
- To obtain browser Window object (Form) or Field object use:
this

Copyright © 2014 Aras All Rights Reserved.

aras.com

The following references are also useful when creating client Methods to be used with Forms:

top.aras	Client Framework API object
document.item	An XML DOM object representing the Item in focus used with the Client Framework API.
document.thisItem	The context Item object (must be used instead of "this" in Form and Field event client Methods)
document.itemID	Context Item ID
document.itemType	Context ItemType
window.handleItemChange("property",value)	The function to update the Item Property value (cache) and the HTML Form Field value
getFieldByName("element_name")	Returns a form element using the control name configured in the form editor.

Saving Form Data

HTML Form

The screenshot shows the Aras Innovator interface. On the left, a 'Design Request' window is open with fields like 'Customer' and 'Customer Phone'. On the right, a table titled 'Documents' lists 'SPEC-0001' with revision 'A'. Below the interface is a large dump of Aras XML Cache (document.item) containing AML code. Two grey arrows point from the bottom of the interface towards the XML dump, indicating the flow of data.

```

<major_rev>A</major_rev>
<minor_rev>0</minor_rev>
<modified_by>innovator Admin</modified_by>
<modified_on>2014-01-29T11:22:02</modified_on>
<new_version>0</new_version>
<not_lockable>0</not_lockable>
<permission_id keyed_name="Udesign Request" type="Permission">42f166a24dd7e4861d4c7a2d72a849c13</permission>
<state>New</state>
<team_id keyed_name="Mobile Products" type="Team">DF7C5EF945F647019D/A443D/FCAE896</team_id>
<item_number>DR-000010</item_number>
</item>
</AML>

```

Aras XML Cache (document.item)

Copyright © 2014 Aras All Rights Reserved. aras.com

Form Data

Aras Innovator represents the current Item as a DOM object that stores property values. (You can view the cache object in the debugger by inspecting this.*document.item*).

As a user enters data the HTML Form captures that data in the form control. That data must also be set in the AML cache (document.item) in order for the data to be sent to the server.

In normal operations, the cache is updated each time a user leaves a field (field loses focus). When a user saves an Item the appropriate XML tags are defined with values to be submitted to the server.

A problem can occur however if you programmatically set the value of an Item using a Method associated with a Form or Field event.

Both the HTML control and the Item cache must be updated to the new value so that the Form presentation and data cache are the same. Aras provides a built in function (described) on the next page to simplify this procedure.



Getting and Setting a Field Property Value

- **this.value**
 - Set or get value in HTML control only
- **document.thisItem.getProperty**
 - Retrieves value from the XML Cache
- **document.thisItem.setProperty**
 - Sets value to XML Cache
 - HTML control is not updated
- **window.handleItemChange(property, value)**
 - Sets value to XML Cache
 - Updates HTML control

Copyright © 2014 Aras All Rights Reserved.

aras.com

Setting and Getting Data

Data can be obtained and manipulated in several ways using a Client Method.

this.value

Obtains data from the HTML Form (not the Item cache) and is useful in field onChange events to obtain the changed value of the current field.

document.thisItem.getProperty

Obtains data from the Item cache (not the Form control).

document.thisItem.setProperty

Assigns data to the Item cache (not the HTML Form).

window.handleItemChange

Each Aras Innovator Form contains a predefined JavaScript window function named *handleItemChange* that will set the value of a property in the cache as well as update the HTML control programmatically. You can use this function in your form client Methods to update Form fields and the Item cache using a single operation.

Creating a Field Event Method



■ onClick (Button) Event Example

```
//Get the value of cost property
var cost = document.thisItem.getProperty("_cost", "0");

var taxcost = +cost + (+cost * 0.0625);

//Set calculated value to total_taxed property
handleItemChange("_total_taxed", taxcost);
```

Calculate Tax

Total Taxed Cost:

Copyright © 2014 Aras All Rights Reserved.

aras.com

Creating a Field Event Method (onClick)

This example shows a simple calculation to obtain a Taxed Cost of a Design Request (assumes tax rate is 6.25 percent). A button has been added to the form to show the calculated result:

Cost	Patent?	Completion Date
<input type="text"/>	<input type="checkbox"/>	<input type="text"/>
Classification	State	Thumbnail Select an image...
<input type="text"/>	<input type="text"/>	
Customer	<input type="text"/>	
<input type="text"/>	Calculate Tax	Total Taxed Cost: <input type="text"/>

The Method above is then added to the onClick Field Event of the Button control:

Field Event					
Actions ▾ Create Related New Edit Delete Search Lock Unlock Refresh Find Find Previous Hide Search Criteria					
Name	Method Type	Ver	execution_allowable	Comments	Event
CalculateTax	JavaScript	World			onClick

The Method obtains the value of the cost property from the context Item. The taxed cost is then calculated using simple JavaScript operators and assigned to the total_taxed property using the handleItemChange function.

Creating a Field Event Method



■ onChange Event Example

```
producttype = this.value;
if (producttype=="TB") {
    window.handleItemChange("_patent_required", "1");
}
```

The screenshot shows a user interface with two main input fields. On the left, there is a label 'Patent?' followed by a checkbox which is checked. On the right, there is a label 'Product Types' followed by a dropdown menu that has 'Tablet' selected. Below the form, a black bar contains the text 'Copyright © 2014 Aras All Rights Reserved.' and 'aras.com'.

Creating a Field Event Method (onChange)

This example will set the Patent required field to true (boolean "1") on a Design Request if the user selects the Product Type of Tablet (list value = "TB") from the Form. The Method is associated with the onChange event of the Product Type field:

Field Event						
Actions ▾		Pick Related	+	X	Excel	Word
	Name	Method Type	Ver	execution_all..	Comments	Event
	_SetPatent	JavaScript	1	World		onChange

When capturing data from a Text field or Dropdown type, you can use `this.value` in a field event onChange Client Method to capture the changed value. The Aras Item cache is updated when the Text field loses focus, so calling `document.thisItem.getProperty` in the onChange event will return the original field value (from the Item cache).

Creating a Form Event Method



- Hiding/showing a control:

```
var country = document[thisItem.getProperty("country")];
var field = getFieldByName("address_state");
if (country!="United States") {
    field.style.visibility='hidden';
}
else {
    field.style.visibility='visible';
}
```

	Name	Method Type	Ver	execution_all...	Comments	Event	Sort Ord...
	HideControl	JavaScript		World		onFormPopulated	

Copyright © 2014 Aras All Rights Reserved. aras.com

This example shows how a Form event can be used to alter the user interface dynamically at runtime.

The **onFormPopulated** event is triggered each time the Form is refreshed on the screen. In this example the field “address_state” will be hidden if the country is not “United States”. Standard style attributes can be set in the client Method to alter the display and characteristics of the HTML Form dynamically.

When you create a Form in Aras Innovator you are also generating an HTML page that contains named controls. You can access those controls to change style or visibility on the control using a client Method with JavaScript. A standard method is provided on each Aras form named *getFieldByName* which will return the DIV element representing the control (and label) on the form. In this example, if the country property is not set to the “United States” then the address_state (DIV) on the Form is hidden from view.

This method can then be associated with the onFormPopulated event of a Form. Note the reference to the context Item uses the **document.thisItem** keyword.

To access the individual HTML field elements within a tearoff window form you can also use the following reference:

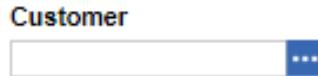
```
var f = document.getElementById("MainDataForm").address_state;
f.style.visibility='hidden';
```

OnSearchDialog Overview

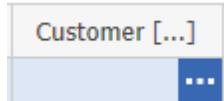


- Client Event that is triggered when a user:

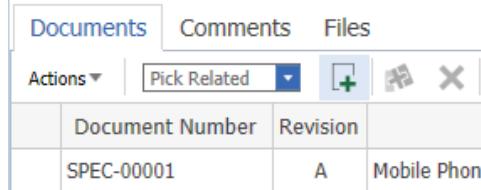
- Presses Ellipsis Button to Search for Item



- Presses F2 Lookup Key on Searchable Field



- Adds a Related Item to a Parent Item



Copyright © 2014 Aras

All Rights Reserved.

aras.com

OnSearchDialog Overview

The OnSearchDialog event is available to respond to the following user actions:

- User presses the ellipsis [...] button or F2 lookup key on a form field to display an Item search grid.
- User adds a new *Pick Related* relationship and the related Item search grid is displayed.

Search Options



- Prefill Search Criteria in the Item Search Grid

The screenshot shows a grid interface with columns: Document Number, Revision, Name, Type, and State. A red circle highlights the 'State' column header 'Released'. Below it, a row shows data: SPEC-00004, A, Mobile Phone Specification, Specification, Released.

	Document Number	Revision	Name	Type	State
	SPEC-00004	A	Mobile Phone Specification	Specification	Released

- Create custom search result lists based on business rules

The screenshot shows a grid interface with columns: Product Number, Name, and Description. It lists three printer models: DesignJet 2000 Large Format Printer, DeskJet 400 Series, and LaserJet 1200 Retail Series, each with its description.

	Product Number	Name	Description
	DesignJet 2000 Large For...	Large Format Printer	Commercial high capacity large format
	DeskJet 400 Series	Inkjet Printer	Low cost inkjet printer for consumer and bus...
	LaserJet 1200 Retail Series	LaserJet Retail Printer	Low cost retail printer

Copyright © 2014 Aras

All Rights Reserved.

aras.com

Search Options

When any of the actions described on the previous page occurs you can:

- Prefill the Simple Search criteria in the Item search grid – this criteria can be set to be read-only (cannot be changed by the user).
- Create a custom search result set based on business logic. The user is only able to select the Items that are defined by the custom logic. The search criteria bar is hidden and cannot be displayed (see Appendix for information on creating custom result sets).



Predefining Search Criteria

- Choose properties to provide criteria
- Supply search criteria for properties
- Determine change access
 - Fixed – user cannot change default criteria supplied
 - Not Fixed – default is provided but user can override

Predefining Search Criteria

To predefine search criteria, you must decide which columns (properties) will be affected and what criteria you will prefill in the columns.

Criteria provided can be fixed so that a user is not able to change it.

In this unit, we will create a Method that requires that predefines the search criteria for 2 properties. A user must have an email address in their User record and also be managed by "Peter Smith

Creating the Search Method



- Client-side JavaScript Method only
- Create JavaScript Object and assign filter values
- Associate Method with OnSearchDialog event
- Syntax:

```
//create new Object
var Filter = new Object();
//Assign Filter
Filter["state"]={filterValue:"Released", isFilterFixed:true};
return Filter;
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

Creating the Search Method

The search Method must be defined as client-side JavaScript. The Method creates a JavaScript object that is used to set the criteria for the grid before it is presented to the user. Using JavaScript Object Notation (JSON) you can indicate filter values and whether the filter can be changed by a user at runtime.

To Create the Search Method

1. Create a new Search Method that is of type Client.
2. Create a new JavaScript Object in the Method and assign it to a variable.
3. Create assignment "filters" by supplying the property name as an array element. The Aras client expects an object array where each element is the name of property (grid column). Two attributes are available to set criteria and read only status in each element (*filterValue*, *isFilterFixed*)
4. Indicate the value of the search criteria using the *filterValue* setting.
5. Indicate whether the value can be changed by a user with the *isFilterFixed* setting (true or false).
6. Return the filter object variable to the client.

In this example, a Method named *SearchCriteria* is created with code as shown above. This method will predefine the Document search to only show Released Document Items. The user will be unable to change the state criteria. JavaScript object attributes are case-sensitive. Make sure to enter the attributes as shown in the example.

The screenshot illustrates the process of associating a method to an event in the Aras Innovator client. It consists of two main windows:

- Properties Dialog (Top Window):** Shows a list of properties for an item type. The selected property is '_attachment'. A context menu is open over this property, with the option 'View "Properties"' highlighted (circled 1).
- Event List (Bottom Window):** Shows a table of events. A new row is being created, indicated by a green plus sign icon. The 'Name' column is populated with 'ReleaseDocumentSearch'. The 'Method Type' column is set to 'JavaScript'. The 'Event' column is set to 'OnSearchDialog' (circled 3).

Aras Corp logo is visible in the top right corner of the client interface.

Associating the Method to the OnSearchDialog Event

The search Method is then associated with the OnSearchDialog event on a property. Note that the property must be of data type *Item*. Only *Item* data type properties provide the ability to search on another Item in the Aras Innovator client.

To Define the OnSearchDialog Event

1. Open the desired ItemType and locate a property that is exposed on a form and has the *Item* data type. Right click to view the context menu and select *View "Properties"*.
2. Add a new Event relationship on the property form and select the search Method.
3. Choose OnSearchDialog as the event type.

In this example the *_attachment* property has been selected in the *Change Request* ItemType. The *ReleaseDocumentSearch* Method is assigned to the OnSearchDialog event.



Summary

In this unit you learned how to apply client Methods to customize a solution.

You should now be able to:

- ✓ Customize New Item Functionality
- ✓ Describe the Layout of the the Aras Innovator Main Window
- ✓ Describe the Layout of the the Tear-off Window
- ✓ Develop Form and Field Validation
- ✓ Predefine Search Criteria



Lab Exercise

Goal:

Be able to create and use client Methods with the appropriate client events to customize a solution that uses the Aras Innovator client.

Scenario:

In this exercise, you will create several client Methods to extend the behavior of the Innovator client. You will then associate these Methods with appropriate client events to provide custom business logic in a solution.

Set Shipping Date on New Sales Order

- When a user creates a new Sales Order the Shipping Date should be set to today's date by default.

A date property is always set using a "neutral" date string and must be formatted as "yyyy-mm-dd". Example: "2010-06-10"

You can use the JavaScript `Date()` function to create a new Date object as shown below:

```
var today = new Date();
var dd = today.getDate();
var mm = today.getMonth() + 1; //January is 0!
var yyyy = today.getFullYear();
if(dd<10) {dd = '0' + dd;}
if(mm<10) {mm = '0' + mm;}
today = yyyy + '-' + mm + '-' + dd;
```

The date should be prefilled on the client before the form is loaded. Which ItemType client event allows you to change the Context Item of a new Item after it has been returned from the server -but before the user sees the form? _____ . Remember to return the modified context item to the client in your Method.

Change the Value of a Property

- If user checks the **Allow Multiple Shipments** box on a Sales Order form the Shipping Method should change to UPS, otherwise the Shipping Method should be set to Federal Express.

A screenshot of a Sales Order form titled "Shipping Info". It contains the following fields:

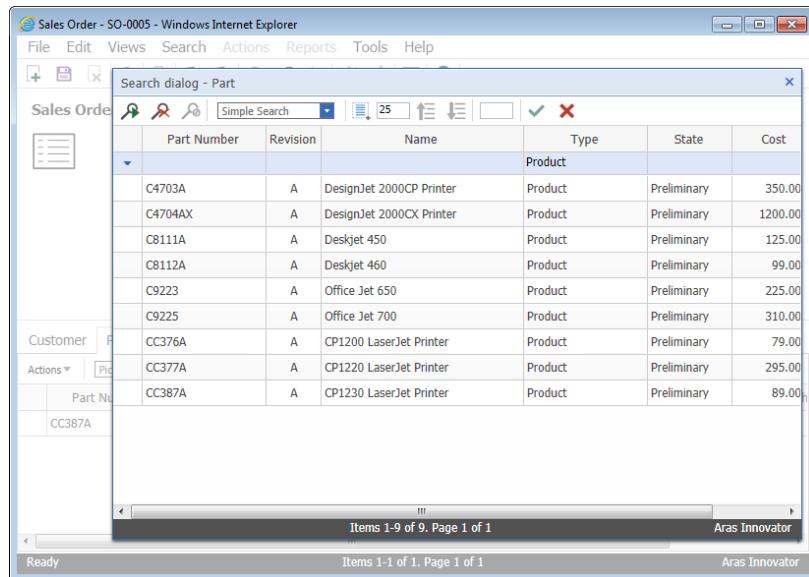
- Shipping Method:** A dropdown menu currently set to "UPS".
- Ship Date:** A text input field containing "6/19/2013".
- Allow Multiple Shipments?**: A checkbox that is checked.

Use the `getProperty` method to get the value of the `multiple_shipment` property on the current item. What syntax allows you to access the context item on a Form or Field event?

If `multiple_shipment` is checked (true) then use a function described in this unit to change the `shipping_method` property value to "UPS". Otherwise set it to "Federal Express".

Filter the Parts List

3. A user should only be able to select from Parts with the **Product** Type Classification when they add a Part to the Sales Order. You will need to create a client filter Method for the `related_id` property of the Sales Order Part relationship ItemType.



Create a client Method that creates a filter object to set the `classification` property to "Product". The user should not be able to change this value.

Edit the "Sales Order Part" relationship ItemType and attach this Method to the `onSearchDialog` event of the `related_id` property.

This page intentionally left blank.



Unit 12 Using Federated Items/Properties

Overview: In this unit you will learn how to use and store data in an external resource while still presenting the information to the user in the standard client user interface. Federation allows you to integrate an Aras Innovator solution with data that does not reside in the Innovator database.

- Objectives:**
- ✓ Reviewing Federated Items and Properties
 - ✓ Configuring a Federated Property
 - ✓ Populating a Federated Property Value
 - ✓ Saving a Federated Property Value
 - ✓ Configuring a Federated ItemType



Federated Items and Properties

- Allow data to be accessed and shared with other systems
- Federated Properties display in grids and forms but the value of the data may come from another system
- Federated Items have at least some of their properties stored in other database or system
- Data is provided to properties using server events

Implementation Type

Single Item Poly Item Federated Item

Federated Items and Properties are useful for allowing integration of data from other systems to be incorporated into a solution.

When you mark an ItemType as Federated you are indicating that at least one property of the Item will not be stored in the Innovator database. Remember that each ItemType that is created in Aras Innovator represents a table in the database and each property represents a column.

With federation, you can seamlessly display and edit data and decide when and where that data should be retrieved or stored.

Federated Options



■ Mixed System

- Some of the data is stored in a remote system
- ItemType is Federated
- Certain properties are marked as Federated
- Federated properties do not have a column in the ItemType database table

■ Separate System

- All data is stored in a remote system
- ItemType is Federated
- All properties of the ItemType are Federated
- No data rows are created in the ItemType database table

You have two options when designing federation into a solution. Should some of the data reside in the Innovator database, or should all of the data be located in an outside resource.

In the “separate system” option no rows are created in the Innovator database and all data population must be accomplished with custom logic.

In a “mixed” environment some basic data may be stored in the system (e.g. Item ID , created_on, created_by) and data which currently resides in an outside resource can be retrieved (and updated) but not saved to the Innovator database.

Reviewing Possible Data Sources



- Web Services
 - Web Service Configuration
- SQL
 - External Relational Database
- IOM
 - Using API calls to remote system

In this unit, we will discuss and use an external Microsoft Access database to retrieve and update data, however you can use any resource accessible through a .NET program.

Working with a Federated Property Value - Mixed



- ItemType server events are available to populate data:
 - onAfterGet – used to populate Federated property
 - onAfterDelete – removes remote data when Item is deleted
 - onBeforeAdd – adds remote data to a remote system when Item is added
 - onBeforeUpdate – changes remote data on an Item edit

To support federation, you respond to a set of server events on the ItemType. These events allow you intercept the AML messages being passed to and from the Innovator Server to create, retrieve, update and delete external data.



Working with a Federated Property - Separate

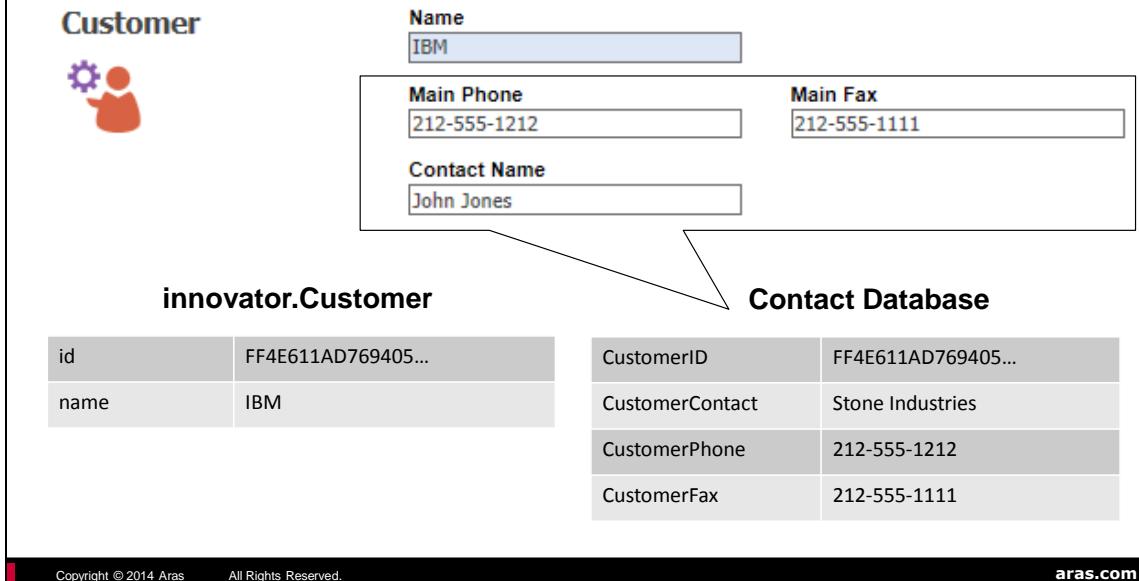
- All Item logic is replaced with developer code to obtain and return data to the remote system using:
 - onGet – replaces the standard retrieve logic
 - onAdd – replaces the standard add logic when a new Item is added
 - onDelete – replaces the standard delete logic
 - onUpdate – replaces standard edit logic

If no data is required to be stored in the Innovator database you are required to replace the current retrieve/update logic with your own. These events completely replace the add, edit, get and delete standard behavior in Innovator. You are responsible for creating appropriate AML messages to and from the server.

Federated Example



■ Customer -> Contact Database



Customer Example

The standard solution database contains an ItemType named Customer that stores information about each customer including contact data. In this example, the customer contact information will be stored and retrieved from a federated Microsoft Access database. This could be an existing database with legacy data or a database that is used by other applications and shared by Aras Innovator.

The **Customer** ItemType will become a "mixed" Federated Item with the following data stored in an Microsoft Access Contact Database. The **CustomerID** will use the Aras Innovator Item **id** property to maintain a link between databases.

The table **Contact** in the Microsoft Access Database has been defined as:

Column Name	Column Details
CustomerID	WideString(32), Primary Key
Customer Contact	WideString(50)
CustomerPhone	WideString(20)
CustomerFax	WideString(20)

Configuring a Federated ItemType

The screenshot shows the 'ItemType' configuration page. At the top, there is a border labeled 'Implementation Type' containing three radio buttons: 'Single Item' (unchecked), 'Poly Item' (unchecked), and 'Federated Item' (checked). Below this, the main configuration area has several sections:

- Name:** Customer
- Singular Label:** Customer
- Plural Label:** Customers
- Small Icon:** Select an image...
- Large ICON:** Select an image...
- History Template:** A dropdown menu with options: 'None', 'Table', 'Automatic', and 'Default'.
- Default Structure View:** A dropdown menu with options: 'None', 'Search', and 'List'.
- Search:** Includes 'Auto Search' (checkbox) and 'Default Page Size' (25).
- Max Records:** A dropdown menu with options: 'None', '100', '200', and '500'.
- Implementation Type:** A border containing the same three radio buttons as the top section: 'Single Item', 'Poly Item', and 'Federated Item' (checked).

At the bottom left of the configuration area, there is a 'Class Structure' section with a 'Show Parameters Tab' button and a dropdown menu set to 'When Populated'. The bottom right corner of the configuration area contains the 'aras.com' logo.

To Configure a Federated Item

Edit the ItemType and select the Federated Item radio button in the Implementation Type border.

Note

Selecting this option has no affect on system behavior but is simply used as a “marker” or reminder that some or all of the data for this Item will be stored in a remote location. Associating Methods with the server events discussed on the previous pages dictates if an item or property is federated.

Configuring a Federated Property



- Choose Federated Data Type

The screenshot shows a software interface titled 'Properties'. At the top, there's a toolbar with icons for Actions (dropdown), No Related (dropdown), Create (+), Delete (X), Export (Excel), Import (Word), and Lock (padlock). Below the toolbar is a table with three columns: Name, Label, and Data Type. The table contains four rows of data:

Name	Label	Data Type
contact_name	Contact Name	Federated
main_fax	Main Fax	Federated
main_phone	Main Phone	Federated

At the bottom of the interface, there's a footer bar with the text 'Copyright © 2014 Aras All Rights Reserved.' and 'aras.com'.

To Configure a Federated Property

Choose Federated from the Data Type dropdown list.

Choosing this option will remove the database column for this property from the Customer ItemType table in the Innovator database.

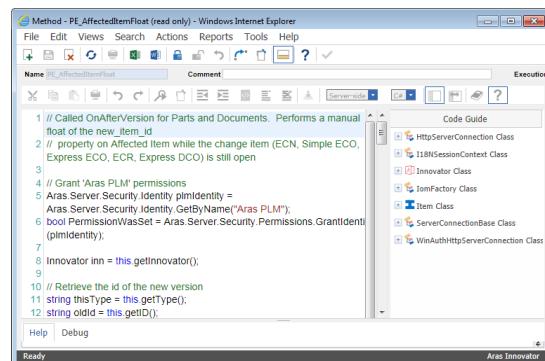
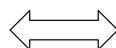
In this example, the contact_name, main_phone and main_fax numbers will no longer be stored in the Aras database.

Adjusting Method Configuration



- Editing Method-config.xml
- Reviewing Referenced Assemblies
- Adding a Using or Import Statement

