

## Applies to:

SAP MII 12.2

## Summary

The following document highlights the newer capabilities of the MII 12.2 release and provides a link to example content which demonstrate how each of these features works. The document walks you through the various examples contained in the “Examples 12.2” project in order to explain the purpose and operation of the content, so it can be used to improve your MII project implementations. It is targeted towards experienced MII content developers and assumes you are familiar with various SAP terms and MII content development.

**Author(s):** Salvatore Castro

**Web Site:** <http://www.scastro.net/sam>

**Company:** SAP Labs, LLC

**Created on:** August 9, 2010

## Author Bio



Salvatore Castro of SAP Labs has a Bachelors Degree in Computer Engineering and a Masters Degree in Computer Science both through the Rochester Institute of Technology. He came aboard SAP as part of the Lighthammer acquisition in 2005 and is currently a member of the MII Solution Management group.

## Table of Contents

Introduction.....	4
Project Structure .....	4
Configuration.....	4
Manufacturing Data Objects (MDO).....	5
OnDemand.....	6
Reporting Interface .....	6
Persistent .....	8
Asynchronous Reporting.....	8
Synchronous Reporting.....	10
Communication Buffer.....	12
Miscellaneous .....	15
Content Indexing .....	15
Content Usage .....	15
Transaction Performance Statistics.....	15
Custom Actions .....	16
Data Buffering .....	16
Enterprise Service Repository (ESR) .....	16
Expression Functions & Types.....	16
Grid.....	17
JCo Timeout.....	17
Limit Chart.....	17
Localization .....	18
Logic Editor .....	18
Web Pages.....	18
XSLT & Navigation.....	19
PCo Query .....	20
Peer to Peer .....	20
Asynchronous via Messaging Services .....	20
Direct Call via Runner Servlet.....	21
Query Caching .....	22
SPC Chart.....	22
Step-Through Debug .....	23
Template Security.....	23
Time zone calculations .....	24
Try/Throw/Catch/Finally .....	24
Value Mapping .....	25
XML Document.....	25
Value Lookup.....	26
Appendix A .....	27
Miscellaneous Custom Action .....	27
InterfaceTestActions.java .....	27
ConnectionAliasDialog.java.....	29