```
<?xml version="1.0?>
<!-- name = method-config.xml
   purpose = Provides limited customization for Innovator server method
            interface, code generation and compilation phase.
   (c) Copyright by Aras Corporation, 2004-2010.
-->
<!DOCTYPE MethodConfig [ 
<!ELEMENT MethodConfig (ReferencedAssemblies, Support*, Template)
<!ELEMENT ReferencedAssemblies (#PCDATA)
<!ELEMENT Support (#PCDATA)
<!ELEMENT Template (#PCDATA)
<!ATTLIST Template
  direction CDATA #REQUIRED
  template CDATA #REQUIRED
  compiler CDATA #REQUIRED
  >
<!ELEMENT Template (#PCDATA)
<!ATTLIST Template
  name CDATA #REQUIRED
  line_number_offset CDATA #REQUIRED
  >
]
<MethodConfig>
<ReferencedAssemblies>
<name>System.dll</name>
<name>System.Data.dll</name>
<name>System.Web.dll</name>
<name>System.Core.dll</name>
<name>System.Configuration.dll</name>
<name>System.ComponentModel.dll</name>
<name>System.Innovator.Core.dll</name>
<name>${Stringpath}CoreCS.dll</name>
<name>${Stringpath}SIConnector.dll</name>
<name>${Stringpath}ConversionManager.dll</name>
<name>${Stringpath}ConversionBase.dll</name>
<name>${Stringpath}WebServiceProxy.dll</name>
</ReferencedAssemblies>
```



Copyright © 2014 Aras All Rights Reserved.

aras.com

Adjusting Method Configuration

The source code that you enter in an Aras Method Item is actually inserted into a method configuration template which defines how a Method Item is compiled. You normally do not need to make any modifications to this file unless you plan to add the ability to call external libraries (DLL) from an Aras Method.

The method-config.xml file is located in the Innovator/Server subdirectory and contains:

- **<ReferencedAssemblies>** - lists the DLL library files available to every Aras Method.
- **<Support>** - lists the supported language templates. These should not be modified.
- **<Template>** - defines each language template noted in the <Support> tags.

Note that any changes you make to the method-config.xml file are used immediately by the server (does not require a restart). You should always keep a backup of the original config file should it be necessary to restore back to the original configuration. The examples presented in this unit use the System.Data.OleDb namespace from *System.Data.dll* already installed on the server. To make the Aras Methods easier to read in this unit you can add the following using (or Imports) statement to the C# (or VB) templates:

```
<Template name="CSharp" line_number_offset="39"><![CDATA[
using System.Data.OleDb;
or
<Template name="VBScript" line_number_offset="39"><![CDATA[
Imports System.Data.OleDb
```

Populating a Federated Property



onAfterGet ▾

```

OleDbConnection conn
= new OleDbConnection( ← Get External Connection
    "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\\AccessDB\\Training.mdb");
conn.Open();
for (int i = 0; i < this.getItemCount(); i++) { ← Check for Collection
    Item idx_itm = this.getItemByIndex(i); ← Get Each Item
    string id = idx_itm.getID();
    string sql = "select * from Contact where CustomerID = '{0}'";
    sql = String.Format(sql, id);
    OleDbCommand cmd = new OleDbCommand(sql, conn);
    OleDbDataReader reader = cmd.ExecuteReader(); ← Retrieve Data
    if (reader.Read()) {
        idx_itm.setProperty("contact_name", reader["CustomerContact"].ToString());
        idx_itm.setProperty("main_phone", reader["CustomerPhone"].ToString());
        idx_itm.setProperty("main_fax", reader["CustomerFax"].ToString());
    }
}
Assign Data to Item
conn.Close();
return this;

```

Copyright © 2014 Aras All Rights Reserved.

aras.com

Populating a Federated Property

To populate a federated property use the onAfterGet server event.

Note that retrieving data may return more than one database row so your Method code should expect and handle this situation. Remember that federated property data can appear in relationship grids and the main search grids.

In this example, the Customer Item is being populated with external data from a Microsoft Access Contact database. The setProperty method is used to set the appropriate values based on data retrieved from the external database table.

VB.NET

```

Dim conn As New OleDbConnection(
    "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\\AccessDB\\Training.mdb")
conn.Open()

For i As Integer = 0 To Me.getItemCount() - 1

    Dim idx_itm As Item = Me.getItemByIndex(i)
    Dim id As String = idx_itm.getID()

    Dim sql As String = "select * from Contact where CustomerID = '{0}'"
    sql = [String].Format(sql, id)

    Dim cmd As New OleDbCommand(sql, conn)
    Dim reader As OleDbDataReader = cmd.ExecuteReader()
    If reader.Read() Then

```

Developing Solutions

```
    idx_itm.setProperty("contact_name", reader("CustomerContact").ToString())
    idx_itm.setProperty("main_phone", reader("CustomerPhone").ToString())
    idx_itm.setProperty("main_fax", reader("CustomerFax").ToString())
End If
Next

conn.Close()
Return Me
```

Adding New Data to the Remote System



onBeforeAdd

```

OleDbConnection conn
= new OleDbConnection(      ← Get External Connection
    "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\\AccessDB\\Training.mdb");
conn.Open();
string id = this.getID();
string contact = this.getProperty("contact_name", "");
string phone = this.getProperty("main_phone", "");      ← Get Property Values
string fax = this.getProperty("main_fax", "");
string insertsq1 =
    "insert into Contact (CustomerID, CustomerContact, CustomerPhone, CustomerFax) values ('{0}', '{1}', '{2}', '{3}')";
insertsq1 = String.Format(insertsq1, id, contact, phone, fax); ← Insert Record

OleDbCommand cmdUpdate = new OleDbCommand(insertsq1, conn);
int rowsAffected = cmdUpdate.ExecuteNonQuery();
conn.Close();
return this;

```

Copyright © 2014 Aras All Rights Reserved.

aras.com

Adding New Data to Remote System

To add a new data row to an external resource you can use the onBeforeAdd server event.

In this example a new Contact row will be created in the external database using a sql insert statement. The .NET String.Format method is used to provide substitution of values obtained from Innovator to be included in the sql insert statement.

VB.NET

```

Dim conn As New OleDbConnection(
    "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\\AccessDB\\Training.mdb")
conn.Open()

Dim id As String = Me.getID()
Dim contact As String = Me.getProperty("contact_name", "")
Dim phone As String = Me.getProperty("main_phone", "")
Dim fax As String = Me.getProperty("main_fax", "")

Dim insertsq1 As String = "insert into Contact (CustomerID, CustomerContact,
CustomerPhone, CustomerFax) values ('{0}', '{1}', '{2}', '{3}')"
insertsq1 = [String].Format(insertsq1, id, contact, phone, fax)

Dim cmdUpdate As New OleDbCommand(insertsq1, conn)
Dim rowsAffected As Integer = cmdUpdate.ExecuteNonQuery()

conn.Close()

Return Me

```

Updating Changes to the Remote System



onBeforeUpdate ▾

```

OleDbConnection conn
= new OleDbConnection(
    "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\\AccessDB\\Training.mdb");
conn.Open();
string id = this.getID();
string contact = this.getProperty("contact_name", "");
string phone = this.getProperty("main_phone", "");
string fax = this.getProperty("main_fax", "");
string updatesql =
"update Contact SET CustomerContact='{0}',CustomerPhone='{1}',CustomerFax='{2}'"
    where CustomerID='{3}'";
updatesql = String.Format(updatesql, contact, phone, fax, id);

OleDbCommand cmdUpdate = new OleDbCommand(updatesql, conn); ← Update Record
int rowsAffected = cmdUpdate.ExecuteNonQuery();
conn.Close();
return this;

```

Copyright © 2014 Aras All Rights Reserved.

aras.com

Updating Changes to the Remote System

Like the previous example, another sql string is constructed to update an existing row in an external table. The onBeforeUpdate server event is used to trigger this Method when an existing Item is saved.

VB.NET

```

Dim conn As New OleDbConnection(
    "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\\AccessDB\\Training.mdb")
conn.Open()

Dim id As String = Me.getID()
Dim contact As String = Me.getProperty("contact_name", "")
Dim phone As String = Me.getProperty("main_phone", "")
Dim fax As String = Me.getProperty("main_fax", "")

Dim updatesql As String = "update Contact SET
CustomerContact='{0}',CustomerPhone='{1}',CustomerFax='{2}' where CustomerID='{3}'"
updatesql = [String].Format(updatesql, contact, phone, fax, id)

Dim cmdUpdate As New OleDbCommand(updatesql, conn)
Dim rowsAffected As Integer = cmdUpdate.ExecuteNonQuery()

conn.Close()

Return Me

```

Removing Data from the Remote System



onAfterDelete

```
OleDbConnection conn ← Get External Connection
= new OleDbConnection(
    "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\\AccessDB\\Training.mdb");
conn.Open();

string id = this.getID();

string deletesql ="delete from Contact where CustomerID='{0}'";

deletesql = String.Format(deletesql, id); ← Delete Record

OleDbCommand cmdUpdate = new OleDbCommand(deletesql, conn);
int rowsAffected = cmdUpdate.ExecuteNonQuery();

conn.Close();
return this;
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

Removing Data from Remote System

The onAfterDelete server event is used in this example to allow deletion of data from an external resource using the sql delete statement.

VB.NET

```
Dim conn As New OleDbConnection(
    "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\\AccessDB\\Training.mdb")
conn.Open()

Dim id As String = Me.getID()

Dim deletesql As String = "delete from Contact where CustomerID='{0}'"

deletesql = [String].Format(deletesql, id)

Dim cmdUpdate As New OleDbCommand(deletesql, conn)
Dim rowsAffected As Integer = cmdUpdate.ExecuteNonQuery()

conn.Close()
Return Me
```



Summary

In this unit you learned how to use Federated Items and Properties to integrate data external to the Innovator database.

You should now be able to:

- ✓ Configure a Federated Property
- ✓ Populate a Federated Property Value
- ✓ Save a Federated Property Value
- ✓ Configure a Federated ItemType



Lab Exercise

Goal:

Purchase Order data for each Sales Order needs to be maintained in a separate database. The following table has been established in the Microsoft Access Training database:

Table Name: PurchaseOrder		Primary Key Field(S): SalesOrderID;					
Field Name	Field Type	Size	Field No.	Required	Precision	Display Name	
SalesOrderID	WideString	32	1	No	0	SalesOrderID	
Shipper	WideString	20	2	No	0	Shipper	
PONumber	WideString	20	3	No	0	PONumber	

You must add (or update) this table with the data entered by the user on the Sales Order. Use the following Sales Order properties to map the data to this table:

Aras Property	Microsoft Access Column
id	SalesOrderID
shipping_method	Shipper
po_number	PONumber

Steps

1. Make a copy of the 3 Methods provided in your student database to federate a Customer property (TRN_AddFederatedData, TRN_UpdateFederatedData, TRN_GetFederatedData). These are the examples shown in this unit.
2. Adjust the Methods to use the Microsoft Access Columns and Aras properties described in the table above.
3. Attach each Method to an appropriate Server Event on the Sales Order ItemType.
4. Create a new Sales Order and test the configuration. Why will these Methods only work for new Sales Orders? _____

Federate Properties

5. Once the Methods are running successfully, you can return to the Sales Order ItemType and set the datatype of the **shipping_method** and **po_number** properties to *Federated*. What happens to any existing data in the Aras database? _____

This page intentionally left blank.



Unit 13 Integrating a .NET Application

Overview: In this unit you will learn how to install and reference the IOM in a .NET application to provide Innovator services to an outside application. As an alternative, you can also send standard SOAP message requests to Innovator Server and process the responses.

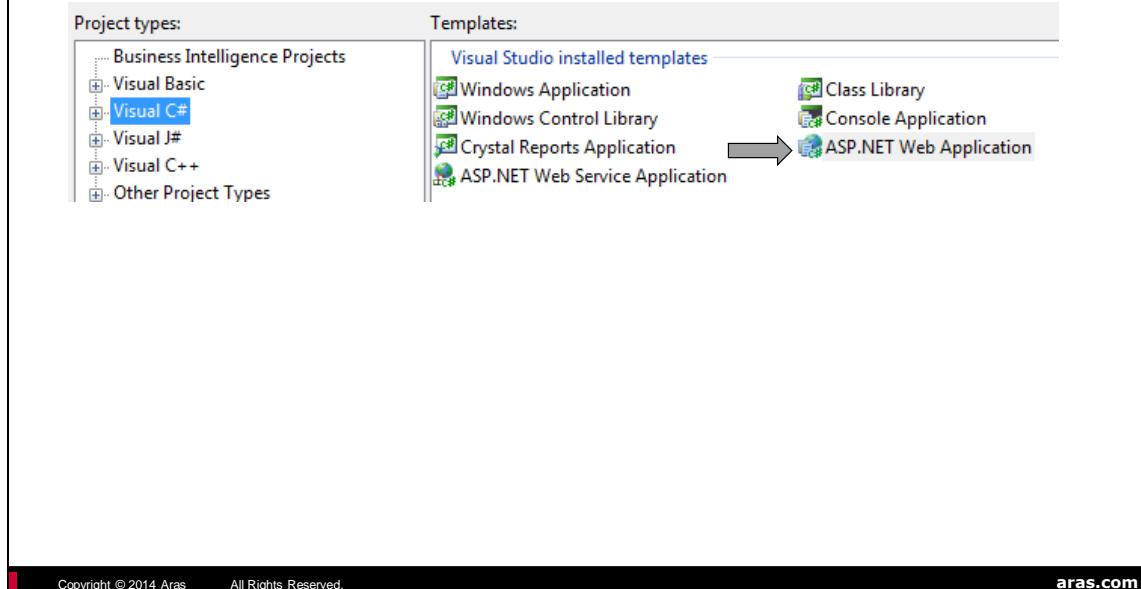
Objectives:

- ✓ Using the IOM in a Visual Studio Project
- ✓ Calling Methods from a .NET Application

Using the IOM in Visual Studio



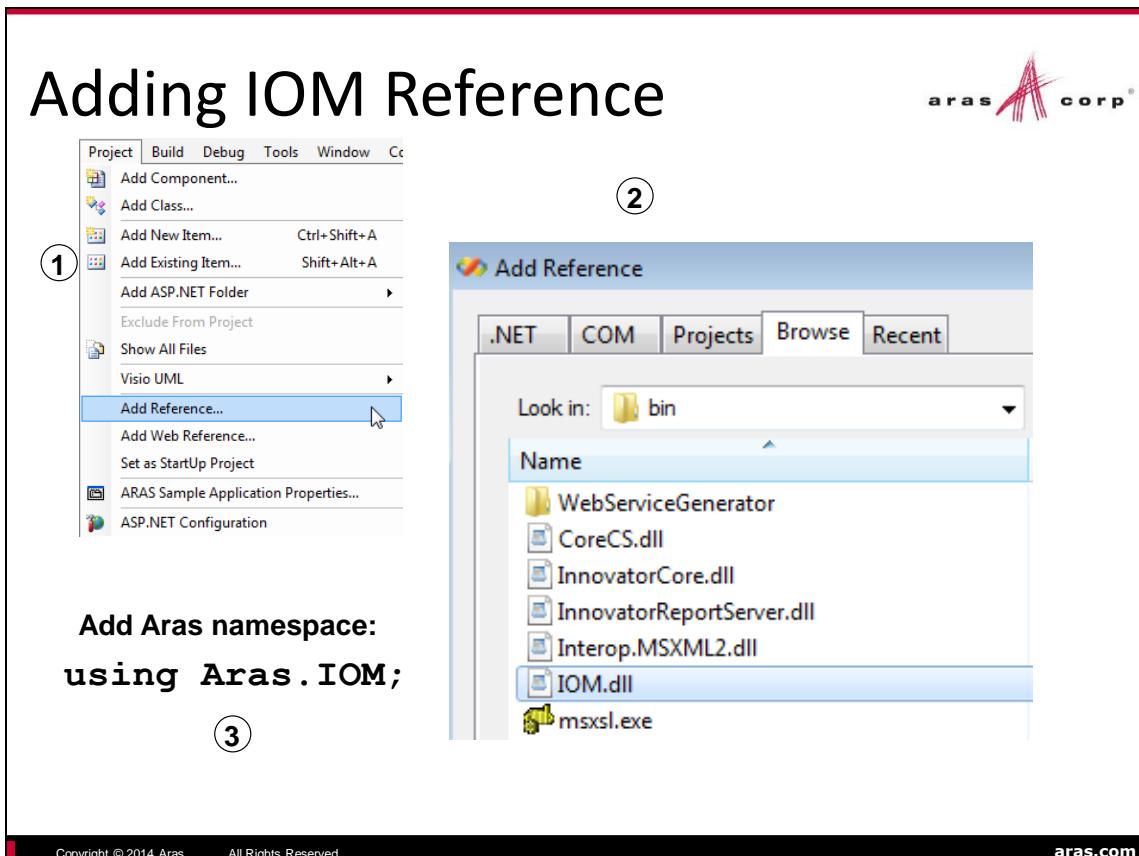
■ Create a new ASP .Net Web Application



So far in this course we have focused on using the Aras Innovator standard client as a way for an end user to gain access to the system.

In this unit, we look at alternate ways to interact with the Innovator Server by creating an ASP.NET application that uses the IOM to interact with the system.

In this example, we will use Visual Studio 2010 to create a simple ASP .NET web application that will interact with the Innovator Server.



To Add IOM Reference to a Visual Studio Project

To gain access to the classes and methods we have discussed in this course requires adding a reference to the IOM.dll.

The IOM SDK has been created for use in developing external applications that connect to Aras Innovator. This SDK specifically contains .NET, COM, and Windows RT compatible DLLs, to support all 3 types of connections. (Aras Version 10 SP3 and above).

Previously, application developers would copy the IOM file from the Aras Innovator code tree when developing applications. The new SDK DLLs eliminate the process of copying the IOM from the Aras server code tree so that Aras can remove all external dependencies from the IOM DLLs.

1. In an open Visual Studio project select Project > Add Reference... from the main menu.
2. Choose the Browse tab and select the IOM.dll from the appropriate IOM SDK directory (this example uses .NET) .
3. Finally, add the namespace Aras.IOM to your project code to gain access to the IOM classes and methods.



Establishing Connection

- Define the login credentials
 - User Id, Password, Database, Server URL
- Establish a server connection
 - Using the IomFactory class
- Login to the server using the connection
 - Using the HttpServerConnection class
- Create Innovator instance
 - Using the IomFactory class

This example shows a small project that will make a connection to the student development server.

The Visual Studio ASP.Net project uses the Global.asax page to define global settings initialized when the session begins. This example uses that page to define the Innovator variable that will be available to any page in the application.

C#:

```
public class Global : System.Web.HttpApplication
{
    public static Innovator inn = null;
    private HttpServerConnection conn = null;
    void Session_Start(object sender, EventArgs e)
    {
        // Code that runs when a new session is started
        //Define credentials
        string url = "http://localhost/Innovator10";
        string db = "DevelopingSolutions100";
        string user = "admin";
        string pw = "innovator";

        //STEP ONE MAKE A SERVER CONNECTION
        conn =
            IomFactory.CreateHttpServerConnection(url, db, user, pw);

        //STEP TWO LOGIN
```

```
    Item login_result = conn.Login();

    //STEP THREE - INNOVATOR CREATE
    inn = IomFactory.CreateInnovator(conn);
}

}
```

VB:

```
Public Class Global_asax
    Inherits System.Web.HttpApplication
    Public Shared inn As Innovator = Nothing
    Private conn As HttpServerConnection = Nothing

    Private Sub Session_Start(sender As Object, e As EventArgs)
        ' Code that runs when a new session is started
        'Define credentials
        Dim url As String = "http://localhost/Innovator10"
        Dim db As String = "DevelopingSolutions100"
        Dim user As String = "admin"
        Dim pw As String = "innovator"

        'STEP ONE MAKE A SERVER CONNECTION
conn = IomFactory.CreateHttpServerConnection(url, db, user, pw)

        'STEP TWO LOGIN
        Dim login_result As Item = conn.Login()

        'STEP THREE - INNOVATOR CREATE
        inn = IomFactory.CreateInnovator(conn)
    End Sub
End Class
```

Using IOM Item Methods



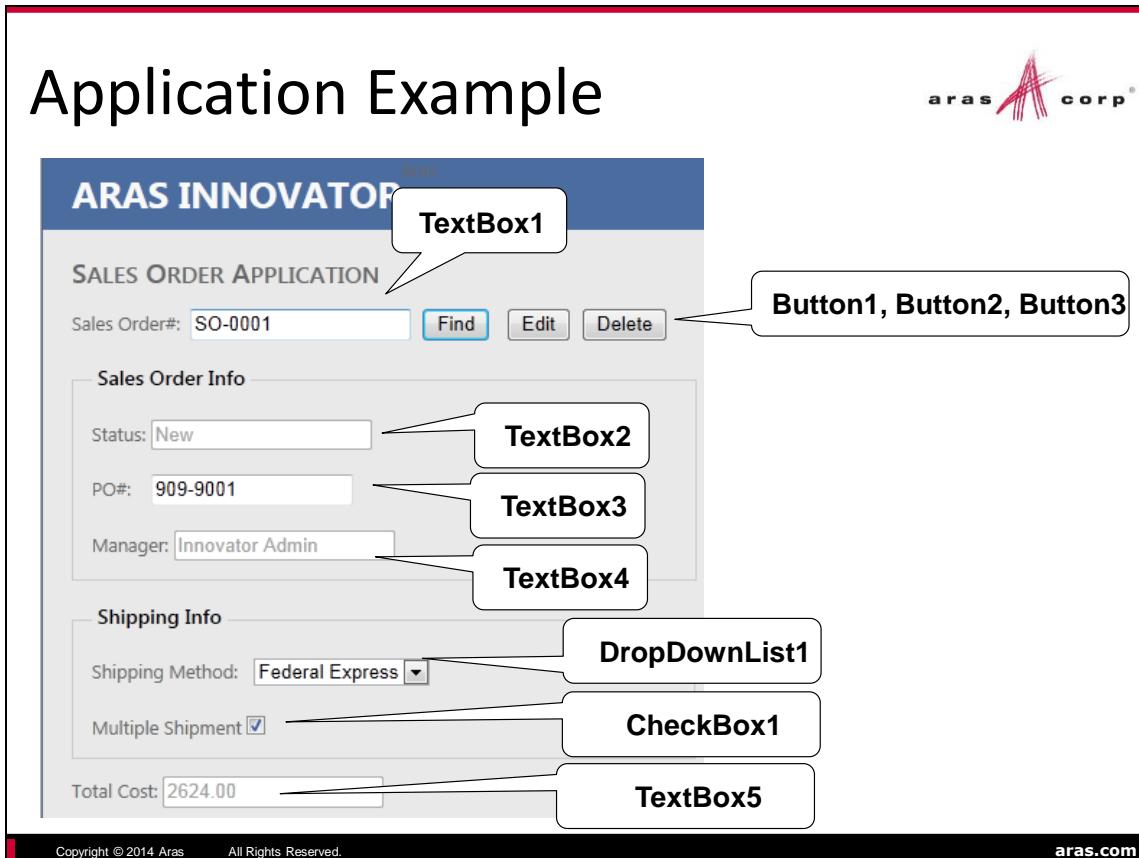
- All IOM and Innovator class methods discussed in class are now available
- Use the Innovator instance to create a new Item
- Microsoft Visual Studio “IntelliSense” automatically completes method syntax as you type

```
itm.setProperty()  
▲ 1 of 2 ▼ void Item.setProperty (string propertyName, string PropertyValue)
```

The IOM.dll assembly contains all of the Item and Innovator class methods we have used in the course.

Once a reference to the Innovator object is established in your program code, you can begin to access and modify Aras Innovator Items.

In addition, Visual Studio "Intellisense" make coding easier as valid options are presented as you create your project code.



Application Example

The following code is placed on the ASP.Net page displayed above using the ASP.Net Button1 click method to retrieve the details of a Sales Order by the `so_number` property:

C#:

```
protected void Button1_Click(object sender, EventArgs e)
{
    Item so = Global.inn.newItem("Sales Order", "get");
    so.setProperty("so_number", TextBox1.Text);
    so = so.apply();

    if (so.isError())
    {
        StatusMessage.Text = so.getErrorString();
        return;
    }
    TextBox2.Text = so.getProperty("state", "");
    TextBox3.Text = so.getProperty("po_number", "");
    DropDownList1.Text = so.getProperty("shipping_method", "");
    TextBox4.Text =
        so.getPropertyAttribute("managed_by_id", "keyed_name");
    string ms = so.getProperty("multiple_shipment");
    CheckBox1.Checked = (ms == "1") ? true : false;
}
```

VB:

```
Protected Sub Button1_Click(sender As Object, e As EventArgs)
    Dim so As Item = [Global_asax].inn.newItem("Sales Order", _
        "get")
    so.setProperty("so_number", TextBox1.Text)
    so = so.apply()

    If so.isError() Then
        StatusMessage.Text = so.getErrorString()
        Return
    End If
    TextBox2.Text = so.getProperty("state", "")
    TextBox3.Text = so.getProperty("po_number", "")
    DropDownList1.Text = so.getProperty("shipping_method", "")
    TextBox4.Text = so.getPropertyAttribute("managed_by_id", _
        "keyed_name")
    Dim ms As String = so.getProperty("multiple_shipment")
    CheckBox1.Checked = If((ms = "1"), True, False)
End Sub
```

The method above uses the Innovator instance variable (inn) established in the Global.asax page on the previous page.

Apply a Server Method

Earlier in this class you created a server Method named **Server_CalculateCost** to calculate the roll up cost of all Part items on a Sales Order item.

You can use the IOM apply method to execute this program and return a result to the solution:

```
Item costItm = so.apply("Server_CalculateCost");
    string totalcost = costItm.getResult();
TextBox5.Text = totalcost;
```

VB

```
Dim costItm As Item = so.apply("Server_CalculateCost")
Dim totalcost As String = costItm.getResult()
TextBox5.Text = totalcost
```

Sending SOAP Messages



- Create a Web request to send AML Soap Message to Innovator Server
- Request should contain valid `HttpWebRequest` containing:
 - Login user
 - Password (encrypted)
 - Database
 - Server URL
- Program should process `HttpWebResponse` to obtain content from Innovator Server

An alternative to using the IOM is to create appropriate SOAP requests that are sent to the Innovator server. You can then process the response requests and decide how to apply the data in your custom application client.

See the *Aras Innovator Programmer's Guide* for examples of creating a properly formatted SOAP request.



Summary

In this unit you learned how to integrate a .NET ASP application with the Innovator server using the IOM or sending SOAP messages.

You should now be able to:

- ✓ Establish a connection to an Aras Innovator Server using a .NET application.
- ✓ Use IOM methods in a .NET ASP application to access or modify items.



Lab Exercise

Goal:

Integrate an ASP.NET application with Aras Innovator.

Scenario:

Build an ASP.NET application that allows a user to perform basic maintenance on a Sales Order. The application should be able to find, edit and delete Sales Orders using IOM methods you have used in the course.

1. Open the ASP.NET project named **SalesOrderApplication** in the Visual Studio Projects folder.
2. Provide the logic for the Edit and Delete buttons to allow a user to modify an existing Sales Order item. What attribute must you set to allow an edit or delete to an Item? _____

This page intentionally left blank.



Unit 14 Using an External Client

Overview: In this unit you will learn how to use an external COM client (Microsoft Excel) to access the services of the Innovator Server. You will populate and allow editing of a spreadsheet by creating a VBA macro that uses the IOM.

- Objectives:**
- ✓ Using a COM Client with the IOM
 - ✓ Creating/Registering a Type Library
 - ✓ Configuring Microsoft Excel
 - ✓ Creating the Excel Macro (VBA)
 - ✓ Establishing a Connection to the IOM
 - ✓ Retrieving Data into Microsoft Excel
 - ✓ Saving Spreadsheet Data to Aras Innovator

Using a COM Client with IOM



- An Innovator SDK COM library is available to allow a client to communicate with the Innovator Server
- IOM.dll is located in the COM folder of the IOM SDK
- Requires registration to create a Type Library file

The client IOM.dll can be used with alternate clients that support the Common Object Model (COM) interface. You are required to create a Type library file and register it which is explained in this unit. The COM version of the IOM library is available in the IOM SDK packaged separately from the Aras Innovator server.

Note

Prior to Aras Version 10 SP3 the COM version of the IOM.dll was available in the Aras server code tree in the Innovator/Client/cbin directory.

Registering the Type Library



- Use command line utility Regasm.exe:

```
>C:\Windows\Microsoft.NET\Framework64\v4.0.30319\  
    RegAsm.exe C:\ARAS_SDK\COM\IOM.dll  
        /tlb /codebase /verbose
```

Copyright © 2014 Aras

All Rights Reserved.

aras.com

Use the regAsm.exe (assembly registration) program provided by the Microsoft .NET framework to define and register the required Type library. This registration process must be performed on any client that will access the COM IOM SDK.

Example Results:

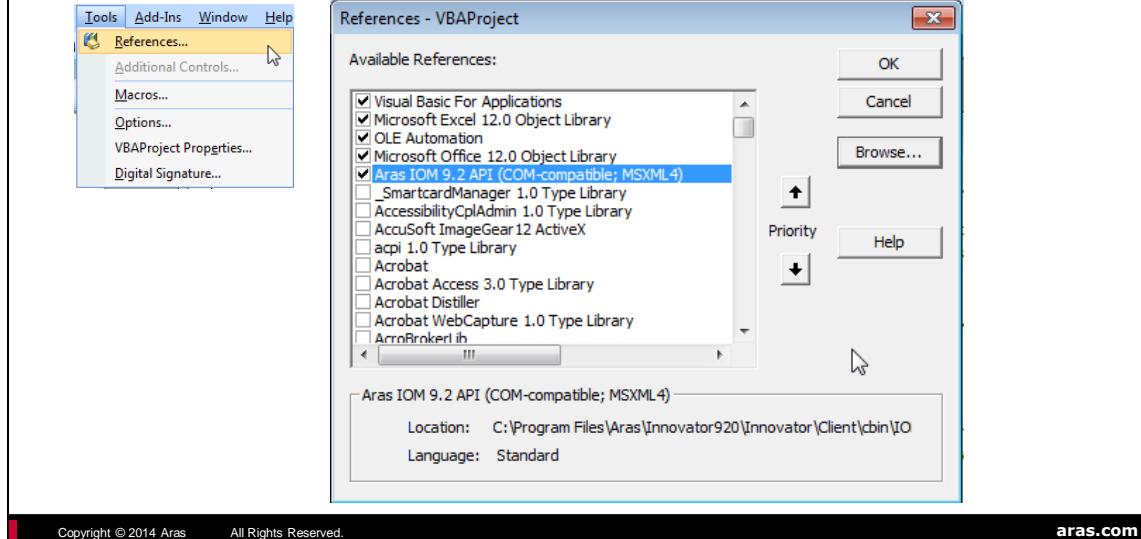
```
Microsoft .NET Framework Assembly Registration Utility version  
4.0.30319.18408 for Microsoft .NET Framework version 4.0.30319.18408  
Copyright (C) Microsoft Corporation. All rights reserved.  
Types registered successfully
```

```
Type 'A' exported.  
Type 'A' exported.  
...  
Assembly exported to 'C:\ARAS_SDK\COM\IOM.tlb', and the type library  
was registered successfully
```

Configuring Microsoft Excel



- Create a new VBA Macro
- Access Tools Menu > References to add Type



To Configure MS Excel

In this example, we will use the MS Excel application as an alternate COM client to the Innovator server.

1. Create a new VBA Macro.
2. Select the Tools > References... menu item and press the Browse... button.
3. Locate the tlb (Type Library File) you created previously to make the IOM API available to the macro.

Building the Spreadsheet



- In the following example a connection will be made to the Innovator Server to retrieve/edit Design Requests:

	A	B
1	Item Number	Description
2	DR-000010	High density plastic case resists cracking
3	DR-000020	Detachable Keyboard design for portable table

This example will demonstrate retrieving and editing Design request Items from MS Excel.

Creating new IOMFactory and Innovator Object



```
Dim f As New IomFactory 'must use the New keyword  
Dim innov As Innovator  
Set innov = f.CreateInnovator(Nothing)  
  
'Establish User Credentials  
URL = "http://localhost/Innovator930"  
db = "InnovatorSolutions"  
user = "admin" : pw = "innovator"
```

As always, you establish an authenticated connection to the Innovator server. As we discussed in a previous unit, you can use the IomFactory class to establish this connection.

Here the credentials are hard coded into VBA variables but they could be obtained easily from a VBA Form as well.

Establishing a Connection to IOM



```
Dim conn As HttpServerConnection  
  
Set conn = f.CreateHttpServerConnection(URL, db,  
                                         user, pw) ' create a connection  
  
Dim result As Item  
  
Set result = conn.Login  
  
If result.IsError Then  
  
    MsgBox result.getErrorDetail  
  
    Exit Sub  
  
End If
```

Copyright © 2014 Aras

All Rights Reserved.

aras.com

As we learned previously, the `HttpServerConnection` class allows you to log on to Innovator server using the appropriate credentials.

Note

It is a good practice to use the `Logout` method once the Excel session has ended to terminate the connection to the server.



Retrieving Data

```
Set innov = f.CreateInnovator(conn)
Dim qry1 As Item
Set qry1 = innov.newItem("Help Ticket", "get")
qry1.setAttribute "order_by", "item_number"
Dim res1 As Item
Set res1 = qry1.Apply

For i = 1 To res1.getItemCount
    Cells(i+1,1) =
    res1.getItemByIndex(i-1).getProperty("item_number")
    Cells(i+1,2) =
    res1.getItemByIndex(i-1).getProperty("details")
Next i
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

The CreateInnovator method is then used to establish an Innovator object in the program which then allows us to work with any of the IOM methods we have discussed in this course.

In this example, all of the Design requests will be retrieved from the Innovator database, sorted by item_number.

A for loop is then used to work with each Item individually and assign appropriate data to the MS Excel cells (using the VBA Cells method).

Saving Data



```
'Connection has been established to server...

active_row = ActiveCell.Row
item_number = Cells(active_row, 1)

Dim qry1 As Item
Set qry1 = innov.newItem("Design Request", "edit")
qry1.setAttribute "where",
    "design_request._item_number=''' + item_number + '''"
qry1.setProperty "_description", Cells(active_row, 2)
Dim res1 As Item
Set res1 = qry1.Apply
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

This example gets the active row of the MS Excel spreadsheet and edits the item number based on the contents of column 2 in the active row.



Summary

In this unit you learned how to use an alternate COM client to communicate with the Innovator server.

You should now be able to:

- ✓ Use a COM Client with the IOM
- ✓ Create and Register a Type Library using REGASM.exe
- ✓ Configure Microsoft Excel to use a Type Library
- ✓ Create an Excel Macro (VBA) that connects to the IOM
- ✓ Retrieve Data into Microsoft Excel
- ✓ Save Spreadsheet Data to Aras Innovator



Lab Exercise

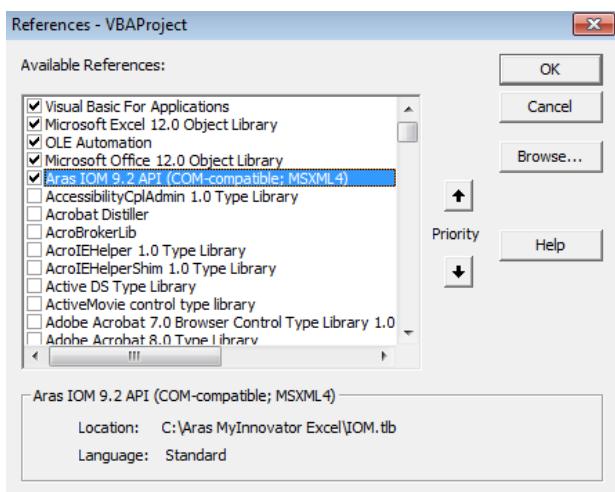
Goal:

Use the Common Object Model (COM) interface to Aras Innovator with an external client.

Scenario:

In this exercise you will use Microsoft Excel to display the current Sales Orders in the Aras Innovator database.

1. Register the client IOM.dll as a Microsoft Type Library (.tlb) using the steps outlined in this unit.
2. Create a new Microsoft Excel workbook that supports macros (.xslm)
3. Create a new Macro named **SalesOrders**.
4. Add the IOM.tlb type library to the workbook from the Tools > References menu.



5. Establish a connection to the Aras Innovator server by:
 - Defining Credentials
 - Creating an HttpServerConnection
 - Logging into the Server
 - Creating an Innovator instance variable
6. Use IOM API methods to retrieve all of the Sales Order items into an Item variable.

7. Loop through the list and show the so_number, po_number and shipping_method as rows in the Excel spreadsheet.

	A	B	C	D
1	SO-0001	9088	Federal Express	
2	SO-0002	23451	UPS	
3	SO-0003	12231155	Federal Express	
4	SO-0005	123456	Federal Express	
5	SO-0007		Federal Express	
6	SO-0008	989111	UPS	
7	SO-0011	12345	Federal Express	
8	SO-0013	111111	Federal Express	
9	SO-0014		Federal Express	



Unit 15 Importing Data with Batch Loader

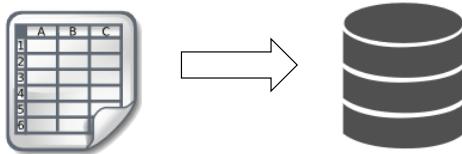
Overview: In this unit you will learn how to use the Batch Loader utility to migrate data from an outside resource into the Innovator database. The Batch Loader is a subscriber only feature available on the Aras Innovator CD.

- Objectives:**
- ✓ Reviewing the Batch Loader Features
 - ✓ Working with AML Templates
 - ✓ Using the BatchLoader Tool
 - ✓ Importing Data from a File
 - ✓ Using Command Line BatchLoader
 - ✓ Creating AML Templates

Reviewing Batch Loader Features



- Separate Executable designed to import data from external flat files
- Transfers data into AML statements to load new Items
- Two modes available
 - Windows Application
 - Command Line



Copyright © 2014 Aras

All Rights Reserved.

aras.com

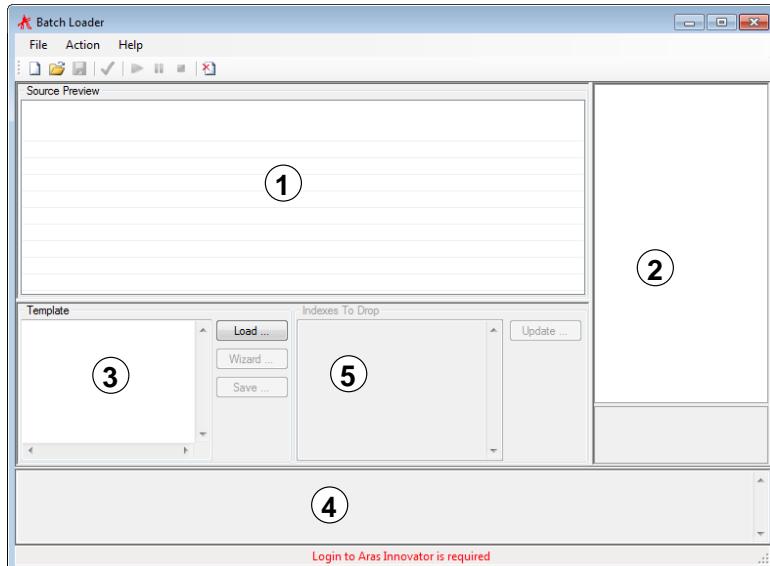
The Aras Batch Loader tool is a utility for loading data from a flat file into Aras Innovator. This tool transforms the flat file data into AML for loading directly into Aras Innovator much like an alternative client.

The Batch Loader has two distinct modes for loading data. The first, more direct method is through execution of the Batch Loader utility at the command line. This allows for rapid data loading into an Aras Innovator instance based upon a configuration pre-defined in an xml file. The use of the command line executable also makes it suitable for use in a daily batch job through the use of Windows Scheduler.

Note

The Batch Loader is a subscriber only tool. You will need to obtain a license key from Aras in order to operate the tool.

Using the Batch Loader Application



Copyright © 2014 Aras All Rights Reserved.

aras.com

The Batch Loader tool is divided into 5 areas:

- ① **Preview Pane**
Shows a preview of the flat file data that will be loaded into the Innovator database.
- ② **Configuration Pane**
You define the file to be processed as well as appropriate parameters in this pane.
- ③ **Location**
The AML Template is displayed in this pane. AML templates are used to build AML instructions for the server to process the flat file data.
- ④ **Status**
Status messages are presented in the status pane as the tool processes a file.
- ⑤ **Indexes to Drop**
To improve performance on very large imports, you can disable database indexes using an alter index statement in SQL Server. Contact support@aras.com if you need more information/recommendations regarding this option.

Completing the Configuration Page



Database Options	
Server	http://localhost/Innovator920
Database	InnovatorSolutions920SP1
User	admin
Load Options	
DataFilePath	C:\Users\bellis\Desktop\CustomerData.csv
Delimiter	.
WorkerProcesses	1
Threads	1
LinesPerProcess	100
Encoding	Unicode (UTF-8)
FirstRow	1
Last Row	-1
Log Options	
LogFile Path	c:\dataload.log
LogLevel	2
Preview Options	
PreviewRows	10
MaxColumns	10
FirstRowIsColumnName	False

Copyright © 2014 Aras All Rights Reserved.

aras.com

You configure what file will be processed as well as other parameters to the tool in the Configuration Pane. The following fields are available:

Server – The connection URL for Aras Innovator. If all the defaults were taken during the Innovator installation, the path should be something like: `http://localhost/InnovatorServer`. If unsure, please check in IIS to get the exact path. This URL should not include reference to the `/Client` folder

Database – The database to which the data will be loaded. Selecting this field after defining the connection URL will make this a pick list of available databases.

User – The user login to be used for connecting to and loading data into the database.

DataFilePath – The fully qualified path to data file that contains the data to be loaded into the database. In our case this is `C:\Parts.txt`

Delimiter – The delimiter used to separate data, usually a tab (`\t`), or a space (), or a comma (`,`).

WorkerProcesses – The number of worker processes to be used by the Batch Loader while loading data. Recommend using the default of 1.

Threads – The number of threads per worker process. Recommend using the default of 1.

LinesPerProcess – The number of lines in the data file that will be loaded by a single worker process. If the worker process finishes processing all its lines and the data file has more lines to be processed, then a new worker process will be started.

Encoding – Encoding (or codepage number) of data file

FirstRow – The number of the row where the actual data starts. Sometimes the first row is used for row headings. In that case, the data will start in the second row. (See FirstRowIsColumnNames property below.)

LastRow – The number of the row where the actual data stops. Default of -1 indicates that the file should be read until the end of the file

LogFile Path – The fully specify the name of the log file where all information and errors are to be written by the Batch Loader.

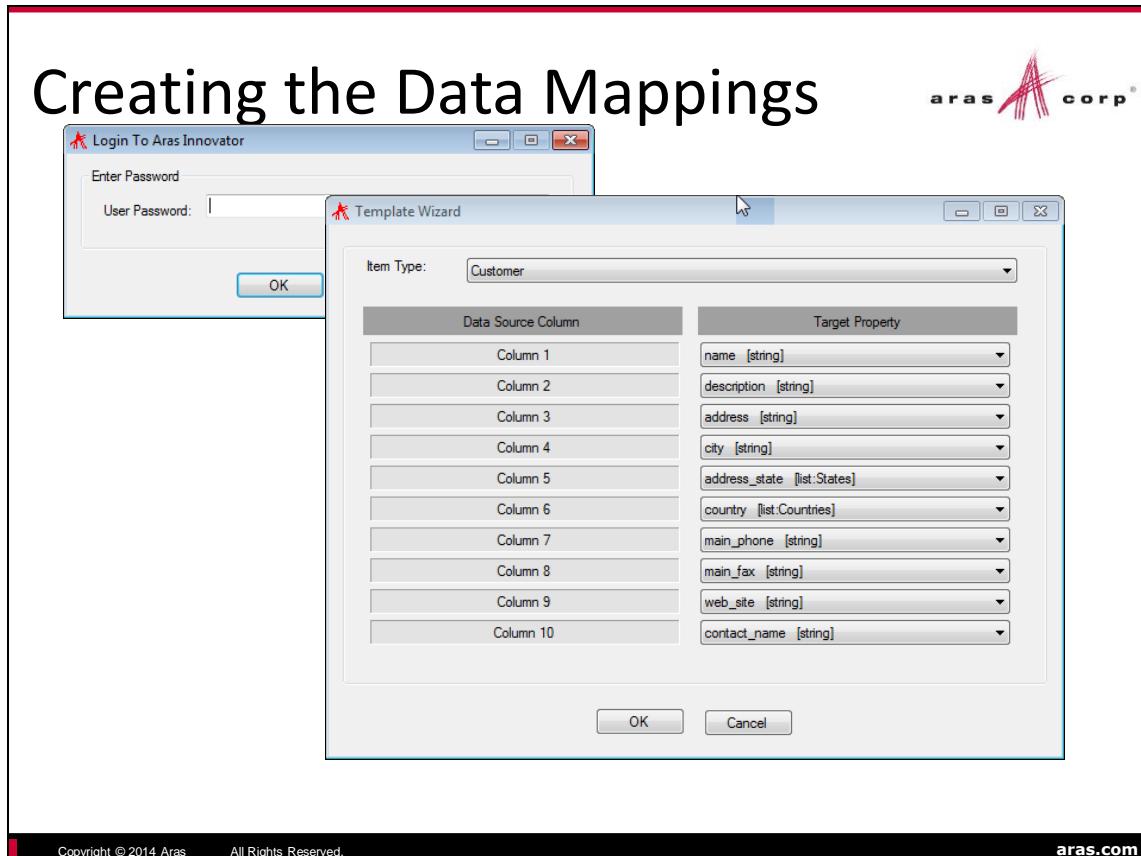
LogLevel – The Level of detail included in the logging.

- 1 – Low, recommended for automated jobs with low risk on failure. Details about start and stop, and how many items succeeded.
- 2 – Medium, recommended for use while developing new Batch load job. Logs details about failure, as well as details logged in low mode.
- 3 – High, recommended for debugging. Provides detail and AML about every line loaded.

PreviewRows – The number of rows visible in the Preview Pane

MaxColumns – The number of columns in the Preview Pane.

FirstRowIsColumnNames – When set to True, the Batch Loader starts parsing the data file from the second row after the FirstRow value.



Using the Template Wizard

Press the *Wizard* button in the AML Template Pane to start the Data Mapping Wizard. You will be required to enter the password for the User specified in the Configuration Pane before you can begin.

1. Select the ItemType to indicate the type of Item to be created from the imported data.
2. Use the Target Property dropdown list to choose a target property for each Column in the delimited flat file. You can view the contents of the file in the Preview Pane to help you to map the data.
3. Press OK when mapping is complete.

Reviewing the Batch Loader Template



Template

```
<Item type='Customer' action='add'>
<name>@1</name>
<description>@2</description>
<address>@3</address>
<city>@4</city>
<address_state>@5</address_state>
<country>@6</country>
<main_phone>@7</main_phone>
<main_fax>@8</main_fax>
<web_site>@9</web_site>
```

Copyright © 2014 Aras

All Rights Reserved.

aras.com

The AML Template is used to indicate to the server on how data should be added (or merged) into the Innovator database. The template uses standard AML statements you have learned about in this course.

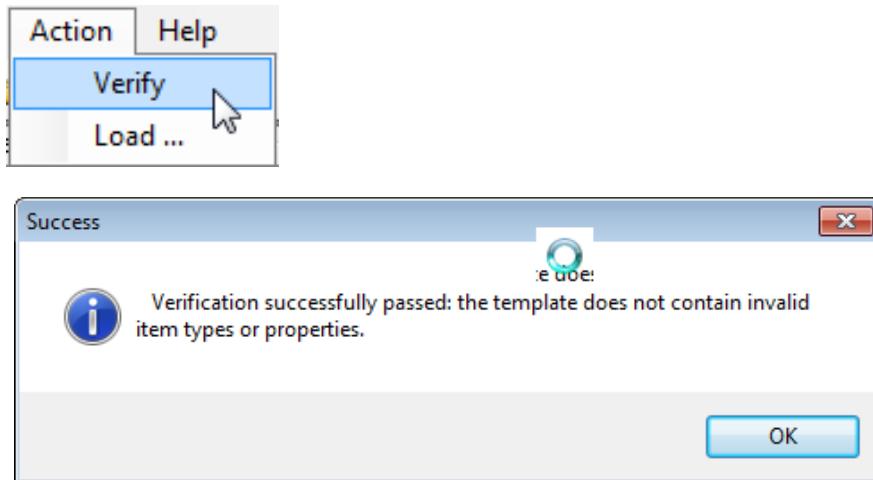
Each data column from the flat file is marked with a “@” sign followed by a number. In the example above, the first column of the file is expected to have the “name” field.

You can modify this template file and also save it by pressing the Save key in the AML Template Pane.

Verifying the Template



- Use Verify Action to check template for problems

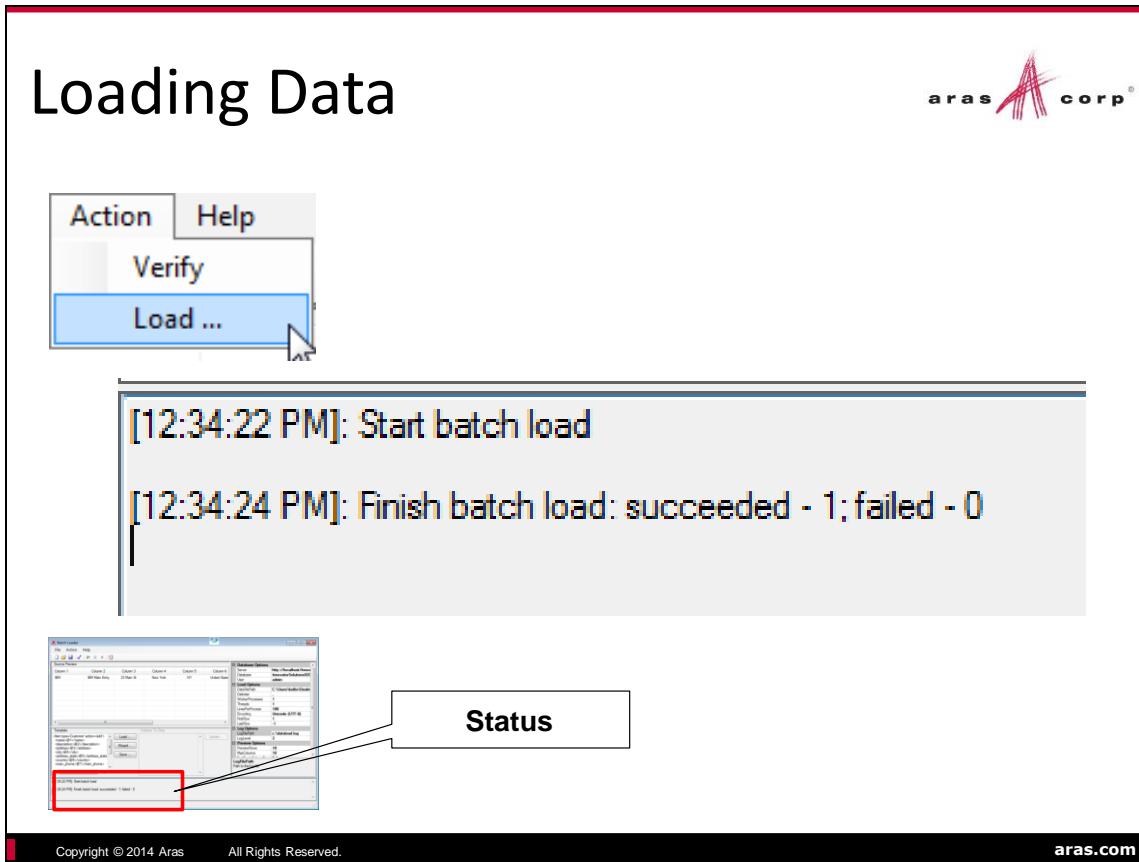


Copyright © 2014 Aras

All Rights Reserved.

aras.com

To assist with defining a correct AML Template, the Verify action is available to check that the AML syntax presented is correct. Any error messages will be displayed in a message box. If no errors are found the success message displayed above appears.



Use the Load... action to begin the import process. The Status Pane will display the data as it is imported and indicate success or failure.

Remember that a data log (configured in the Configuration Pane) is also available for review when the import is completed.



Importing Relationships

- Define an external file that relates two Items together:

	A	B
1	Document Name	Help Ticket
2	AssemblyJ100	TKT-0005-2010

Copyright © 2014 Aras All Rights Reserved.

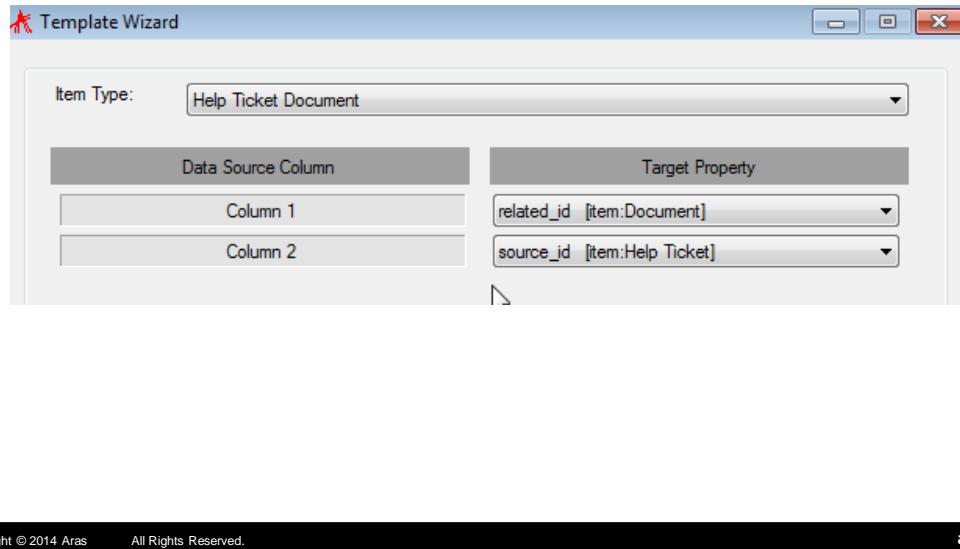
aras.com

Along with single items, you can also import Relationship data.

One simple way to define relationships is to create a “join” spreadsheet that indicates the source and related Item based on some unique property.

In this example, a spreadsheet has been created that denotes a Document named AssemblyJ100 that should be related to a Help Ticket TKT-0005-2010.

Defining Relationship Mapping



Using the Template Wizard described previously, choose a Relationship ItemType and then match the columns of the spreadsheet with the related_id and source_id properties of the relationship.

In this example, the Document is the related Item and the Help Ticket is the source.

Reviewing Relationship Template



Source Preview

Column 1	Column 2
AssemblyJ100	TKT-0005-2010

Template

```
<item type='Help Ticket Document' action='add'>
  <related_id>
    <item type='Document' action='get' select='id'>
      <item_number>@1</item_number>
    </item>
  </related_id>
  <source_id>
    <item type='Help Ticket' action='get' select='id'>
      <item_number>@2</item_number>
    </item>
  </source_id>
</item>
```

Copyright © 2014 Aras

All Rights Reserved.

aras.com

Review the template to make sure that the AML syntax is correct (also check with the Verify action).

In this example, the item_number property is used to find the Document and Help Ticket in the database and relate them together using the Help Ticket Document relationship.

Using Command Line Mode



- Use the BatchLoaderCMD.exe executable
- Allows for scheduled imports using previously saved configurations
- Server connection configured using an XML file
- Uses Configuration Template Files to process import data

The Batch Loader can also be execute from the windows command line. The command line version of the Batch Loader allows for the use of a known configuration that can be used in a regularly scheduled process (e.g. adding new Users to the database).. Combined with the Scheduler Service, the command line Batch Loader can also import data at scheduled time intervals.



BatchLoaderCMD Arguments

- BatchLoaderCMD –d {datafilename} [arguments]
- Most arguments can be contained in an XML configuration file (called with the –c switch)
- File delimiter character must be defined in the XML config file
- Example:

```
BatchLoaderCMD -d c:\import.csv -c c:\config.xml  
                  -t c:\template.xml
```

The BatchLoaderCMD executable has no user interface but accepts a number of switch arguments which can provide connection and configuration information.

Usage: BatchLoaderCmd.exe -d {data file} [optional arguments]

where:

-d {data file} - path to a delimiter-separated data file

(Optional arguments can either be specified in the command line or in a configuration file.)

-c {config file} - path to configuration file; default is 'config.xml' if no file is specified

-t {template file} - path to template file; default 'template.xml'

-l {log file} - path to log file

-ll {log level} - verbosity of logging (1 - 3); default 1

-e {encoding} - name of data file encoding; e.g. 'us-ascii'

-s {url} - Innovator server URL

-db {db name} - Database name

-u {user name} - Innovator user login name

-p {user password} - Innovator user password

-th {threads} - Number of threads in a worker process; default 1

-pr {processes} - Number of worker processes; default 1

-lpp {lines} - Lines processed by each worker process; default 1000

-fr {row number} - First row to process in the data file; default 1

-lr {row number} - Last row to process in the data file; default -1

Note

Most of these parameters are optional and are generally included in the xml configuration file. However, the command line arguments will override any values in the configuration file.

Sample Configuration File

```
<BatchLoader>
<server>http://localhost/InnovatorServer</server>
<db>InnovatorSolutions</db>
<user>admin</user>
<password>innovator</password>
<max_processes>1</max_processes>
<threads>1</threads>
<lines_per_process>250</lines_per_process>
<delimiter>\t</delimiter>
<encoding>utf-8</encoding>
<first_row>2</first_row>
<last_row>-1</last_row>
<log_file>C:\User</log_file>
<log_level>3</log_level>
</BatchLoader>
```

Example Template File



- Add new Users with password "innovator":

```
<BatchLoaderPrototype>
<Item type="User" where="login_name='@1'" action="add">
<login_name>@1</login_name>
<password>607920b64fe136f9ab2389e371852af2</password>
<logon_enabled>@2</logon_enabled>
<first_name>@3</first_name>
<last_name>@4</last_name>
<email>@5</email>
</Item>
</BatchLoaderPrototype>
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

As discussed previously in this unit, AML Templates define the instructions for loading data from an external file.

This example shows how a series of Users could be added from a file with 5 columns of data defined.

Note

Additional examples are available in the *Aras Innovator Batch Loader 9.2* documentation.



Summary

In this unit you learned how to use the Batch Loader utility to add new data to an Innovator database from an outside file.

You should now be able to:

- ✓ Create an AML Template File
- ✓ Import Items and Relationships from an External File
- ✓ Use the Graphical Tool for Import
- ✓ Use Command Line Batch Loading



Lab Exercise

Goal:

Be able use the Batch Loader to import data from an external file into Aras Innovator.

Scenario:

In this exercise, you will use the Batch Loader to import new Users into the system using the spreadsheet you created earlier in this course to create a alternate client to Aras Innovator using Microsoft Excel.

Time:

60 minutes

Steps:

1. Locate the spreadsheet you created earlier in this course to add Users to the system.

	A	B	C	D	E	F	G	H	I
1	Logon Name	Password	First Name	Last Name	Enabled?	Message			
2	Test1	test1	TestFirst	TestLast		1 User with Logon Name 'Test1' already exists.			
3	Test3	test2	Test3	User		1 OK			

2. Change the logon name and First/Last name data to Users that do not currently exist in the system.
 3. Save the spreadsheet as a comma delimited text file.
 4. Start the Batch Loader utility and complete the Configuration Pane to import the records from the comma delimited file you saved above. You can use the MD5 encrypted password for "innovator" shown in this unit as the password column by modifying the AML template.
 5. Import the records and test to make sure they exist in the system.
 6. Save the template file you created/modified in the Batch Loader tool and attempt to import a different set of Users using command line mode. How could you configure the system so that the command line Batch Loader was executed once a day to check for new Users?
-



Unit 16 Configuring the Scheduler Service

Overview: In this unit you will learn how to set up a windows service that will execute a specified Innovator Method periodically based on configured settings. This is useful for sending reminders or processing escalations.

- Objectives:**
- ✓ Reviewing the Scheduler Service
 - ✓ Installing and Configuring the Service
 - ✓ Monitoring the Service

Reviewing the Scheduler Service



- Executable Program Installed as a Windows Service
- Connects to Aras Innovator Server
- Runs one or more Methods at Scheduled Intervals
- Useful for:
 - Sending reminders
 - Processing timed escalations
 - Running nightly reports
 - Performing scheduled file replications



Copyright © 2014 Aras All Rights Reserved.

aras.com

The Aras Innovator Service is a Windows executable program that runs as a Windows Service. It is used to automatically run Innovator Methods on a periodic schedule, as specified in the InnovatorServiceConfig.xml file.

A single running instance of the Aras Innovator Service can be configured to periodically connect to any number of Aras Innovator Servers over the network, and for each Aras Innovator Server, run any number of Methods. The standard Windows Event Viewer is used for logging of program status and error messages.

Installing the Scheduler Service



■ Components:

- Innovator Service Executable
- InstallService Batch File
- UninstallService Batch File
- InnovatorServiceConfig.xml

To Install the Scheduler Service

1. Copy the InnovatorService.exe file and the two BAT files into any Folder.
2. Copy the file InnovatorServiceConfig.xml into the same folder.
3. Run the BAT file InstallService.bat. This program runs the standard .NET utility named INSTALLUTIL. This program will leave a set of .NET log files in the working directory which should be checked for error messages, and then deleted.
4. You will be prompted for a user name and password. This is the user that will have privileges to run the Windows Service. Use a fully qualified user name (domain_name\user_name). If the user name and password are not valid the installation will fail.
5. Access the Services Management Console to make sure the service named “Innovator Service” has been added to system. You can then configure when to start the service, set to Automatic, etc. You can also edit the user name and password that will execute this service or use the Local System Account.



Configuring the Service – InnovatorServiceConfig.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<innovators>
  <innovator>
    <server>SERVER </server>
    <database>DATABASE</database>
    <username>USERNAME</username>
    <password>PASSWORD</password>
    <http_timeout_seconds>600</http_timeout_seconds>
    <job>
      <method>METHOD NAME</method>
      <months>MONTH STRING</months>
      <days>DAY STRING</days>
      <hours>HOUR STRING</hours>
      <minutes>MINUTE STRING</minutes>
    </job>
  </innovator>
  <eventLoggingLevel>LOGGING LEVEL</eventLoggingLevel>
  <intervalMinutes>INTERVAL</intervalMinutes>
</innovators>
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

The following parameters are available in the InnovatorServiceConfig.xml file:

<innovator> tag is used to specify an Aras Innovator server to connect to. There can be multiple **<innovator>** tags in the configuration file.

<server> is the HTTP url of the Innovator server.

<database> is the ID of the database in the InnovatorServerConfig.xml.

<username> is the login name that should be used for running the Methods.

<password> is the password for the login account, in plain-text.

<method> is the Method name in the Innovator instance.

<months> is the month of the year that the Method should be run. Supports a comma delimited list of 1 through 12. January is month=1 and December is month=12.

<days> is the day of the week that the Method should be run. Supports a comma delimited list. Sunday is day=0 and Saturday is day=6. A special value, “last”, is used to indicate the last day of the current month.

<hours> is the hours of the day that the Method should be run. Supports a comma delimited list of hour values from 0 through 23.

<**minutes**> is the minute of the hour that the Method should be run. Supports a comma delimited list of hour values from 0 through 59. A special value, “once”, is used to indicate the job should be run only “once” when the <hours> criteria is met, but on which minute is not important.

<**eventLoggingLevel**> sets the level of detail for messages that are sent to the Windows Event monitor.

- 0 = Service startup announcement and error messages from Innovator only
- 1 = Announces the start of every Method run
- 2 = Detailed reporting of every run cycle.

<**intervalMinutes**> sets the Windows timer to control how often the InnovatorService is polled and the <job> timing logic is tested. Any value from 1 to 60 is supported.

Monitoring the Service – Event Viewer



A screenshot of the Windows Event Viewer window. The left pane shows categories: Application, Security, and System. The right pane is titled 'Event Viewer (Local)' and shows 'Application 3 event(s)'. A table lists three events: 1) Information event on 1/13/2005 at 6:25:43 PM from source InnovatorService with category None, event ID 0, user N/A, and computer REDWALL. 2) Information event on 1/13/2005 at 6:25:43 PM from source InnovatorService with category None, event ID 0, user N/A, and computer REDWALL. 3) Information event on 1/13/2005 at 6:25:43 PM from source InnovatorService with category None, event ID 0, user N/A, and computer REDWALL. An 'Event Properties' dialog box is open over the main window, centered on the third event. The dialog shows the event details: Date: 1/13/2005, Source: InnovatorService, Time: 6:25:43 PM, Category: None, Type: Information, Event ID: 0, User: N/A, Computer: REDWALL. The 'Description' section contains the text: 'Innovator Service Startup', 'Time Interval: 2 minutes Logging Level: 2', 'Server: http://Redwall/DelhiQuality', 'username:admin', 'Method: Schedule Test Run at: 0,1,2,3,4,5,6, days', '0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23, hrs', '0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,5', '0,52,54,56,58, minutes'. It also includes a link to 'http://go.microsoft.com/fwlink/?events.asp'. At the bottom of the dialog are buttons for 'OK', 'Cancel', and 'Apply'.

Copyright © 2014 Aras All Rights Reserved.

aras.com

Use the Windows Event Viewer to track when the Service is executed.

The level of logging information displayed is configured in the InnovatorServiceConfig.xml file.

Example



```

<innovators>
  <innovator>
    <server>http://localhost/Innovator10</server>
    <database>DevelopingSolutions</database>
    <username>admin</username>
    <password>innovator</password>
    <http_timeout_seconds>600</http_timeout_seconds>
    <job>
      <method>Scheduled Email Delivery</method>
      <months>*</months>
      <days>*</days>
      <hours>*</hours>
      <minutes>0,15,30,45 </minutes>
    </job>
  </innovator>
  <eventLoggingLevel>2</eventLoggingLevel>
  <intervalMinutes>1</intervalMinutes>
</innovators>

```

Copyright © 2014 Aras All Rights Reserved.

aras.com

In this example, the Method *Scheduled Email Delivery* is executed every 15 minutes. Source code for this Method appears on the next page. The Service Interval check is set for 1 minute and detailed reporting (service level 2) is defined.

Note

Two standard server Methods are also provided in the Solutions database to check for workflow reminders as well as escalations. The example below shows two sample jobs that can be included in the configuration file to run these recommended Methods on a periodic basis:

```

<job>
  <method>Send Email Reminders</method>
  <months>*</months>
  <days>*</days>
  <hours>0</hours>
  <minutes>2</minutes>
</job>
<job>
  <method>Check Escalations</method>
  <months>*</months>
  <days>*</days>
  <hours>0</hours>
  <minutes>12</minutes>
</job>

```

Sending an Email Reminder



```
01 //Get email message
02 Item email_msg = this newItem("EMail Message", "get");
03 email_msg.setProperty("name", "Service Reminder");
04 email_msg = email_msg.apply();
05 //Get admin Identity
06 Item iden = this newItem("Identity", "get");
07 iden.setProperty("name", "Innovator Admin");
08 iden = iden.apply();
09
10 bool result = this.email(email_msg, iden);
11
12 if (!result) {
13     return this.getInnovator().newError("Mail not sent");
14 }
15 return this;
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

The Scheduled Email Delivery Method uses the `email` method of the `Item` class to send an e-mail message to a user (in this example it's the admin).

Note

The E-mail message must have a From Identity or the `email` method will fail.



Summary

In this unit you learned how to install and configure the Scheduler Service.

You should now be able to:

- ✓ Install the Scheduler Service
- ✓ Configure the Scheduler Service to execute Methods periodically
- ✓ Monitor the Scheduler Service



Lab Exercise

Goal:

Be able to use install and use the Scheduler Service to execute a Method based on a time schedule.

Scenario:

In this exercise, you will create a simple server Method that logs a message to the debug log each time the Method is executed. You will then install and configure the Scheduler Service to call this Method and check the log to make sure the service is running.

Time:

30 minutes

Steps:

1. Create a server Method named Schedule Test that creates a message in the debug log. Use the WriteDebug method (CCO class) you learned about earlier in this course to write a message to the log each time the Method is executed. Test to make sure the message is written to the log.
2. Obtain the Scheduler Service from the instructor and copy the files to a directory named "Scheduler" on the C:\ drive.
3. Configure the Scheduler Service to connect to your installation.
4. Set the time interval to run the Method once every 2 minutes.
5. Start the Innovator Service and check the log file for results.



Appendix A: Custom Search Result Sets

Overview: There may be occasions when it is necessary to restrict the search criteria on the Item search grid. The grid appears when a user presses the ellipsis button or F2 on a form field representing an Item property - or when a user searches for a related Item when building a new relationship.

The OnSearchDialog event is invoked in these situations and is available to allow you to predefine the search criteria for an Item search, or to build a custom search result set based on your own business logic.

Objectives: ✓ Defining Custom Search Result Sets



Defining Custom Result Sets

- Define custom business logic to define a result list for a user
- Client side method must create a comma delimited list of item id's
- ID list is set on the Query object
- User cannot access the search criteria bar

A screenshot of a software interface titled "Search dialog - Product". At the top, there is a toolbar with icons for search, filter, and other functions, along with a dropdown menu for "Hide Search Criteria" and a button for "25" results. Below the toolbar is a table with three columns: "Product Number", "Name", and "Description". A single row is visible, showing "V111" in the Product Number column, "SmartPhone 111" in the Name column, and "Android based smart phone" in the Description column. The bottom of the screen shows a footer with copyright information: "Copyright © 2014 Aras All Rights Reserved." and the website "aras.com".

Defining Custom Result Sets

To control the results in the Item search grid you can create a Method that populates the grid with Items based on your own business logic. The search criteria bar will be hidden from the user and they will not be able to change the criteria.

This option is useful for presenting a restricted set of Items based on the value of other Items or conditions. In this unit, we will create a Method that prevents a user from selecting a *Product* unless a Model (relationship) has been established for the Product Item.

The client side method works directly with the Query object to indicate what results should be displayed to the user. The method must create a comma delimited list of valid Item id's that are passed to the Query object when a user performs an Item search.

Using Method Parameters



■ Parameters passed to calling Method:

- **inDom**
 - Contains the AML request for the search grid
- **inArgs.windowContext**
 - Window that invoked the search
- **inArgs.itemTypeName**
 - ItemType name of items to be searched
- **inArgs.itemContext**
 - The Item from which the search is being invoked
- **inArgs.itemSelectedID**
 - If search is invoked from Relationship Grid – the id of the row

Using Method Parameters

Two objects are passed to the custom search Method which allows you to determine the current object the user is working on as well as other runtime criteria as shown above.

Using Method Parameters



- Parameter that must be set in the Method to define the result set
 - `inArgs.QryItem`
 - Contains "idlist" attribute that is set to comma delimited list of Item id's

Using Method Parameters

The Method must set the *idlist* attribute of the *QryItem* object with a valid comma-delimited list of Item id's. The Items will be then presented to the user in the search grid for selection.

Creating the Custom Result Method



- Product Search Example
 - User can only choose from Products that have a Model
- Retrieve a collection of Products that have an assigned Model relationship
- Build a comma-delimited list of id's based on the result
- Set the inArgs.QryItem attribute to define the result set

Creating Custom Result Method

The following Method example named *ProductModelSearch* demonstrates how to set the parameter object *QryItem* with a valid idlist.

The goal of this example is to prevent a user from selecting a *Product* that does not have an assigned Model. Only *Products* that have an assigned Model relationship will be displayed in the search grid. You can easily customize this query to return a different result for different use cases.

Programmatically supplying items to the search grid prevents the user from overriding the search criteria.

Example source code:

Developing Solutions

```
//FIND PRODUCTS with MODELS ONLY
var prods = this newItem("Product", "get");
var relationship = this newItem("Model", "get");
relationship.setPropertyCondition("item_number", "is not null");
prods.addRelationship(relationship);
prods = prods.apply();

///GET THE COUNT OF PRODUCT ITEMS RETURNED
var count = prods.getItemCount();

//CREATE A COMMA DELIMITED STRING OF ID'S BY ADDING EACH ID
//TO A SIMPLE ARRAY
var idArray = new Array();
for (i=0; i < count; i++) {
    var item = prods.getItemAtIndex(i);
    idArray[i]=item.getID();
}

//USE THE JOIN FUNCTION TO CONVERT THE ARRAY TO A COMMA DELIMITED
//STRING
var idlist=idArray.join(",");

//***SET THE idlist ATTRIBUTE ON THE QRYITEM TO THE LIST
inArgs.QryItem.item.setAttribute("idlist", idlist);

//RETURN MODIFIED inArgs
return inArgs;
```

The screenshot shows two windows from the Aras Innovation application:

- Properties Window (Top):** Shows the context menu for the **related_id** property of a relationship item. The menu is open at point ①, showing options: New, Delete, Copy. The **related_id** column is highlighted in yellow.
- Event Window (Bottom):** Shows the list of events. A new event named **ProductModelSearch** is listed, with its **Method Type** set to **JavaScript**. The event is highlighted in yellow at point ②.

Both windows have standard toolbars and status bars at the bottom.

Associating Custom Method to OnSearchDialog Event

In this example, we want the custom search Method described on the previous page to be invoked when a user tries to establish a new relationship to a *Product* from a *Change Request*.

Remember that all relationship Items are configured with two standard properties – *source_id* and *related_id*.

In this example, we want to invoke the custom search Method when the user attempts to add or change the *related_id* of the *Change Request Product* relationship.

To Associate the Custom Method to the OnSearchDialog Event

For this example, open the *Change Request Product* relationship ItemType and locate the *related_id* property.

1. Display the Property form by selecting *View "Properties"* from the right mouse context menu.
2. Add the custom Method and select the *OnSearchDialog* event.



Summary

In this unit you learned how to customize Item searches using the *OnSearchDialog* Event.

You should now be able to:

- ✓ Create a Method that sets defines a result set in an Item search grid based on business logic



Appendix B: Creating Form Dialogs

Overview: You can use Form Items that are created in Aras Innovator as custom dialog boxes in a solution. This unit explains the steps to create and process a dialog box that uses a standard Aras Form.

- Objectives:**
- ✓ Creating a Form dialog
 - ✓ Populating the dialog
 - ✓ Returning information from the dialog



Creating Display Dialog Actions

- Use Forms created in Aras Innovator
- Forms can be displayed as an Action result
- Dialog can display (or edit) item data
- Dialog can be sized using form attributes
- Button and Field Events can return data to the server

You can create your own custom dialogs that take advantage of the Aras Innovator Form Item to display and edit data in a solution.

Dialogs can be used as a result of selecting an Action menu or triggered by a client Method to provide additional information or capture additional data.

Steps to Create an Action Dialog Box



- 1) Create a Form with named fields and optional button(s)
- 2) Create a Client Method to display the Form as a dialog box and process results
- 3) Create a Client Method to populate the fields of the dialog box
- 4) Create a Client Method to return data from the Dialog back to Innovator server
- 5) Create an Action that calls the Method created in Step 2

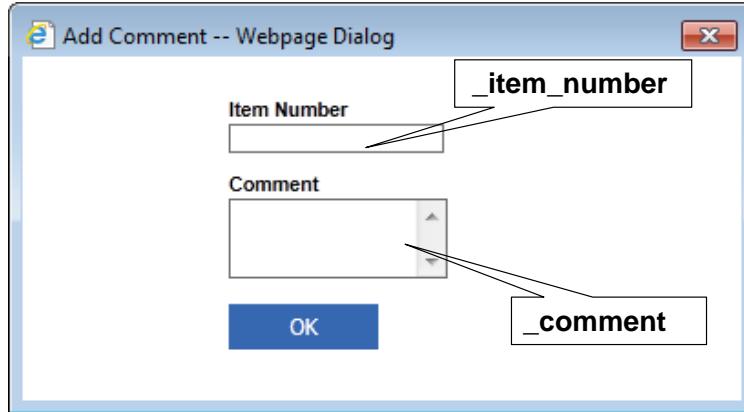
Steps to Create an Action Dialog Box

Action Dialog Boxes use standard Aras Innovator forms which you display programmatically. You can use them to display additional information about an Item or to provide custom editing capabilities.

The basic steps to create an Action Dialog Box include:

1. Creating a form that will be displayed as the dialog. You can add named fields, buttons and any other HTML controls that will be used in the dialog.
2. Creating a client Method that will show the dialog box based on an Action event and also process the results.
3. Creating a client Method on the form that will populate data in the HTML controls.
4. Creating a client Method on the form to return back results to the calling Method Step 2 above.
5. Creating an Action that calls the Method created in Step 2 above.

Step 1 – Create the Form



Copyright © 2014 Aras All Rights Reserved.

aras.com

Make sure to supply a name for each of the controls that will be displayed in the dialog. These names will be used to either display or capture data from the dialog.

Step 2 – Create Client Method to Display Form



■ Get a handle to the Form:

```
var q = this newItem("Form", "get");
q.setAttribute("select", "id,width,height");
q.setProperty("name", "Add Comment Form");
var r = q.apply();
```

■ Get Width and Height

```
var height = r.getProperty("height", "500");
var width = r.getProperty("width", "500");
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

Create a client Method that will be used to invoke the dialog.

You must first get a handle to the Form created in the first step. In this example the Form is named “Add Comment Form”.

Optionally, you can capture the Form size attributes and use them to size the dialog box to the same dimensions. In this example if the width and height are not supplied in the Form then 500 px will be used as the default dimensions.

Step 2 – Client Display Method



■ Set a Parameter Object

```
var param = new Object();  
param.title = "Add Comment";  
param.formId = r.getItemByIndex(0).getID(); // the id of  
the Form to be used  
param.item = this;  
param.aras = top.aras; // the aras object  
param.isEditMode=true;
```

Copyright © 2014 Aras

All Rights Reserved.

aras.com

The next part of the same client Method needs to populate a parameter Object that will be used to drive the dialog logic.

title

Defines what will appear in the windows title bar.

formID

The ID of the Form Item created in Step 1.

item

You can pass an Item to the dialog which can then be used as the context Item for other dialog Methods that either populate or capture data from the dialog.

aras

The aras client object is required to display the dialog

isEditMode

If set to true allows editing within the dialog. If false the dialog becomes display only.

Step 2 – Client Display Method



- Call showModalDialog function

```
var res = showModalDialog(  
    "ShowFormAsADialog.html",  
    param,  
    "dialogHeight:" + height + "px; dialogWidth:" + width +  
    "px; " +  
    "status:0; help:0; resizable:1; scroll:0;"  
);
```

Copyright © 2014 Aras

All Rights Reserved.

aras.com

The next part of the client Method calls the showModalDialog function which is included in the Aras Innovator client scripts.

The function requires the following parameters:

HTML form to use as dialog container

ShowFormAsDialog.html is provided in the Aras Innovator installation

Parameter Object

Discussed on the previous page

Attributes

A string containing attributes to set on the dialog window including size, scrolling, resizing, etc.

Step 2 – Client Display Method



■ Process results (optional)

```
if (res==undefined) {  
    return this;  
}  
  
//Add remark to current item  
  
var itm = this newItem("Design Request", "edit");  
itm.setID(this.getID());  
  
var rel = this newItem("Design Request Comments", add);  
rel.setProperty("_comment", res);  
itm.addRelationship(rel);  
  
itm = itm.apply();  
  
return itm;
```

Copyright © 2014 Aras All Rights Reserved.

aras.com

For dialogs that will return data for processing you can obtain the results of the dialog data which are returned in a string and use them for further processing.

In this example, a new Remark is added to the current Design Request by adding a new null relationship and setting the `_comment` property as the result of the returned data (`res`).

Step 3 – Create Client Method to Populate Form



- Method can use passed Item as Context:

```
var item_num =  
    document.thisItem.getProperty("_item_number");  
  
//Set value on form element  
document.getElementById("MainDataForm").name.value =  
    item_num
```

To pre-populate the dialog create a separate client Method and attach it to the onFormPopulated event of the Form to be used as the dialog.

If you pass an Item in the parameter object (shown on a previous page) this will become the context Item of the Method which allows you to retrieve values from the DOM to populate the HTML form fields.

In this example, the property “_item_number” is retrieved from the DOM and assigned to the “item_number” field on the Form.

Step 4 – Create a Client Method to Return Data



- Use SetReturnValue function:

```
var result =  
  
document.getElementById("MainDataForm").remark_text.value  
  
//return a value string  
top.setReturnValue(result);  
  
//close the window  
top.close();
```

Copyright © 2014 Aras All Rights Reserved.

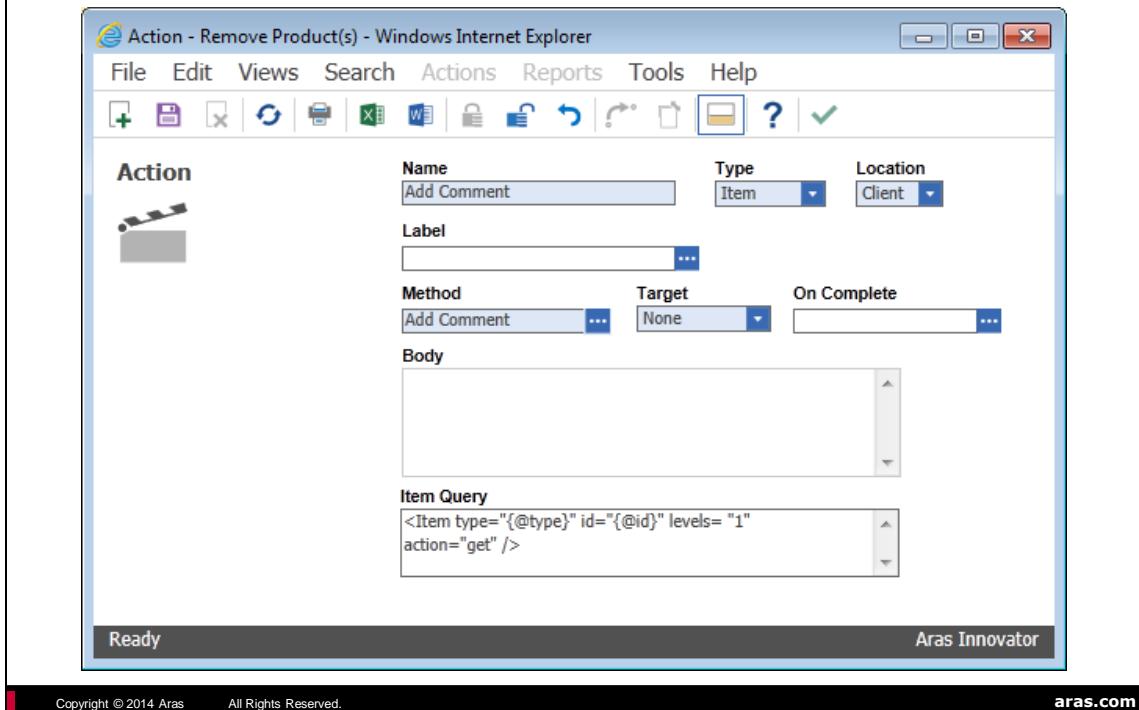
aras.com

You can return values back to the calling program by creating another client Method and associating it with a control event (typically a button onClick event).

In this example the value of the field “comment” is retrieved from the form and then returned using the setReturnValue function which is supplied another standard client function with the show dialog logic.

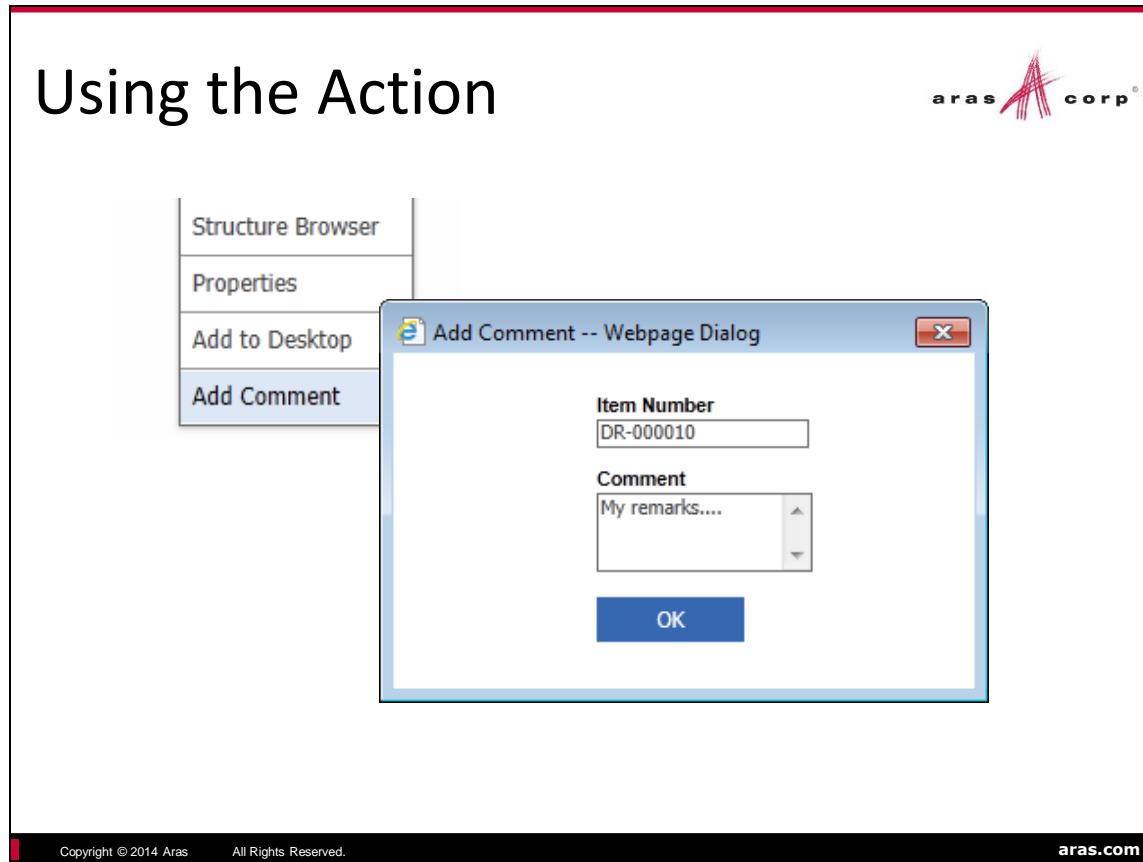
Finally, the top.close() JavaScript method closes the window and control is returned to the calling program.

Step 5 – Create the Action



To display the dialog from a client Action you will need to create an Action and supply the name of the first Method created to invoke the dialog.

In this example an Item Action is used so that the context Item can be passed to the displayed dialog box Form Methods.



Once complete, this Action can now be used as an alternate way to quickly add a Remark to a Help Ticket without opening/editing the Help Ticket Form.



Summary

In this unit you learned how to create a custom dialog using a Form Item.

You should now be able to:

- ✓ Create a dialog that uses a Form Item
- ✓ Populate and return data from the dialog

This page left blank intentionally.



Appendix C: Customizing a Form

Overview: When you create a Form in Aras Innovator, the form is rendered as an HTML page at runtime. You can modify the attributes of the form or perform custom operations on HTML controls using client Methods.

Objectives: ✓ Customize logic on form HTML controls

Using HTML Form Controls



- Form/field events and Client Methods define the values and behavior of the control
- Can be bound to a property using a Client Method

Creating Custom HTML Form Controls

You can define your own controls on an Item Form by selecting the HTML Control element in the Form Editor toolbar.

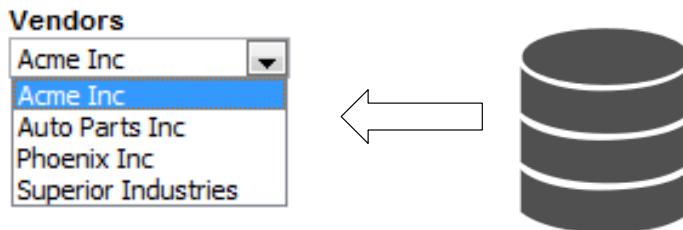
HTML Controls allow you to provide your own HTML source code to define the visual display and how the control will interact with the user.

Because forms use standard HTML with JavaScript you can define how the control will behave and appear to the user.

Building a Dynamic Drop Down List



- List is created dynamically from any data source
 - Create a Form Dropdown element
 - Create a Form Event Method to populate the list
 - Create a Field Event Method to capture result



Copyright © 2014 Aras All Rights Reserved.

aras.com

Building a Dynamic Drop Down List

This example shows how to create a dynamic drop down list that can be populated from any data source using a client Method. In this example we will be using a collection of Items from the Aras database. Later in the course you will learn how to use federated data to access outside resources.

To Build a Dynamic Drop Down List

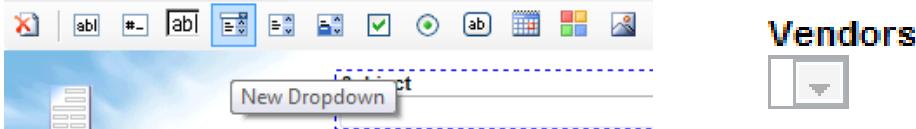
You must:

- Place the HTML Dropdown control on the Form and supply the appropriate actions to be performed.
- Create a Form Event Method to populate the control using JavaScript.
- Optionally, create a Form Event Method to capture the selected result into a property.

Building a Dynamic Drop Down List



- Create the HTML Dropdown Control



- Create a property to hold the selected value
- Populate Control
 - Create a Client Method for the onFormPopulated Event
- Capture Result
 - Create Client Method to set a property for the onChange Event

Copyright © 2014 Aras All Rights Reserved.

aras.com

Building a Dynamic Drop Down List

Defining the HTML Form Control

1. Create a string property to hold the value of the selected choice. In this example, the property is named "vendor".
2. Create an HTML Form element by selecting the Dropdown control from the Form control toolbar menu.
3. Name the control and place it on the form in an appropriate location. In this example, the control is named "VendorList".

Populating the Control

4. Create a Client Method to populate the control using Javascript.
5. Associate this Method with the onFormPopulated Form event. In this example, the drop down will be populated with Vendor items. The vendor name is displayed to the user and the selected vendor id is set in the vendor property.

```
//get Dropdown Control Element
var div_id = getFieldByName("VendorList").id;
var control_id = div_id.substring(0, div_id.indexOf("span"));
var select = document.getElementById(control_id);

//get Vendors
var returnItm = document.thisItem newItem("Vendor", "get");
returnItm.setAttribute("select", "name");
returnItm = returnItm.apply();

//populate the select list with options
for (var i=0;i < returnItm.getItemCount(); i++) {
    var vendor = returnItm.getItemByIndex(i);
    select.options[i]=
        new Option(vendor.getProperty("name"),
                   vendor.getProperty("id"));
}
//Set dropdown value to current property value
select.value = document.thisItem.getProperty("vendor");
```

Capturing the Selection

6. Create a Client Method to set the vendor property. Associate this Method with the onChange event of the Dropdown field.

```
document.thisItem.setProperty("vendor", this.value);
```

Changing Field Attributes



- Change Colors, Set Focus

Customer

**Name****Main Phone****Contact Name**

Copyright © 2014 Aras

All Rights Reserved.

aras.com

Changing Field Attributes

You can obtain a handle to any form control and set styles or attributes using standard JavaScript. This example shows how to set the focus on a specific field (Name) and change the field color.

Place the following code in a Form onLoad event:

```
//get the name text field
var div_id = getFieldByName("name").id;
var control_id = div_id.substring(0, div_id.indexOf("span"));
var name_field = document.getElementById(control_id);

//change the color of the part number text field
item_number_field.style.backgroundColor="#808080";

//Put focus on the name field (cursor) when form opens
name_field.select();
```



Summary

In this unit you learned how to create define custom logic for HTML form controls.

You should now be able to:

- ✓ Create custom logic for HTML form controls

This page intentionally left blank.



Appendix D: Creating a Configurable Grid

Overview: In this unit you will learn how to build a Configurable Grid which is used to replace the standard relationship grid in a tear off window. Configurable grids allow you to present related information in different ways to an end user.

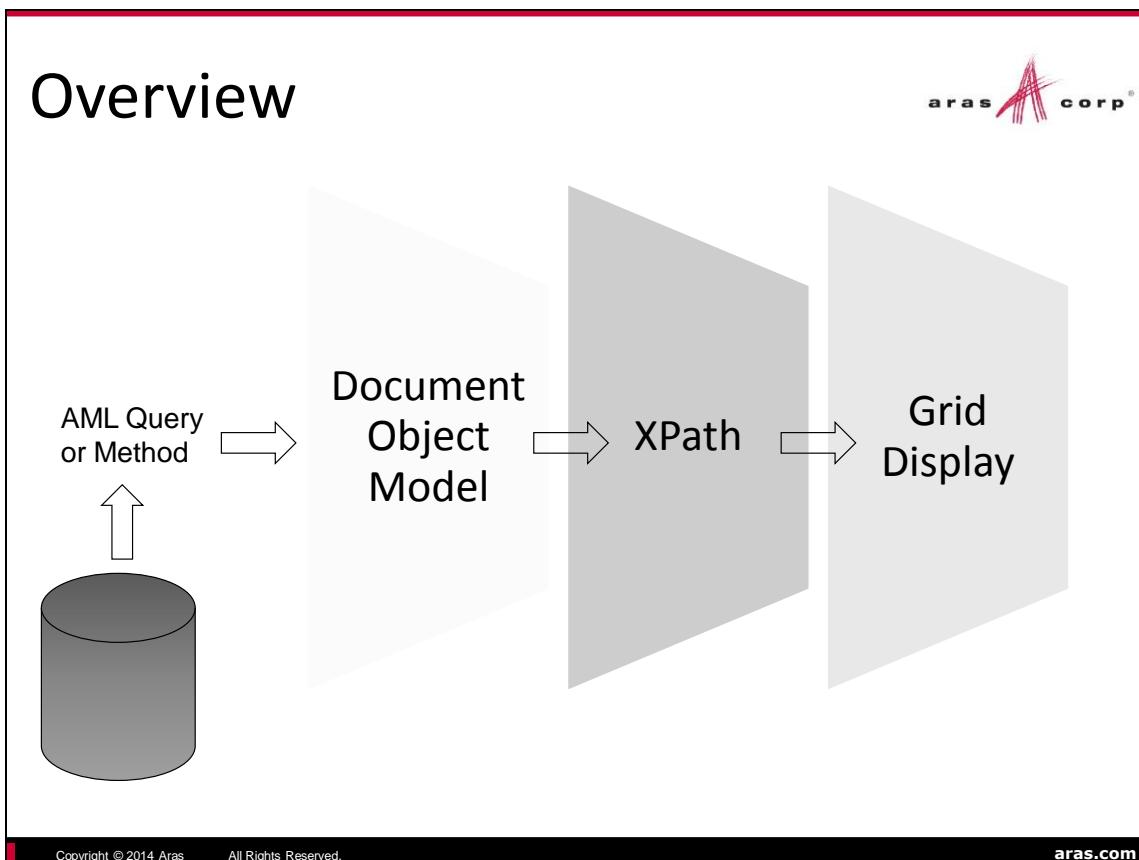
- Objectives:**
- ✓ Reviewing Purpose
 - ✓ Using a Configurable Grid
 - ✓ Defining the Grid Column and Events
 - ✓ Building the Select Query
 - ✓ Reviewing XPath Query Basics
 - ✓ Configuring Grid Columns
 - ✓ Attaching a Grid to a Relationship



Overview

- Allows creation of a custom relationship grid which is displayed on the Item form
- Can display multiple hierarchical Itemtype and Relationship data in a single display
- Grid Items manage display of complex data in a spreadsheet format – in place editing supported
- Data can be supplied using:
 - AML Query
 - Method
 - Federation

Configurable grids are ideal for presenting complex Item relationships or hierarchically related data in a “spreadsheet” format. You can populate the grid using an AML Query, a Method or through Federated Items and Properties (discussed later in this course).



A Configurable Grid is populated by creating a DOM using a standard AML Query or the return of a Result Item from a Method.

The Grid form allows you to define columns which are supplied data with standard XPath instructions. The output of the XPath queries result in a grid display of rows (for each Item specified in the DOM) and columns.

Using a Configurable Grid

Change Requests

Request Number	Title	Type	Product Item	Product Name
CR-000040	Cable assembly...	Defect	DesignJet 2000...	Large Format Pri...
CR-000010	Update to Windo...	Enhancement	DeskJet 400 Seri...	Inkjet Printer
			OfficeJet Busines...	All-in-one Office...

Customer - GHC (read only) - Windows Internet Explorer

Customer

Main Phone: Main Fax:

Contact Name: Dennis Koller

Address: 123 Main St.

City: Boston State: MA Country: Zip Code: 02101

Sales Orders Design Requests

Request Number	Title	Type	Product Item	Product Name
CR-000010	Update to Windo...	Enhancement		
CR-000040	Cable assembly...	Defect	Designjet 2000...	Large Format Pri...

Copyright © 2014 Aras All Rights Reserved. aras.com

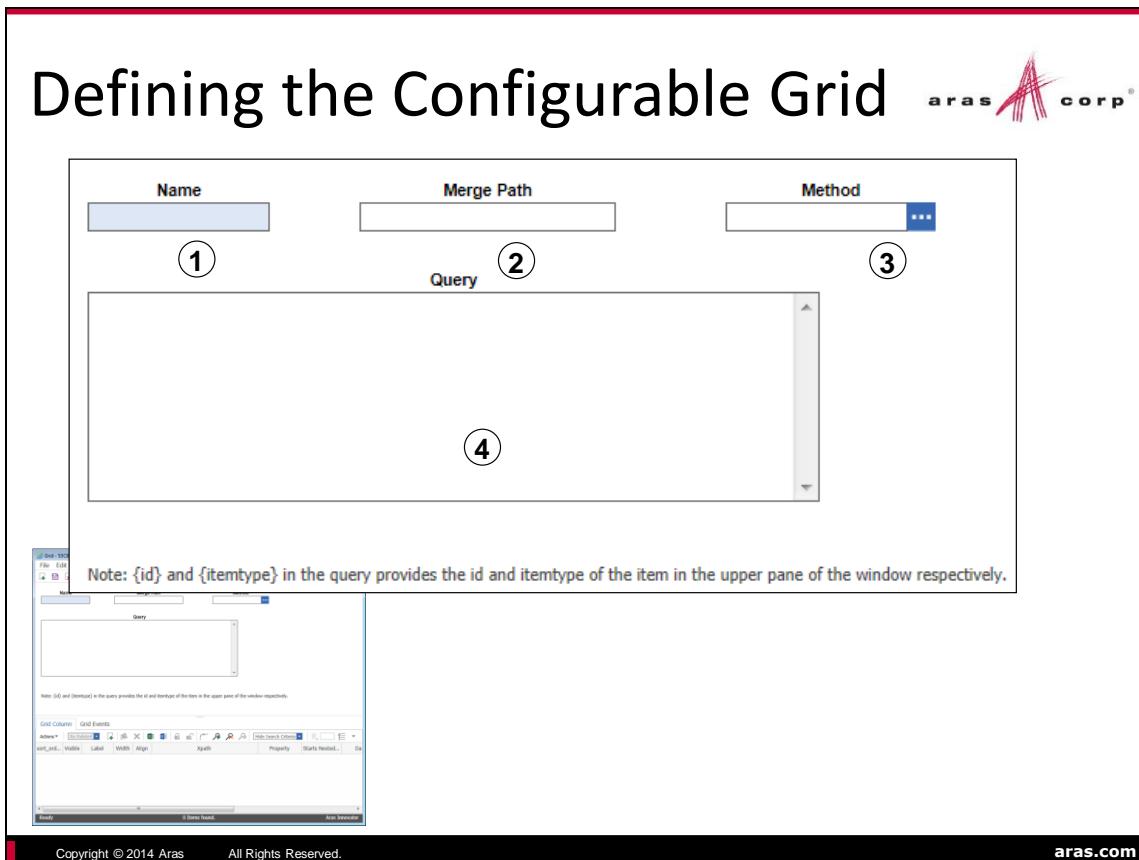
Once a grid has been configured, a user can then review the data presented in a spreadsheet format where each top level item defines a row of the grid.

Each column can represent data on the top level item as well as related data. If more than one related item exists from the top level item column “nesting” is supported which will display the contents of the nested items in the same top level row.

Grids support both insert and delete of items and related items as well as editing of individual cells (Method code required). Cell events are also available to validate or populate data using additional Methods.

In this example, the starting item is a Customer which is related to one or more Change Request Items (source item) which have a request number, title and type (classification). Each request can then be related to one or more Product Items.

When a user selects a Customer they would like to see all of the Change Requests associated with that Customer.



To Define a Configurable Grid

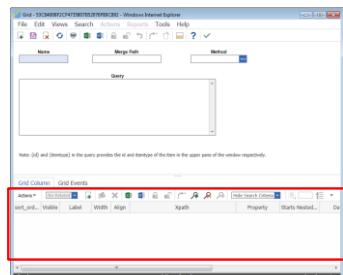
Select Administration > Grids from the TOC and create a new Grid Item. The following fields are available:

- ① **Name**
Name of the Grid Item.
- ② **Merge Path**
XPath Query which indicates where the results of the grid should be merged into the current parent Item DOM if an Item is saved to the database.
- ③ **Method**
Client Method to populate the grid or supply additional logic after the AML Query has executed.
- ④ **Query**
AML Query used to populate the DOM for the grid control use.

Defining Grid Columns



- Sort Order
- Visible
- Label
- Width
- Align
- XPath
- Property
- Starts Nested
- Data Type
- Sort
- Select Query
- Select Method
- Source Form



Copyright © 2014 Aras

aras.com

To Define the Grid Columns

Select the Grid Column tab in the Grid form and add a new relationship. The following fields are available:

Sort Order

Order of the columns displayed in the Grid from left to right.

Visible

Hide or show Grid column (useful for retrieving data that may be need for update but not editing/display).

Label

Column name displayed grid header (support multiple languages).

Width

Size of the grid column (number of characters).

Align

Left, right, center display of data in column.

XPath

The XPath query that will be used to show data from the Item DOM (result of Query or Method).

Starts Nested

If the XPath query results in multiple related Items the data should be shown in a “nested” row on the grid. This feature allows you to display hierarchical data to a user.

Data Type

- Text – normal text display.
- Default – use standard Innovator display behavior for property type.
- Federated – a Method will be used to calculate the value.
- Bound Select – single select dropdown list bound to values in a list (created with Select Query described later in this unit). User must choose one of the presented values.
- UnBound Select - single select dropdown list created from values in a list (created with Select Query described later in this unit). User can select a choice or enter their own.
- Bound MultSelect – same as Bound Select except multiple choices are allowed (returned as comma delimited string).
- UnBound MultSelect – same as UnBound Select except multiple choices are allowed (returned as comma delimited string).
- Dependent – Item is dependent on results in another field.
- Sort – this column will be used to sort on.

Sort

Specifies how the column will sorted when data is displayed in the grid. Similar to ItemType properties OrderBy option.

Select Query

AML query used to populate dropdown bound or unbound select lists.

Select Method

Method used to populate dropdown bound or unbound select lists.

Select Form

Optional Form to be displayed to allow data entry for a column. Requires a custom Method.

Defining Grid Events



- OnCopy
- OnPaste
- OnMenuInit
- OnMenuItemClick

A screenshot of a software application window titled "Grid Column" with a tab labeled "Grid Events" selected. The main area shows a table with columns: Actions, Name, Method Type, Ver, execution_allo.., Comments, Event, and sort_order. A row is selected for "CopyData" with "JavaScript" in the Method Type column and "World" in the Comments column. The "Event" column for this row has a dropdown menu open, showing four options: OnCopy, OnPaste, OnMenuInit, and OnMenuItemClick. The "sort_order" column for the selected row is also highlighted with a yellow background. The bottom of the window displays copyright information: "Copyright © 2014 Aras All Rights Reserved." and the website "aras.com".

To Define Grid Events

The configurable grid control supports four events available from the Grid Events tab on the Grid form.

OnCopy

If a user copies a cell or row (Ctrl + C) this event is executed to process the selected row.

OnPaste

If a user pastes a cell or row (Ctrl + V) this event is executed to process the row.

On MenuInit

If a user right clicks on an editable cell a context menu is presented. This event is triggered just before the menu is displayed.

OnMenuItemClick

User clicks on a context menu choice.

Defining the AML Query



- DOM returned to Grid must return one or more root Items
- Each root item can return any number of related Items
- Properties from Items at any level can be displayed as column data (using XPath notation)
- If a root Item contains multiple nodes the property display can be nested
- The current Item ID and ItemType is passed to Query:
 - {id}
 - {itemtype}

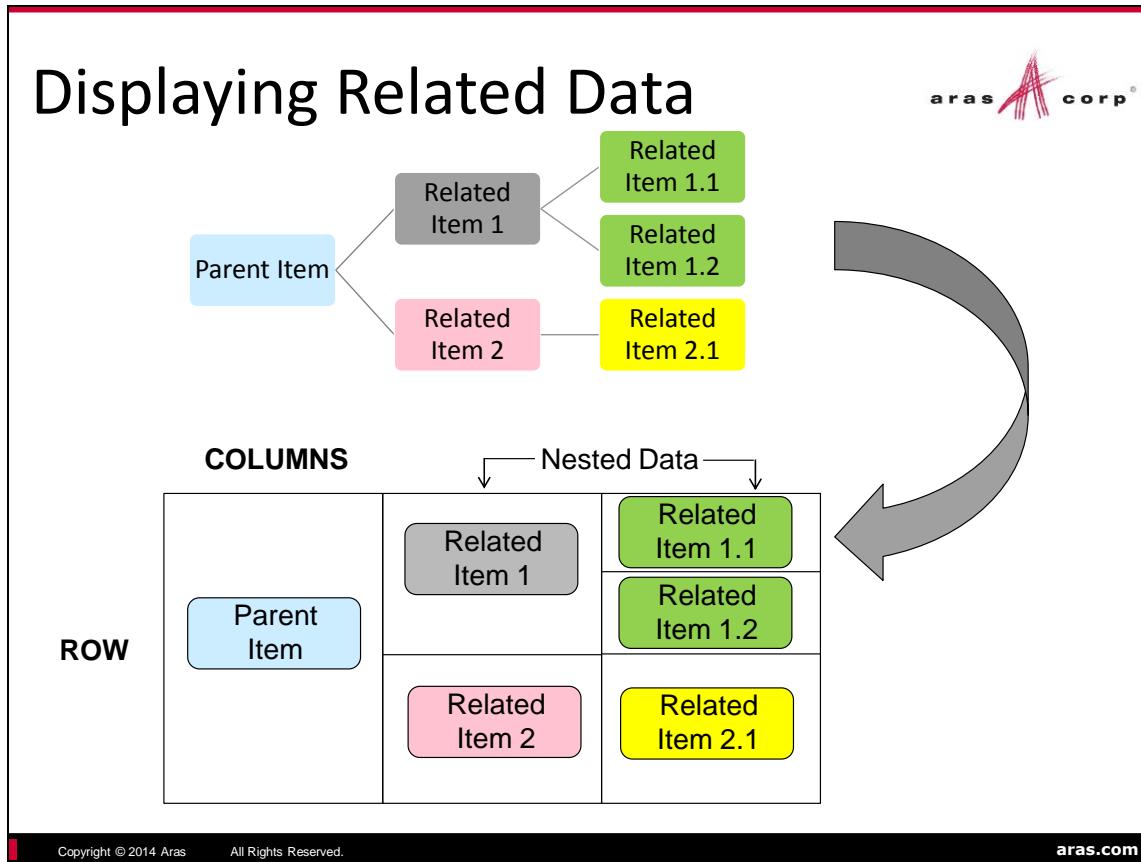
Copyright © 2014 Aras All Rights Reserved.

aras.com

AML Query Requirements

The inDom that is returned for the grid to process must follow this format:

```
<Result>
  <Item type='ParentItem'>
    <description>Item 1</description>
    <Relationships>
      <related_id>
        <Item type='RelatedItem'>
          </Item>
        </related_id>
      </Relationships>
    </Item>
  <Item type='ParentItem'>
    <description>Item 2</description>
    <Relationships>
      <related_id>
        <Item type='RelatedItem' />
      </related_id>
    </Relationships>
  </Item>
</Result>
```



This diagram illustrates how the inDOM data is translated into a Configurable Grid for display.

The control uses a format similar to a spreadsheet to “nest” related hierarchical Items.

Reviewing XPath Basics



- Industry standard syntax to search a DOM
- Basic location path expressions:
 - /node (match root node)
 - //node (match node anywhere)
 - node/childnode
 - node/childnode[property="value"]
 - node[@attribute="value"]
 - node/childnode[1]
 - . (single period – select current node)
 - .. (double period – select parent node)

Creating a Configurable Grid assumes you have some knowledge of the XPath query language. This course is not designed to make you an expert in XPath but you can use some basic expressions to query a DOM if you are not familiar with the language.

There are numerous resources for learning XPath both on-line and in many published books and articles. A handy starting tutorial is also available on-line at:

http://articles.techrepublic.com.com/5100-10878_11-1054416.html



Reviewing Xpath Examples

- Locating a source Item
 - source_id/Item
- Locating a Relationship Item
 - source_id/Item/Relationships/Item
- Locating a related item from a Relationship
 - source_id/Item/Relationships/Item/related_id/Item
- Locating a specific Relationship Item
 - source_id/Item/Relationships/Item[@type="Part BOM"]

Copyright © 2014 Aras All Rights Reserved.

aras.com

These examples show some common XPath expressions used with an AML Query.

The first example will locate a source Item in a Relationship Item.

The next example gets the Relationship Item from the source Item.

The third example demonstrates how to locate a related Item from a Relationship Item.

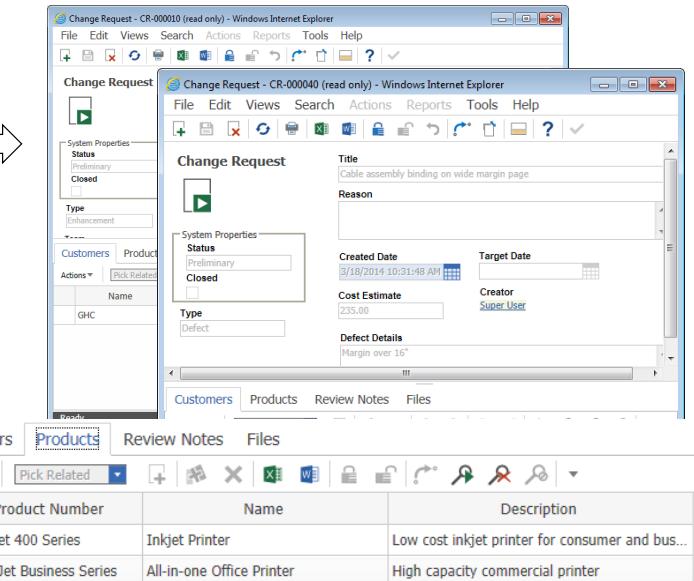
You can also qualify the Item search by using attributes (e.g. type).

Reviewing the Customer Change Request Example



Customer

Change Requests



Product Number	Name	Description
DeskJet 400 Series	Inkjet Printer	Low cost inkjet printer for consumer and bus...
OfficeJet Business Series	All-in-one Office Printer	High capacity commercial printer

Product(s)Attached to Request

Copyright © 2014 Aras All Rights Reserved. aras.com

This working example shows how a Change Request can be related to a Customer Item.

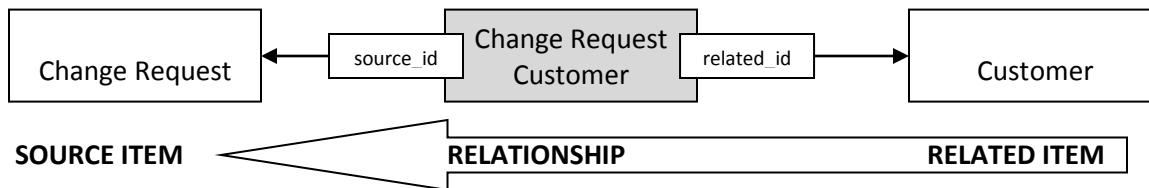
The users of the solution would like to be able to see all of the Change Requests that “belong” to a Customer as well as related Product data for each request in a concise display.

Each Change Request in this example supports one or more Products.

Reviewing the AML Query

- AML query receives the Customer id using {id}
- Query searches for Change Request Customer Relationships that have are related to current Customer
- Gets the source Item (Change Request) and retrieves the Product Item(s) (nested)

A common use for Configurable Grids is a “reverse” lookup. The current context Item is the related Item in a relationship and the source Item(s) is the desired result.



Review the AML Query below to see how this is accomplished:

```
<Item type="Change Request Customer" action="get">
  <related_idrelated_idsource_idsource_id
```

sort_order	Visible	Label	Width	Align	Xpath	Property	Starts Nested...
128		Request Number	100	Left	source_id/Item	_item_number	
256		Title	100	Left	source_id/Item	_title	
384		Type	100	Left	source_id/Item	classification	
512		Product Item	100	Left	source_id/Item/Relationships/Item/related_id/Item	item_number	
640		Product Name	100	Left	.	name	

Copyright © 2014 Aras All Rights Reserved. aras.com

Once the Query has been defined, you then specify the column information including the XPath query for each column data element.

In this example, the source Item information will be presented in the first 3 columns. A new nested data relationship (Products) is defined on column 4.

In column 5 the Product name property is specified. Because the Product property item_number and name are returned in the same nest, the XPath expression for column 5 uses the period (.) to access the name property value. Note that the Starts Nested checkbox is unchecked in column 5 indicating the property belongs to the current nest level.

If you need to nest within a nest - the current position of the XPath query will be positioned at the level set on the previously nested XPath statement.

Review these expressions with the AML Query on the previous page to see how the data will be extracted.

Creating the Grid Relationship



- Define a new Relationship from Source Item
- Specify the Grid in the Relationship View of the RelationshipType

The screenshot shows the 'Relationship View' tab selected in the top navigation bar. Below the toolbar, there is a table with columns: Name, Start Page, Parameters, Grid [...], and Form [...]. A row is selected with the value 'World' in the 'Name' column. Two numbered circles point to specific fields: circle 1 points to the 'Grid [...]' field which contains 'Change Request Product', and circle 2 points to the 'Form [...]' field which is empty.

	Name	Start Page	Parameters	Grid [...]	Form [...]
	World ①			Change Request Product	②

Copyright © 2014 Aras All Rights Reserved. aras.com

To Create the Grid Relationship

Define a new RelationshipType (or edit existing) that defines the Parent Item with the Configurable Grid. The following fields are available:

- ① **Name**
Identity that will be allowed to work with this Grid
- ② **Grid**
Name of the Configurable Grid.

In this example, a RelationshipType named *Customer Change Request* is created on the Customer ItemType and then Configurable Grid named *Change Request Product* is applied to the RelationshipType. When a user opens a Customer Item, the Grid will be populated.

Testing the Grid

aras corp®

Each row is a returned Item

Request Number	Title	Type	Product Item	Product Name
CR-000040	Cable assembly...	Defect	DesignJet 2000...	Large Format Pri...
CR-000010	Update to Windo...	Enhancement	DeskJet 400 Seri...	Inkjet Printer
			OfficeJet Busines...	All-in-one Office...

Nested relationships

The screenshot shows a web application interface. At the top, there's a navigation bar with links like 'File', 'Edit', 'Views', 'Search', 'Actions', 'Reports', 'Tools', and 'Help'. Below the navigation is a form for a 'Customer' record, with fields for 'Name', 'Main Phone', 'Main Fax', 'Contact Name', 'Address', 'City', 'State', 'Country', and 'Zip Code'. A red box highlights a tab labeled 'Design Requests' in the bottom left corner of the customer form. To the right of the customer form, a grid table is displayed, which is also highlighted with a red box. This grid contains the same data as the one shown above, representing nested relationships between the customer and the design requests.

Copyright © 2014 Aras All Rights Reserved. aras.com

Once a grid has been configured, you can open the source item and display the tab to view the grid query. In this example, requests have been returned from the current customer.

Starting from the left, each row in the grid represents a returned item from the AML query. Nested relationships are shown as nested rows in the item row.

Adding Select Lists

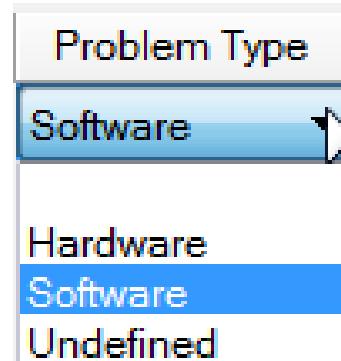


■ Bound Select

- Data must be selected from list presented
- Single/Multi-select support

■ Unbound Select

- Data may be selected from list presented
- User can enter text as well
- Single/Multi-select support



Any property that is already defined as a List datatype in an Item will automatically be displayed as a dropdown list if you choose the “Default” Datatype setting in the Configurable Grid. Remember, Default means to show the property using standard UI Innovator behavior.

You can also create additional drop down lists that display data from any source by creating a Select Query or Method and using the Bound Select or Unbound Select choices.



Defining a Select Query

- AML query returns one or more Items
- Items returned must have property named "label" (display) and "value" (stored in Item)
- List is presented when user selects column
- F2 support for unbound selects

A simple way to create a list is to simulate a List property by creating a Query or Method that returns Items in the following format:

```
<Item>
  <label>First Choice</label>
  <value>First Value</value>
</Item>
<Item>
  <label>Second Choice</label>
  <value>Second Value</value>
</Item>
```

Custom Select Lists



- Custom Form can be provided
- Custom JavaScript Method can be provided to display custom window – provide custom logic
- F2 key available to invoke custom list or window

For more examples of Unbound and Bound selects see the QP Planning Grids provided in the standard InnovatorSolutions database.



Summary

In this unit you learned how to create a Configurable Grid to display and edit data using a spreadsheet style user interface.

You should now be able to:

- ✓ Edit Data in a Configurable Grid
- ✓ Define the Grid Column and Events
- ✓ Build an AML Select Query
- ✓ Use XPath to Query AML Documents
- ✓ Configuring Grid Columns
- ✓ Build a Select List
- ✓ Attach a Grid to a Relationship



Lab Exercise

Goal:

Be able to create a Configurable Grid to display a collection of Sales Orders related to a Customer.

Scenario:

In this exercise, you will create a Configurable Grid that displays all of the Sales Orders for a Customer.

Create AML Expression

1. Using AML Studio create an AML Request that returns all Sales Orders (and related Part items) that are associated with a Customer. Use the ID of a known Customer Item to test the request. Make sure data is returned from the request before building the new Grid.



Create New Grid

2. Create a new Grid and paste the AML request you defined above into the Query field. Make sure to remove the <AML> tags from the request.
3. The Grid should return the following columns:

Order	Ship Method	PO Number	Qty	Part Number	Part Name
-------	-------------	-----------	-----	-------------	-----------

Supply the appropriate XPath query and property name for each column in the grid.

Sales Order (source) Items can be located using the XPath expression:

source_id/Item

Relationship Items can be accessed using the expression to access quantity (nested):

source_id/Item/Relationships/Item

And Part Items (inside of Quantity nested property) can be located using the expression:

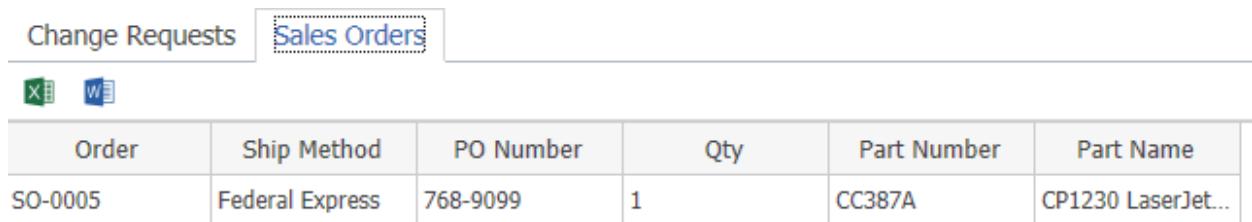
```
./related_id/Item
```

Create New Relationship

4. Create a new Relationship named **Customer Sales Order** on the Customer ItemType and label the tab *Sales Orders*. (The related Item is not important because you will customize the relationship in the next steps.)

Attach the Grid to the RelationshipType

5. Edit the **Customer Sales Order** RelationshipType and add the Grid to the Relationship View tab.
6. Test the Grid by opening a Customer Item and viewing the Sales Orders tab. Your result should resemble the following:



The screenshot shows a software interface with a navigation bar at the top. The 'Sales Orders' tab is selected, indicated by a blue border around its label. Below the navigation bar, there are two small icons: one for Excel and one for Word. A horizontal line separates the navigation from the main content area. The main content area contains a table with six columns: Order, Ship Method, PO Number, Qty, Part Number, and Part Name. There is one row of data in the table, corresponding to the item shown in the navigation bar.

Order	Ship Method	PO Number	Qty	Part Number	Part Name
SO-0005	Federal Express	768-9099	1	CC387A	CP1230 LaserJet...



Appendix E: Tracking Login and Logout

Overview: Three system events can be configured to track successful user login, failed login and user logout. You can provide conditional business logic to extend (or prevent) these events with a server Method. A class utility is also available to create System Log Items which track login activity.

Objectives:

- ✓ Overview of System Events
- ✓ Configuring System Events
- ✓ Creating a System Log Item with Server Method.



System Events

- Allow tracking and validation of
 - Failed Login
 - Successful Login
 - Logout
- Related to System Events Handler Method
- Handler Methods can create a System Events Log Item defined by a System Events Log Descriptor Template

- ▼ System Events
 - ★ System Events
 - ★ System Events Log
 - ★ System Events Log Descriptor

System Events

System Events allow you track successful logins, failed logins as well as session logouts. Each event can be associated with a System Events Handler Method that can access the current login name as well as other session information.

An additional System Events Log Descriptor can be created for each event type that contains a log level number (to determine when logging should occur) as well as a custom log message. Using a provided IOM method you can create System Events Log Items in the database based on the configuration settings in the Log Descriptor.

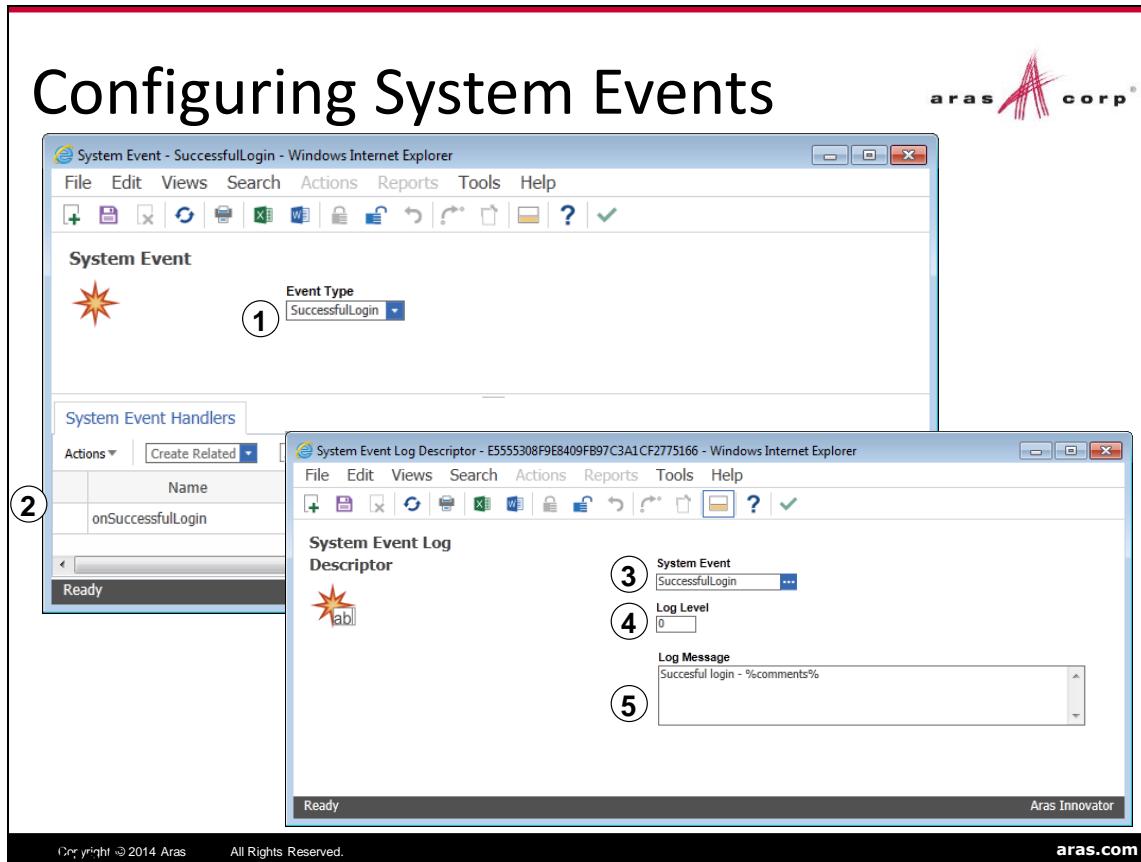
Logging System Events



- Define a new System Event for each Event Type
 - Failed Login
 - Successful Login
 - Logout
- Create a System Events Log Descriptor for each Event Type
- Define one or more Method handlers for each System Event
 - Create System Events Log Items

Logging System Events

1. To log a system event you must perform the following steps:
2. Create a new System Events Item and select the appropriate Event Type.
3. Create a System Events Log Descriptor Item for each Event Type to be configured.
4. Create one or more server Methods for each System Event to handle the event. Typically the Method will create a System Events Log Item to indicate successful or unsuccessful logins.



Configuring System Events

To configure a System Event:

1. Select Administration > System Events from the TOC to create a new event item. Select the Event Type from the drop down list.
2. Select one ore more server Methods that will be executed when this event is triggered.
3. Select Administration > System Events Log Descriptor from the TOC to create a new descriptor item. Choose the Event Type from the selection field.
4. Enter a log level number. This number indicated if the event should be triggered or not and is compared to the System Variable named *SystemEventLogLevel*. If the log level number is greater than or equal to the system event log level the event will be triggered, otherwise it will be ignored.
5. Enter a log message that will appear in the System Events Log Item when it is created by the system event handler Method. The following variables are available for substitution at runtime: *%login_name%*, *%ip_address%*, *%method_name%*, *%event_type%*, *%session_id%*, *%comments%*.

Tracking Successful Login



```

//Store Last Successful Login Name

Item itm = this newItem("Variable", "edit");
itm.setAttribute("where", "[variable].name='LastLogin'");
itm.setProperty("value", eventData.LoginName);
itm.apply();

//Create System Events Log

CCO.Utilities.CreateSystemLogRecord("SuccessfulLogin",
    "onSuccessfulLogon", eventData.LoginName,
    "Login Successful");

return this;

```

Copyright © 2014 Aras All Rights Reserved.

aras.com

The example above demonstrates an event handler Method for a SuccessfulLogin event.

This method first stores that name of the login in a System Variable named LastLogin.

The Method then creates a new System Events Log Item using the CreateSystemLogRecord method available in the CCO (Core Context Object) class.

The CreateSystemLogRecord takes the following arguments:

- Event Type (string) – "SuccessfulLogin", "FailedLogin", or "Logout" are valid system event types.
- Method Name (string) – name of the handler Method.
- Login Name (string) – the user login name. A reserved object (eventData) contains an attribute named LoginName which contains the current user's login.
- Comment (string) – additional comments that are added the System Events Log Item (if a comment has not been provided in the System Events Log Descriptor).



Summary

In this unit you learned how to pass parameters between server event Methods.

You should now be able to:

- ✓ Implement the RequestState Class with Server Events
- ✓ Add RequestState Object Keys
- ✓ Retrieve RequestState Object Keys
- ✓ Clear the RequestState of key/values



Appendix F: Working with Relationship Grids

Overview: Relationships are presented to the user as a tabbed grid on a standard tear-off window. You can react to grid events using row and cell callback Methods to provide additional business logic for the grid.

- Objectives:**
- ✓ Access the Tab Bar Control
 - ✓ Access the Grid Control
 - ✓ Create Custom Grid Row/Cell Callback Methods

Working with Relationship Tabs



- Useful for disabling or removing tab from view in Relationship Grid
- Uses the `relTabbar` Control

The screenshot shows a standard Aras Innovator Form interface. At the top, there is a horizontal bar with four tabs: 'Customers' (which is selected and highlighted in blue), 'Products', 'Review Notes', and 'Files'. Below this bar is a toolbar containing several icons: 'Actions' with a dropdown arrow, 'Pick Related' with a dropdown arrow, a plus sign, a delete icon, an Excel icon, a Word icon, a lock icon, and a refresh/circular arrow icon. The main area of the form is a data grid with three columns: 'Name', 'Main Phone', and 'Contact Name'. A single row is visible, showing 'GHC' in the Name column, '49.89.21.99.0000' in the Main Phone column, and 'Verner Kohler' in the Contact Name column. The bottom of the screen has a dark footer bar with the text 'Copyright © 2014 Aras All Rights Reserved.'

`top.relationships.relTabbar`

The standard Aras Innovator Form always contains a grid with a set of tab controls to display relationship information to the user.

You can control the tab(s) by disabling or hiding/showing the tabs.

The `relTabbar` control is accessed from the DOM using the expression `top.relationships.relTabbar`

Tab Bar Methods



- `setTabVisible (tabID, boolean)`
 - Hides (or shows) a Tab in a Grid
- `setTabEnabled (tabID, boolean)`
 - Disables/Enables a Tab
- `GetSelectedTab();`
 - Returns id of user selected tab
- `_getTabByLabel("tab label");`
 - Returns tab object by label name

Copyright © 2014 Aras All Rights Reserved.

Tab Bar Methods

Several tab bar client methods are available to manipulate the tab bar controls. Examples of these methods are demonstrated on the following pages.



Hide or Disable a Tab

■ Hide a tab (Form onLoad Event)

```
var tabbar = parent.relationships.relTabbar;
var tab = tabbar._getTabByLabel("Documents");
var tabid = tab.tabID;
tabbar.setTabVisible(tabid, false);
```

■ Disable a tab (Form onload Event)

```
var tabbar = parent.relationships.relTabbar;
var tab = tabbar._getTabByLabel("Documents");
var tabid = tab.tabID;
tabbar.setTabEnable(tabid, false);
```

Copyright © 2014 Aras All Rights Reserved.

These examples are designed to work with the Form onLoad Event.

The setTabVisible method allows you to hide or show the tab while the setTabEnabled method enables/disables a tab control. Use the boolean true or false as the second argument to indicate whether a tab should be shown/enabled (true) or hidden/disabled(false).

Note

This example uses the _getTabByLabel method to select a control by label name. You can also substitute this method with the GetSelectedTab() function to act on the tab currently selected by a user. GetSelectedTab() returns the tab id of the actively selected tab.

```
var tabid = tabbar.GetSelectedTab();
```

Example

The following example hides a tab labeled *Customers* when a Form opens. The Method is associated with the OnLoad Form Event. A timeout has been added to allow for the form to fully display before the tab operation is executed.

```
top.hideTab = function() {  
  
    if (!parent.relationships ||  
    !parent.relationships.relTabbarReady ||  
    parent.relationships.relTabbar.GetTabOrder(" | ") === "") {  
        setTimeout("top.hideTab", 30);  
        return;  
    }  
  
    var showTab = false;  
  
    var tabbar = parent.relationships.relTabbar;  
    var tabID = tabbar.GetTabId("Customers");  
    if (!tabID) {  
        top.aras.AlertError("Cannot access tab");  
    }  
    else {  
        tabbar.SetTabVisible(tabID, showTab);  
    }  
    return;  
};  
  
setTimeout("top.hideTab()", 30);
```

Working with Relationship Grids



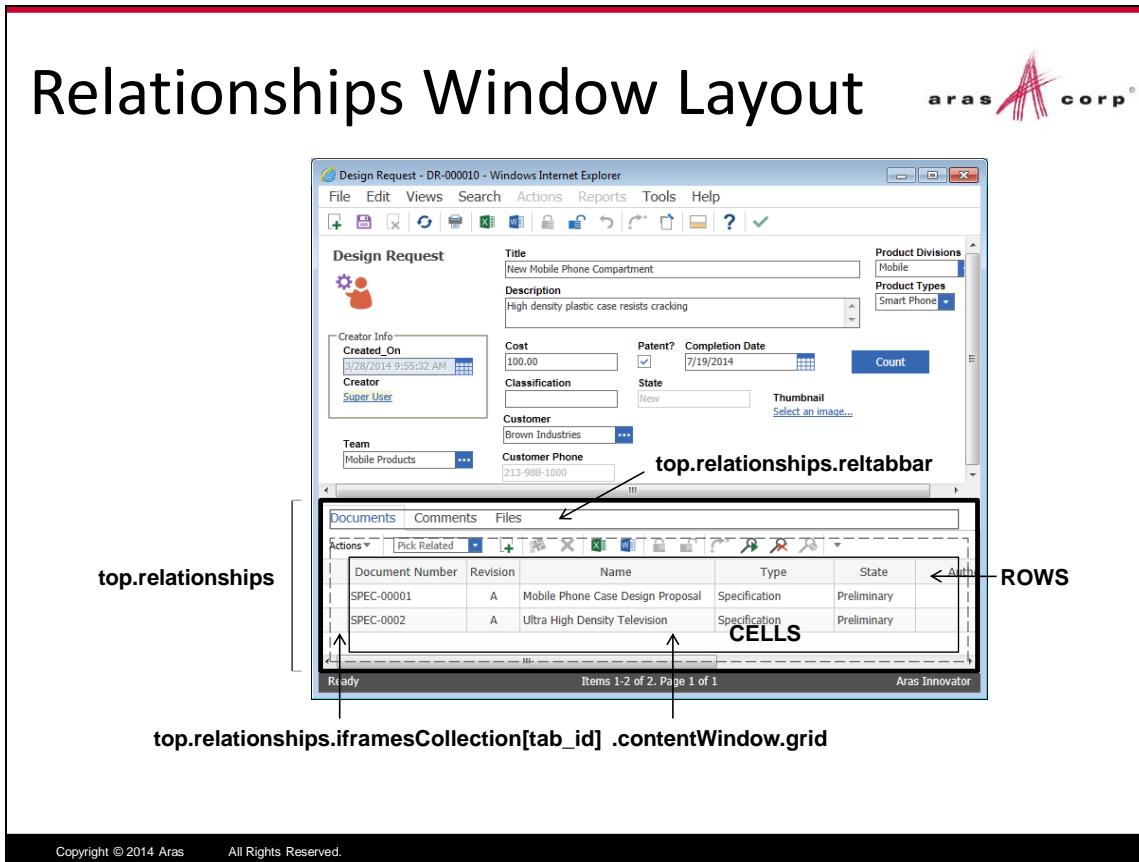
- Standard Tear-off Window contains a Relationship Grid
- Represented by the Grid Control
- Reacts to User Input using Grid Events

A screenshot of a software interface showing a relationship grid. At the top, there are tabs for 'Customers', 'Products', 'Review Notes', and 'Files'. Below the tabs is a toolbar with buttons for 'Actions', 'Pick Related', and various file operations like add, delete, and export. The main area is a grid with columns labeled 'Name', 'Main Phone', and 'Contact Name'. A single row is selected, containing the values 'GHC', '49.89.21.99.0000', and 'Verner Kohler'. The word 'Cells' is centered below the grid, with two arrows pointing upwards from it to the selected row.

Rows	Name	Main Phone	Contact Name
	GHC	49.89.21.99.0000	Verner Kohler

Copyright © 2014 Aras All Rights Reserved.

Relationship grids represent relationships from the current source item. You can modify and provide custom logic in the standard relationship grid by accessing the grid control.



Relationships Window Layout

Each tear-off window can contain a relationships grid which allows users to manipulate relationship data from the form.

When a user selects a tab a corresponding iFrame appears based on the id of the tab. A content window is nested in each iFrame which contains the corresponding relationship grid data in rows and cells as shown.



Relationship Grid Methods

- **GetCellValue**
 - Return value of a cell
- **cells**
 - Set or get value of a cell
- **getRowCount**
 - Get number of grid rows

Copyright © 2014 Aras All Rights Reserved.

Relationship Grid Methods

The relationship grid container allows you to either set or get the value of a cell as well as obtain the number of rows in the grid.

The methods shown above are current as of Version 10 SP1 of Aras Innovator and may be amended in future releases.

Accessing Grid Control



■ Create Method for Form onLoad Event

```
document.TabRowCount = function() {
    var tabbar = top.relationships.relTabbar;
    var tabid = tabbar.GetSelectedTab();
    var ifrm = top.relationships.iframesCollection[tabid];
    var grid = ifrm.contentWindow.grid;
    var count = grid.getRowCount();
    return count;
};
```

■ Call from any Field Event

```
var numrows = document.TabRowCount();
document.thisItem.setProperty("row_count", numrows);
```

Copyright © 2014 Aras All Rights Reserved.

Loading JavaScript Functions with onLoad Event

In this example, a client Method is defined that creates a JavaScript function that will return back the number of rows in a grid. Note that the function will reside on the “page” representing the form in the browser but not be executed until called by a Field Event.

The Field Event code can then call this function anytime the grid count is required by a control.

Note

The example above accesses a tab selected by the user. The tab id is obtained by calling the GetSelectedTab function of the relTabbar control. Each grid is contained in a iFrame collection which is indexed by the corresponding tab id.



Creating Custom Grid Row/Cell Callbacks

- Row Grid Events respond to user interaction with the grid control rows (delete, insert, select)
- Cell Grid Events respond to user interaction within a relationship grid edit cell
- Associated Methods receive arguments relative to the grid control
- Context Item is available using:

parent.thisItem

Copyright © 2014 Aras All Rights Reserved.

To respond to a user's interaction with a Relationship grid, you create client Methods and associate them with events on the Grid Rows and Cells.

Because the grid is considered a child of the current HTML document you can use the `parent.thisItem` reference to access the context of the current Item associated with the grid.

Configuring Row Events



- Row Grid Events defined on the RelationshipType Type
 - OnSelectRow
 - OnInsertRow
 - OnDeleteRow

A screenshot of the Aras Relationship View interface. The top navigation bar includes tabs for "Grid Events" (which is selected), "Relationship View", "Exclusion", "Hide In", and "Hide Related In". Below the navigation is a toolbar with various icons for actions like pick related, insert, delete, and search. The main area is a grid table with columns: Name, Method Type, Ver, execution_allo.., Comments, Event, and sort_order. A row is selected with the name "_GridInsertRow", method type "JavaScript", version 3, and comments "World". The "Event" column dropdown menu is open, showing three options: OnSelectRow, OnInsertRow, and OnDeleteRow. The bottom of the interface shows copyright information: "Copyright © 2014 Aras All Rights Reserved."

	Name	Method Type	Ver	execution_allo..	Comments	Event	sort_order
	_GridInsertRow	JavaScript	3	World		OnInsertRow	128

Row events are defined on the RelationshipType template in the Grid Events tab.

Three events allow you respond to the insertion, selection or deletion of a grid row by a user.

Grid Row Event Method Arguments



- Method receives the following arguments:
 - **relationshipID** = ID for the Relationship Item. This is also the selected row ID in the Grid control
 - **relatedID** = the ID for the related Item. (Blank for null relationships)
 - **gridApplet** = handle to the Grid control

The OnInsertRow, OnSelectRow and OnDeleteRow all receive three arguments which allow you to programmatically access the current Related Item, the current Relationship as well as the grid control itself.

Defining a Grid Row Event Method (OnInsertRow)



```
var team = parent.thisItem.getPropertyAttribute("team_id",
    "keyed_name");
var msg = "Assigned team: " + team;
var x = gridApplet.cells(relationshipID, 1).setValue(msg);
```

Get Property from Parent Item

Set Grid Cell Value

```
var relItem = parent.thisItem.getItemsByXPath("//Item[@id='"
    + relationshipID + ']");
relItem.setProperty(" assigned team", msg);
```

Get Item in dom cache to set value

Comments	Assigned Team
Under review	Assigned team: Mobile Products

Copyright © 2014 Aras All Rights Reserved.

Column set in control and cache

In this example, the cells method of the grid control is used to set the value of the column when a new row is inserted. The column number is determined by the sort order of the properties assigned to the relationship and is zero based.

The relationshipID is passed to the grid row event as a standard argument and is used to locate the relationship Item in the cached DOM by using an Item method discussed earlier in the course (getItemsByXPath).

Finally, the property is set in the client cache for the relationship so when the user saves the parent Item this information will be saved to the database as well.

Note

You must set both the control value as well as the Item cache property for the value to be successfully displayed and saved to the database.

Configuring Cell Events



- Cell Grid Events defined on the **Property** of the corresponding Grid Cell
 - onEditStart
 - onEditFinish
 - onChangeCell
 - onSearchDialog

A screenshot of the Aras Event configuration interface. The window title is 'Event'. The 'onSearchDialog' checkbox is selected. The main table has columns: Name, Method Type, Ver, execution_allo.., Comments, and Event. A row for '_GridCellChange' is selected, showing JavaScript as the Method Type and '1|World' in the execution column. The 'Event' dropdown is open, showing options: OnChangeCell, OnEditStart, OnEdit, OnEditFinish, OnChangeCell, Default Search, and OnSearchDialog. The 'Default Search' option is highlighted.

Name	Method Type	Ver	execution_allo..	Comments	Event
_GridCellChange	JavaScript	1 World			OnChangeCell OnEditStart OnEdit OnEditFinish OnChangeCell Default Search OnSearchDialog

Each cell in the relationship grid is also capable of responding to a series of events while a user interacts with the grid.

A grid cell is represented as a property value of either the relationship item or the related item based on the configuration of the relationship.

Access the appropriate ItemType and right click on property that represents the relationship cell. Select View Property to access the Event tab and select the appropriate event.

Note

Default Search is a deprecated event – use *onSearchDialog* for any new projects.

Grid Cell Event Method Arguments



- Method receive the following arguments:
 - **relationshipID** = ID for the Relationship Item. This is also the selected row ID in the Grid control
 - **relatedID** = the ID for the related Item. (Blank for null relationships)
 - **gridApplet** = handle to the Grid control
 - **propertyName** = name of the Property for the currently selected cell
 - **colNumber** = zero based column position number in the grid of the current cell

Copyright © 2014 Aras All Rights Reserved.

The client Method receives a set of standard parameters that can be used to identify the current relationship, the related Item, the current Property name of the cell as well as its column number (zero based).

Defining a Grid Cell Method (OnChangeCell)



```

var g = gridApplet;

var val = g.GetCellValue(relationshipID, colNumber);

var upperval = val.toUpperCase();

g.cells(relationshipID, colNumber).setValue(upperval);

var relitm =
    parent.thisItem.getItemsByXPath("//Item[@id='"
    + relationshipID + "']");
relitm.setProperty("_comments", upperval);

```

Comments	Assigned Team
UNDER REVIEW	Assigned team: Mobile Products

Copyright © 2014 Aras All Rights Reserved.

In this example, the value of the current cell is retrieved using the GetCellValue method. The letters of the string are then set in upper case using the JavaScript toUpperCase function.

The modified string is then set back into the same cell using the cells method and the DOM is updated so the change will be reflected when the parent Item is saved to the database.

This Method is then attached to the _comments property of the Design Request Comments ItemType using the OnChangeCell event

Name	Required	Data Type
_comments	<input type="checkbox"/>	String

Name	Method Type	Ver	execution_all...	Comments	Event
_GridCellChange	JavaScript	1	World		OnChangeCell



Summary

In this unit you learned how to create define custom logic for Relationships grids.

You should now be able to:

- ✓ Access the tabs in the Relationship Grid
- ✓ Create callback Methods for row and cell events in the Relationship grid

This page intentionally left blank.



Lab Solutions

Overview: Solution code for all Lab Exercises.

Visual Basic solutions are shown in *italicized text*.

Unit 2 Solutions

1.

```
<Item type="Sales Order" action="get">
<multiple_shipment>1</multiple_shipment>
<shipping_method>UPS</shipping_method>
</Item>
```

2.

```
<Item type="Sales Order" action="get">
<or>
<multiple_shipment>1</multiple_shipment>
<shipping_method>UPS</shipping_method>
</or>
</Item>
```

3.

```
<Item type="Sales Order" action="get">
<Relationships>
<Item type="Sales Order Customer" action="get">
<related_id>
<Item type="Customer" action="get">
<city>New York</city>
</Item>
</related_id>
</Item>
</Relationships>
</Item>
```

4.

```
<Item type="Sales Order" action="get">
<ship_date condition="gt">2013-05-01T00:00:00</ship_date>
<Relationships>
<Item type="Sales Order Customer" action="get">
<related_id>
<Item type="Customer" action="get">
<city>New York</city>
</Item>
</related_id>
</Item>
</Relationships>
</Item>
```

5.

```
<Item type="Sales Order" action="add">
  <managed_by_id>
    <Item type="Identity" action="get">
      <keyed_name>Peter Smith</keyed_name>
    </Item>
  </managed_by_id>
  <shipping_method>UPS</shipping_method>
</Item>
```

6.

```
<Item type="Sales Order" action="add">
  <managed_by_id>
    <Item type="Identity" action="get">
      <keyed_name>Peter Smith</keyed_name>
    </Item>
  </managed_by_id>
  <shipping_method>UPS</shipping_method>
  <Relationships>
    <Item type="Sales Order Customer" action="add">
      <related_id>
        <Item type="Customer" action="get">
          <name>Brown Industries</name>
        </Item>
      </related_id>
    </Item>
    <Item type="Sales Order Part" action="add">
      <quantity>1</quantity>
      <related_id>
        <Item type="Part" action="get">
          <item_number>C4703A</item_number>
        </Item>
      </related_id>
    </Item>
  </Relationships>
</Item>
```

7.

```
<Item type="Sales Order" action="edit"
  where="[sales_order].multiple_shipment='1'">
  <shipping_method>UPS</shipping_method>
</Item>
```

8.

```
<Item type="Sales Order" action="edit"
  where="[sales_order].multiple_shipment='1'">
  <shipping_method>UPS</shipping_method>
</Item>
```

Developing Solutions

```
<Relationships>
<Item type="Sales Order Remarks" action="add">
<comment_text>Modified by AML
    Administrator</comment_text>
</Item>
</Relationships>
</Item>
```

Unit 3 Solutions

1.

```
Item so = this.newItem("Sales Order", "get");
so.setProperty("multiple_shipment", "1");
so.setProperty("shipping_method", "UPS");
return so.apply();

Dim so As Item = Me.newItem("Sales Order", "get")
so.setProperty("multiple_shipment", "1")
so.setProperty("shipping_method", "UPS")
Return so.apply()
```

2.

```
var so = this.newItem("Sales Order", "get");
so.setAttribute("select", "so_number");
so.setProperty("multiple_shipment", "1");
so.setProperty("shipping_method", "UPS");
return so.apply();
```

3.

```
Item so = this.newItem("Sales Order", "add");
so.setProperty("shipping_method", "Federal Express");
return so.apply();
```

```
Dim so As Item = Me.newItem("Sales Order", "add")
so.setProperty("shipping_method", "Federal Express")
Return so.apply()
```

4.

```
string d = DateTime.Now.ToString("yyMMddhhmmss");
Item so = this.newItem("Sales Order", "add");
so.setProperty("shipping_method", "Federal Express");
so.setProperty("po_number", d);
return so.apply();
```

```
Dim d As String = DateTime.Now.ToString("yyMMddhhmmss")
Dim so As Item = Me.newItem("Sales Order", "add")
so.setProperty("shipping_method", "Federal Express")
so.setProperty("po_number", d)
Return so.apply()
```

5.

```
Item so = this newItem("Sales Order", "edit");
so.setAttribute("where", "[sales_order].multiple_shipment='1'");
so.setProperty("shipping_method", "UPS");
return so.apply();
```

```
Dim so As Item = Me newItem("Sales Order", "edit")
so.setAttribute("where", "[sales_order].multiple_shipment='1'")
so.setProperty("shipping_method", "UPS")
Return so.apply()
```

6.

```
Item so = this newItem("Sales Order", "delete");
so.setAttribute("where",
    "[sales_order].created_on>='2013-06-26T00:00:00')";
return so.apply();
```

```
Dim so As Item = Me newItem("Sales Order", "delete")
so.setAttribute("where",
    "[sales_order].created_on>='2013-06-26T00:00:00'")
Return so.apply()
```

Optional:

```
string d = DateTime.Now.ToString("yyyy-MM-ddThh:mm:ss");
Item so = this newItem("Sales Order", "delete");
so.setAttribute("where",
    "[sales_order].created_on>=" + d + "'");
return so.apply();
```

```
Dim d As String = DateTime.Now.ToString("yyyy-MM-ddThh:mm:ss")
Dim so As Item = Me newItem("Sales Order", "delete")
so.setAttribute("where",
    "[sales_order].created_on>=" + d + "'")
Return so.apply()
```

Unit 4 Solutions:

1. Add the appropriate debug statement to the method **DebugMe**.

```
System.Diagnostics.Debugger.Break();
Innovator inn = this.getInnovator();
Item itm = this newItem("Sales Order", "get");
itm.setProperty("so_number", "SO-0005");
Item result = itm.apply();

string po = result.getProperty("po_number");

return inn.newResult(po);
```

```
System.Diagnostics.Debugger.Break()
Dim inn As Innovator = Me.getInnovator()

Dim itm As Item = Me newItem("Sales Order", "get")
itm.setProperty("so_number", "SO-0005")
Dim result As Item = itm.apply()

Dim po As String = result.getProperty("po_number")

Return inn.newResult(po)
```

Unit 5 Solutions

1.

```
string x = System.Environment.MachineName;  
return this.getInnovator().newResult(x);  
  
Dim x As String = System.Environment.MachineName  
return Me.getInnovator().newResult(x)
```

2.

3.

```
var today = new Date();  
var dd = today.getDate();  
var mm = today.getMonth()+1; //January is 0!  
var yyyy = today.getFullYear();  
if(dd<10){dd='0' + dd;}  
if(mm<10){mm='0' + mm;}  
today = yyyy+'-'+mm+'-'+dd;  
  
this.setAttribute('action', 'edit');  
this.setProperty('ship_date', today);  
  
return this.apply();
```

Unit 6 Solutions

1. AML Prompt Method

```
var inn = this.getInnovator();
var AML = prompt("Enter AML query:");
var AMLResult = inn.applyAML( AML );
var ss="http://localhost/Innovator10/Client/styles/default.xsl";
return AMLresult.applyStylesheet(ss,"URL");
```

2. Sum Method:

```
Innovator inn = this.getInnovator();
string sql=@"select sum(cost) as TotalCost from innovator.part";
Item qry = inn.applySQL(sql);
return qry;
```

```
Dim inn as Innovator = Me.getInnovator()
Dim sql as String =
    "select sum(cost) as TotalCost from part"
Dim qry as Item = inn.applySQL(sql)
Return qry
```

3. Server_Calculator_Generic:

```
Innovator inn = this.getInnovator();
string column = this.getProperty("column_name");
string table = this.getProperty("table_name");
string function = this.getProperty("function");
string SQL = "select " + function + "(" + column + ")" as " +
    function + " from innovator." + table;

Item totalcost = inn.applySQL(SQL);
return totalcost;

Dim inn As Innovator = Me.getInnovator()
Dim column As String = Me.getProperty("column_name")
Dim table As String = Me.getProperty("table_name")
Dim [function] As String = Me.getProperty("function")

Dim SQL As String = "select " & [function] & "(" & column & ")" as "
& [function] & " from innovator." & table

Dim totalcost As Item = inn.applySQL(SQL)
Return totalcost
```

4. AML Statement Example:

```
<AML>
  <Item type="Part" action="Server_Calculator_Generic">
    <column_name>cost</column_name>
    <table>part</table>
    <function>SUM</function>
  </Item>
</AML>
```

Unit 7 Solutions

1. Calculate Cost:

```
Innovator inn = this.getInnovator();

decimal total_cost = 0;
decimal unit_cost = 0;
decimal line_cost = 0;
int quantity = 0;
int count = 0;

this.fetchRelationships("Sales Order Part");
Item relationships = this.getRelationships("Sales Order Part");
count = relationships.getItemCount();
for (int i = 0; i < count; i++) {
    Item partRel = relationships.getItemByIndex(i);
    quantity=
        Convert.ToInt16(partRel.getProperty("quantity", "0"));
    Item part = partRel.getRelatedItem();
    unit_cost =
        Convert.ToDecimal(part.getProperty("cost", "0"));
    line_cost = quantity * unit_cost;
    total_cost = total_cost + line_cost;
}
return inn.newResult(total_cost.ToString());
```

```
Dim inn As Innovator = Me.getInnovator()

Dim total_cost As Decimal = 0
Dim unit_cost As Decimal = 0
Dim line_cost As Decimal = 0
Dim quantity As Integer = 0
Dim count As Integer = 0

Me.fetchRelationships("Sales Order Part")
Dim relationships As Item = Me.getRelationships("Sales Order Part")
count = relationships.getItemCount()
For i As Integer = 0 To count - 1
    Dim partRel As Item = relationships.getItemByIndex(i)
    quantity =
        Convert.ToInt16(partRel.getProperty("quantity", "0"))
    Dim part As Item = partRel.getRelatedItem()
    unit_cost =
        Convert.ToDecimal(part.getProperty("cost", "0"))
    line_cost = quantity * unit_cost
    total_cost = total_cost + line_cost
Next
Return inn.newResult(total_cost.ToString())
```

Unit 8 Solutions

1.

```
Innovator inn = this.getInnovator();
string shipmethod = this.getProperty("shipping_method");
string po_number = this.getProperty("po_number", "");
if (shipmethod=="Federal Express") {
    if (po_number=="") {
        return inn.newError("PO# Required");
    }
}
return this;
```

```
Dim inn As Innovator = Me.getInnovator()
Dim shipmethod As String = Me.getProperty("shipping_method")
Dim ponumber As String = Me.getProperty("po_number", "")
If shipdate = "Federal Express" Then
    If po_number = "" Then
        Return inn.newError("PO# Required")
    End If
End If
Return Me
```

2.

```
Innovator inn = this.getInnovator();
string shipdate = this.getProperty("ship_date" , "");
if (shipdate=="") {
    return inn.newError("Shipping Date Required");
}
return this;
```

```
Dim inn As Innovator = Me.getInnovator()
Dim shipdate As String = Me.getProperty("ship_date", "")
If shipdate = "" Then
    Return inn.newError("Shipping Date Required")
End If
Return Me
```

3.

Apply Method to onActivate event of Reject Order Workflow Activity

```
Item order = this.apply("Get Controlled Item");

Item rel =
    order.createRelationship("Sales Order Remarks", "add");
rel.setProperty("comment_text", "Sales Order Rejected");
return order.apply();

Dim order As Item = Me.apply("Get Controlled Item")

Dim rel As Item =
    order.createRelationship("Sales Order Remarks", "add")
rel.setProperty("comment_text", "Sales Order Rejected")
Return order.apply()
```

Unit 9 Solution

1. onBeforeVersion

```
RequestState.Add("part_type", this.getProperty("classification", ""));
return this;
```

```
RequestState.Add("part_type", Me.getProperty("classification", ""))
Return Me
```

2. onAfterVersion

```
string msg = "Type cannot be changed between versions.";
if (RequestState.Contains("part_type")) {
    string type = (string)RequestState["part_type"];
    if (type != this.getProperty("classification", "")) {
        return this.getInnovator().newError(msg);
    }
}
RequestState.Clear();
return this;
```

```
Dim msg as String = "Type cannot be changed between versions."
If RequestState.Contains("part_type") Then
    Dim type As String =
        DirectCast(RequestState("part_type"), String)
    If type <> Me.getProperty("classification", "") Then
        Return Me.getInnovator().newError(msg)
    End If
End If
RequestState.Clear()
Return Me
```

Unit 10 Solutions

1.

```
var directory = top.aras.getWorkingDir();
top.aras.confirm("Working Directory is: " + directory);
```

Unit 11 Solutions

1. Apply to the client **OnAfterNew** Event

```
var today = new Date();
var dd = today.getDate();
var mm = today.getMonth()+1;//January is 0!
var yyyy = today.getFullYear();
if(dd<10){dd ='0' + dd;}
if(mm<10){mm ='0' + mm;}
today = yyyy+'-'+mm+'-'+dd;
this.setProperty("ship_date", today);
return this;
```

.

2.

```
var multiple =
    document.thisItem.getProperty("multiple_shipment", "0");
if (multiple==1) {
    window.handleItemChange("shipping_method", "UPS");
    return;
}
window.handleItemChange("shipping_method", "Federal Express");
```

3.

```
var Filter = new Object();
Filter["classification"]=
    {filterValue:"Product", isFilterFixed:true};
return Filter;
```

Unit 12 Solutions

1. **onBeforeAdd**

```
string cxn = @"Provider=Microsoft.ACE.OLEDB.12.0;
    data source=C:\AccessDB\Training.accdb";

//Connect to Access Database
OleDbConnection conn = new OleDbConnection(cxn);
conn.Open();

string soid = this.getID();
string ponumber = this.getProperty("po_number", "");
string shipper = this.getProperty("shipping_method", "");

string insertsq1 = @"insert into PurchaseOrder (SalesOrderID,
Shipper, PONumber) values ('{0}', '{1}', '{2}')";

insertsq1 = String.Format(insertsq1, soid, shipper, ponumber);

OleDbCommand cmdUpdate = new OleDbCommand(insertsq1, conn);
int rowsAffected = cmdUpdate.ExecuteNonQuery();

conn.Close();
```

```
Dim cxn As String = "Provider=Microsoft.ACE.OLEDB.12.0;
data source=C:\AccessDB\Training.accdb"

'Connect to Access Database
Dim conn As New OleDbConnection(cxn)
conn.Open()

Dim soid As String = Me.getID()
Dim ponumber As String = Me.getProperty("po_number", "")
Dim shipper As String = Me.getProperty("shipping_method", "")

Dim insertsql As String = "insert into PurchaseOrder (SalesOrderID,
Shipper, PONumber) values ('{0}', '{1}', '{2}')"

insertsql = [String].Format(insertsql, soid, shipper, ponumber)

Dim cmdUpdate As New OleDbCommand(insertsql, conn)
Dim rowsAffected As Integer = cmdUpdate.ExecuteNonQuery()

conn.Close()
```

2. onBeforeUpdate

```

string cxn = @"Provider=Microsoft.ACE.OLEDB.12.0;
    data source=C:\AccessDB\Training.accdb";

//Connect to Access Database
OleDbConnection conn = new OleDbConnection(cxn);
conn.Open();

string soid = this.getID();
string ponumber = this.getProperty("po_number", "");
string shipper = this.getProperty("shipping_method", "");

string updatesql ="update PurchaseOrder set Shipper='{0}',
PONumber='{1}' where SalesOrderID='{2}'";

updatesql = String.Format(updatesql, shipper, ponumber, soid);

OleDbCommand cmdUpdate = new OleDbCommand(updatesql, conn);
int rowsAffected = cmdUpdate.ExecuteNonQuery();

conn.Close();

```

```

Dim cxn As String = "Provider=Microsoft.ACE.OLEDB.12.0;
data source=C:\AccessDB\Training.accdb"

'Connect to Access Database
Dim conn As New OleDbConnection(cxn)
conn.Open()

Dim soid As String = Me.getID()
Dim ponumber As String = Me.getProperty("po_number", "")
Dim shipper As String = Me.getProperty("shipping_method", "")

Dim updatesql As String = "update PurchaseOrder set Shipper='{0}',
PONumber='{1}' where SalesOrderID='{2}'"

updatesql = [String].Format(updatesql, shipper, ponumber, soid)

Dim cmdUpdate As New OleDbCommand(updatesql, conn)
Dim rowsAffected As Integer = cmdUpdate.ExecuteNonQuery()

conn.Close()

```

3. onAfterGet

```

string cxn = @"Provider=Microsoft.ACE.OLEDB.12.0;
    data source=C:\AccessDB\Training.accdb";

//Connect to Access Database
OleDbConnection conn = new OleDbConnection(cxn);
conn.Open();

for (int i = 0; i < this.getItemCount(); i++) {

    Item idx_itm = this.getItemByIndex(i);
    string soid = idx_itm.getID();

    string sql =
        @"select * from PurchaseOrder where SalesOrderID = '" + soid +
        "'';

    OleDbCommand cmd = new OleDbCommand(sql, conn);
    OleDbDataReader reader = cmd.ExecuteReader();
    if (reader.Read()) {
        idx_itm.setProperty("po_number", reader["PONumber"].ToString()
    );
        idx_itm.setProperty("shipping_method",
        reader["Shipper"].ToString() );
        }
    }
    conn.Close();
    return this;

Dim cxn As String = "Provider=Microsoft.ACE.OLEDB.12.0;
data source=C:\AccessDB\Training.accdb"

'Connect to Access Database
Dim conn As New OleDbConnection(cxn) conn.Open()

For i As Integer = 0 To Me.getItemCount() - 1

    Dim idx_itm As Item = Me.getItemByIndex(i)
    Dim soid As String = idx_itm.getID()

    Dim sql As String =
        "select * from PurchaseOrder where SalesOrderID = '" & soid & "'"

    Dim cmd As New OleDbCommand(sql, conn)
    Dim reader As OleDbDataReader = cmd.ExecuteReader()
    If reader.Read() Then
        idx_itm.setProperty("po_number", reader("PONumber").ToString())
        idx_itm.setProperty("shipping_method", reader("Shipper").ToString())
    End If
Next
conn.Close()
Return Me

```

Unit 13 Solutions

Solution for Aras_Integration:

Edit Sales Order:

```
protected void Button2_Click(object sender, EventArgs e)
{
    //MODIFY SALES ORDER

    string so_number = TextBox1.Text;
    Item so = Global.inn.newItem("Sales Order", "edit");
    so.setAttribute("where", "[sales_order].so_number='"
        + so_number + "'");
    so.setProperty("so_number", TextBox3.Text);
    so.setProperty("shipping_method", DropDownList1.Text );
    so = so.apply();

    if (so.isError())
    {
        StatusMessage.Text = so.getErrorString();
        return;
    }
    StatusMessage.Text = "Changes Saved";
}
```

```
Protected Sub Button2_Click(sender As Object, e As EventArgs)
    'MODIFY SALES ORDER

    Dim so_number As String = TextBox1.Text
    Dim so As Item = [Global_asax].inn.newItem("Sales Order",_
    "edit")
    so.setAttribute("where", "[sales_order].so_number='"
        & so_number & "'")
    so.setProperty("so_number", TextBox3.Text)
    so.setProperty("shipping_method", DropDownList1.Text)
    so = so.apply()

    If so.isError() Then
        StatusMessage.Text = so.getErrorString()
        Return
    End If
    StatusMessage.Text = "Changes Saved"
End Sub
```

Delete Sales Order:

```
protected void Button3_Click(object sender, EventArgs e)
{
    //Delete SO
    string so_number = TextBox1.Text;
    Item so = Global.inn.newItem("Sales Order", "delete");
    so.setAttribute("where", "[sales_order].so_number='"
                    + so_number + "'");
    so = so.apply();

    if (so.isError())
    {
        StatusMessage.Text = so.getErrorString();
        return;
    }
    StatusMessage.Text = "Order Removed";
}
```

```
Protected Sub Button3_Click(sender As Object, e As EventArgs)
    'Delete SO
    Dim so_number As String = TextBox1.Text
    Dim so As Item =
        [Global_asax].inn.newItem("Sales Order", "delete")
    so.setAttribute("where", "[sales_order].so_number='"
                    & so_number & "'")
    so = so.apply()

    If so.isError() Then
        StatusMessage.Text = po.getErrorString()
        Return
    End If
    StatusMessage.Text = "Order Removed"
End Sub
```

Unit 14 Solutions

```
Sub SalesOrders()

    'Connect to Server
    URL = "http://localhost/Innovator100"
    Db = "DevelopingSolutions100"
    user = "admin": pw = "innovator"

    Dim f As New IomFactory
    Dim inn As Innovator
    Set innov = f.CreateInnovator(Nothing)
    Dim conn As HttpServerConnection
    Set conn = f.CreateHttpServerConnection(URL, Db, user, pw)
    Dim result As Item
    Set result = conn.Login
    If result.IsError Then
        MsgBox (result.getErrorString)
    End If
    Set innov = f.CreateInnovator(conn)

    'Get Sales Order List
    Dim so As Item
    Set so = innov.newItem("Sales Order", "get")
    so.setAttribute "order_by", "so_number ASC"
    Set so = so.Apply

    Count = so.getItemCount

    Dim indexItm As Item

    For i = 0 To Count - 1
        Set indexItm = so.getItemByIndex(i)
        Cells(i + 1, 1).Value = indexItm.getProperty("so_number", "")
        Cells(i + 1, 2).Value = indexItm.getProperty("po_number", "")
        Cells(i + 1, 3).Value = indexItm.getProperty("shipping_method",
        "")
    Next i

    conn.Logout

End Sub
```

Unit 15 Solutions

1. See example of Users template in the unit examples. You can use the Wizard to step through the columns to create the template. You can save the template file and use it below to use the command line option.
2. BatchLoaderCMD -d [path to source csv file] -c [path to config file] -t [path to template file]

Sample config file shown in unit examples.

Unit 16 Solutions

1. Service Method to call:

```
string msg = "Service method executed";
CCO.Utilities.WriteDebug("ServerDebug", ref msg);
return this;
```

2. Config file settings:

```
<?xml version="1.0" encoding="utf-8" ?>
<innovators>
    <innovator>
        <server>http://localhost/Innovator100</server>
        <database>DevelopingSolutions100</database>
        <username>admin</username>
        <password>innovator</password>
        <http_timeout_seconds>600</http_timeout_seconds>
        <job>
            <method>Write Debug Log</method>
            <months>*</months>
            <days>*</days>
            <hours>*</hours>
            <minutes>*</minutes>
        </job>
    </innovator>
    <eventLoggingLevel>2</eventLoggingLevel>
    <intervalMinutes>2</intervalMinutes>
</innovators>
```

3. Start Service from Windows Services under Administration.

This page intentionally left blank.



Index

A

actions	
argument passing.....	110
context item.....	105
context Item.....	106
creating	102
defined	100
dialog box, custom	315
generic	109
Item.....	104
addRelationship method	143
AML	
actions.....	46
Add.....	48
and/or tag	33
attributes, passing.....	126
condition property attribute.....	34
creating custom actions	121
date and time.....	35
edit action	50
item attributes	41
item data type properties	40
item property tags	31
item tag attributes (primary)	30
order by attribute	43
related_id tag.....	37
relationships tag	36
reverse lookup	39, 348
saved search	44
select attribute.....	43
where attribute.....	43
API on-line help.....	65
appendItem method.....	146
apply method.....	72, 73, 132
applyAML method	116
applyAML method (client)	205
applyItem method (client)	205
applyMethod method.....	116, 122
applySQL method.....	116
Aras Object (Client).....	198
attachPhysicalFile	138

B

batchloader	
AML template	283

command line.....	289
example template	292
loading data.....	285
main screen	279
overview	278
relationship data	286
template wizard	282

C

cancelWorkflow action.....	154
CCO Utilities class methods	
WriteDebug	95
checkout method	138
clear caches	94
client events	
field.....	219
form.....	217
form and field.....	216
form event method	225
grid/row.....	376
ItemType	211, 213
OnSearchDialog	226
tabs.....	366
clone method	132
closeWorkflow action.....	154
COM Client	266
comparing IOM and AML	72
configurable grid	
creating.....	339
defining columns	350
grid events	342
overview	336
relationship grid event	351
select lists	353
xpath.....	345
context item	66
context item, on a form.....	220
createPropertyItem method	136

D

Debug option.....	92
debugging	
client.....	86
server.....	87
displaying prompt/confirm window.....	200

document.thisItem context	220
document.thisItem.getProperty method.....	222
document.thisItem.setProperty method.....	222
DOM object.....	221
drop list, dynamic from database	329
dynamic drop down list	330

E

Email method.....	153
-------------------	-----

F

federated ItemType	242
federated properties.....	236
federated property	243
federated property events.....	240
federation, using events	245
fetchDefaultPropertyValues method.....	136
fetchLockStatus method	153
fetchRelationships method.....	141
Form dialog box	313

G

generic methods	121
geNewId method	118
Get Controlled Item example.....	167
getAction method.....	135
getAttribute method.....	126, 135
getBaseUrl method	202
getDatabase method	203
getErrorCode method	152
getErrorDetail method.....	152
getErrorSource method	152
getErrorString method.....	152
getID method.....	135
getIdentityList method	203
getInnovator method.....	78
getItemByld method.....	118
getItemByIndex method	146
getItemByKeyedName method	118
getItemByXPath method	146
getItemCount method	146
getItemInDom method	118
getLoginName method	203
getNextSequence method	118
getProperty method	136
getPropertyAttribute method.....	136
getPropertyCondition method.....	136
getPropertyItem method.....	136
getRelatedItem method.....	145
getRelationships method.....	145
getServerBaseUrl method	202
GetTabId method.....	368
getType method	135
getUserid method.....	120
getUserId method	78, 203

getUserType method.....	203
GetUserVaultName example.....	123
getWorkingDir method	202
grid cell method	380
grid control	
accessing	373
callbacks	374
cell events.....	378
customizing	370
row events.....	375
grid/row event method.....	377

H

handleItemChange	222
hide/show tabs.....	368
HTML, custom controls	328

I

Innovator class methods	
applyAML.....	116
applyMethod.....	116, 122
applySQL.....	116
getInnovator.....	78
getItemByld.....	118
getItemByKeyedName.....	118
getItemInDom	118
getNewId	118
getNextSequence	118
getUserID.....	78, 120
newError.....	118
newItem	76, 77, 118
newResult.....	78, 79, 118
ScalcMD5.....	120
instantiateWorkflow method.....	153
isAdminUser method	203
isCollection method	134
isEmpty method	134
isError method	134
isLocked method	134
isLogical method	134
isNew method	134
isRoot method.....	134
Item class methods	
addRelationship.....	143
appendItem.....	146
apply	72, 73, 132
attachPhysicalFile	138
boolean methods (is...)	
.....	134
checkout	138
clone	132
collection methods	146
createPropertyItem	136
Email	153
extended methods	153
fetchDefaultPropertyValues	136
fetchLockStatus	153

fetchRelationships	141
getAction.....	135
getAttribute	126, 135
getErrorCode.....	152
getErrorDetail	152
getErrorSource.....	152
getErrorString	152
getID.....	135
getItemByIndex.....	146
getItemByXPath	146
getItemCount	146
getProperty.....	136
getPropertyAttribute	136
getPropertyAttribute method.....	123
getPropertyCondition	136
getPropertyItem.....	136
getRelatedItem	145
getRelationships.....	145
getType	135
instantiateWorkflow	153
loadAML.....	132
lockItem	153
logical methods (and/or)	149
newItem.....	72, 73, 151
newOr/newAnd/newNot	149
newXMLDocument	151
promote	153
relationship overview	140
removeAttribute	135
removeItem	146
removeLogical.....	149
removeProperty.....	136
removePropertyAttribute	136
setAction	135
setAttribute.....	73, 135
setErrorCode.....	152
setErrorDetail.....	152
setErrorSource	152
setErrorString	152
setId	135
setNewItem.....	135
setProperty	72, 73, 136
setPropertyAttribute	136
setPropertyCondition	136
setPropertyItem.....	136
setRelatedItem.....	143
setType.....	135
ToString.....	153
unlockItem	153
Item class properties	93
item factory	67
item types, supported.....	75

L

loadAML method	132
lockItem method	153
logging	95

logging server events	96
logging system events	361

M

ME keyword	66
method	
creating.....	68
editor	<i>Solution Studio Editor</i>
invoking	63
testing.....	71
method types	62
methodology, Aras	15
methods	
generic.....	121
Microsoft Excel example	268
Microsoft Office, integrating	266

N

newAnd/newOr/newNot method	149
newError method	80, 118, 152
newOMInnovator method	205
newItem	76
newItem method.....	72, 73, 118, 151
newResult method	78, 79, 118
newXMLDocument method	151

O

OnSearchDialog event	226
----------------------------	-----

P

parent.thisItem context	374
promote method	153

R

regasm.exe. assembly registration.....	267
removeAttribute method	135
removeItem method	146
removeLogical method.....	149
removeProperty method.....	136
removePropertyAttribute method	136
RequestState Class	
add.....	180
clear	182
contains	183
count	183
deletion use case	187
overview	179
related items use case	190
remove	182
retrieve	181
versioning use case	185
returning an Item	67

S

ScalcMD5 method.....	120
scheduler service	
configure.....	298
installing.....	297
monitoring	300
overview	296
search grid, custom result sets	306
search grid, prefilling	227
server events	
ItemType.....	161
LifeCycle.....	162
overview	160
System.....	360
tracking login and logout	359
validation example.....	164
Workflow	163
workflow definition.....	168
workflow validation	166
setAction method	135
setAttribute method.....	73, 135
setErrorCode method	152
setErrorDetail method	152
setErrorSource method	152
setErrorString method	152
setId method	135
setNewId method.....	135
setProperty method.....	72, 73, 136
setPropertyAttribute method	136
setPropertyCondition method	136
setPropertyItem method	136
setRelatedItem method.....	143
SetTabVisible method.....	368
setType method.....	135
showModalDialog	319
SOAP Message	261
SOAP Message, sending.....	261
Solution Studio Editor	69
startWorkflow action.....	154
system architecture	12
system events	
logging.....	361

T

tab control, accessing.....	368
tabs, working with.....	366
this keyword.....	66
toolbar, Solution Studio	70
top.aras	
AlertSuccess/AlertError	199
Confirm/Prompt	200
evalMethod	201
evalMethodItem.....	201
get paths.....	202
get URLs	202
get user info	203
get working directory	202
utility functions.....	205
ValidateVote.....	204
top.aras.Object.....	198
ToString method.....	153

U

unlockItem method	153
-------------------------	-----

V

Visual Studio	
adding IOM reference	255
creating a connection	256
Visual Studio and IOM	254

W

Workflow Actions	
cancelWorkflow.....	154
closeWorkflow.....	154
startWorkflow	154
workflow definition items	168
workflow dynamic assignments	172
workflow dynamic routing	170
workflow validation.....	166
WriteDebug	95