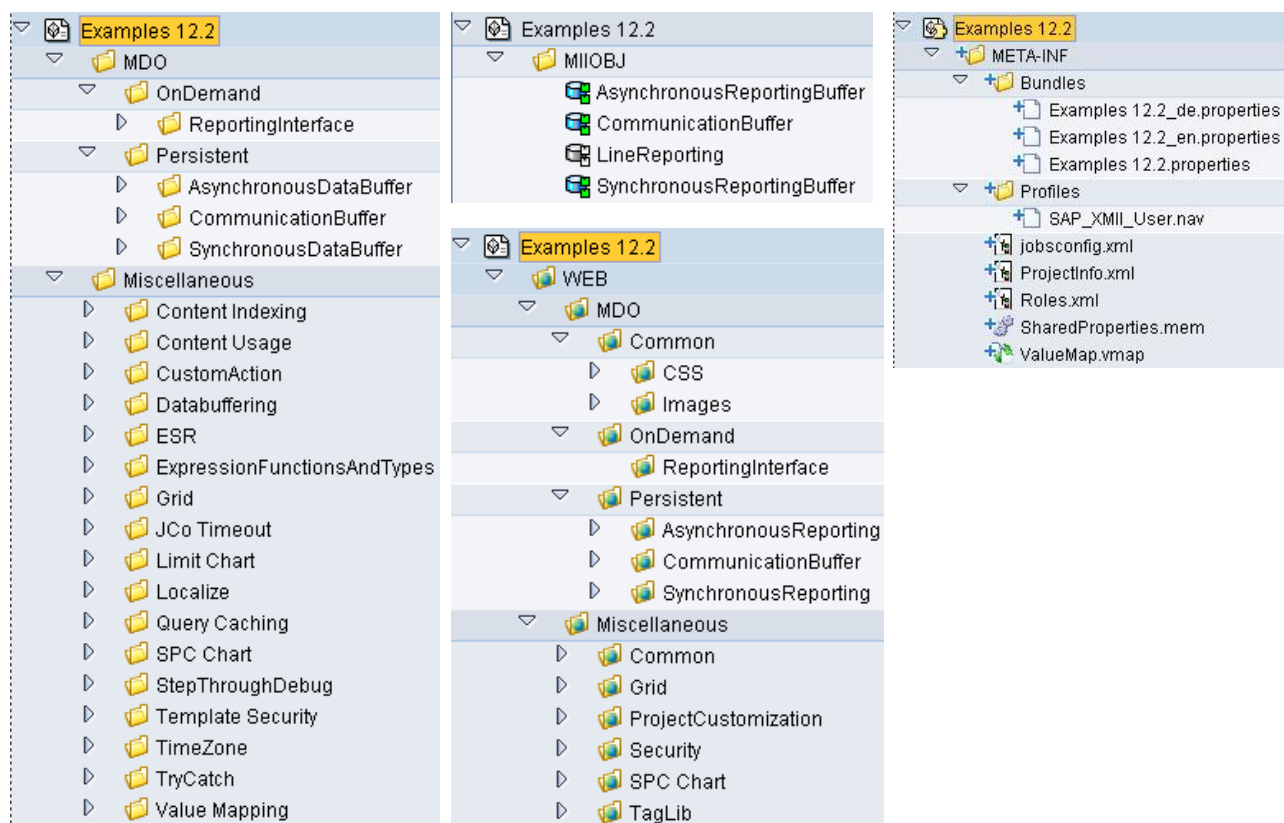
CredentialAliasDialog.java.....	31
QueryDialog.java .....	32
Catalog.xml .....	34
Value Mapping .....	34
Maintenance Rule.....	34
LineTags Rule .....	34
Related Content.....	36
Copyright .....	37

## Introduction

The following document contains both high level and detailed technical information about the newly added features for the MII 12.2 product. This document assumes that you are an experienced MII content developer and are familiar with the various terms and content development techniques. The product capabilities are not limited to what is outlined in this document as it provides only an overview of the newly added features of MII version 12.2, including content scenarios for Manufacturing Data Objects (MDO). The project content is also available on the SDN as a separate download that can be imported into your MII 12.2 environment. The project can then be used as a working reference or as a starting point for utilizing the new feature of MII 12.2. See the [Related Content](#) section for links to additional online content.

## Project Structure

The project associated with this document makes use of a wide range of both existing and new product features. The structure of the project is setup into two primary sections, MDO and Miscellaneous. The topics under each of the categories are broken out into individual folders. The project structure from the Catalog, Object, Web, and Meta-Inf tabs in the workbench is as follows:



The primary starting point for the examples should be the web & content tabs as this will highlight the outlined features best.

## Configuration

The following section contains generic configuration steps that must be completed in order for some of the examples in the referenced project to execute. First import the Examples 12.2 project next access the Workbench -> Meta-Inf -> Examples 12.2 -> SharedProperties.mem. Open the shared properties editor and create two new Persistent String properties without a Namespace called **CredentialAlias** and **ConnectionAlias**. The values of each of properties should correspond with valid Credential and Connection store aliases defined for your MII environment for accessing your ERP environment and are used in the Miscellaneous sections ESR and JCo Timeout examples.

## Manufacturing Data Objects (MDO)

The following section contains some examples of how the SAP MII MDO feature can be utilized in both OnDemand and Persistent modes. If you're not sure about the difference between these two modes please be sure to read the MDO help documentation available on <http://help.sap.com> before proceeding any further in this document. In short, an OnDemand object retrieves its data upon request from the specified query template or transaction and does not stage or persist any data. An OnDemand object is merely a pointer to a run time query. The MDO representation of the run time query provides a friendlier namespace to the UI developer, but the result is still a query result set. A typical use case would be to create a UI front-end whose data can be "wired-in" later. In this scenario, the on demand object acts as an abstraction layer between the UI and the underlying data source.

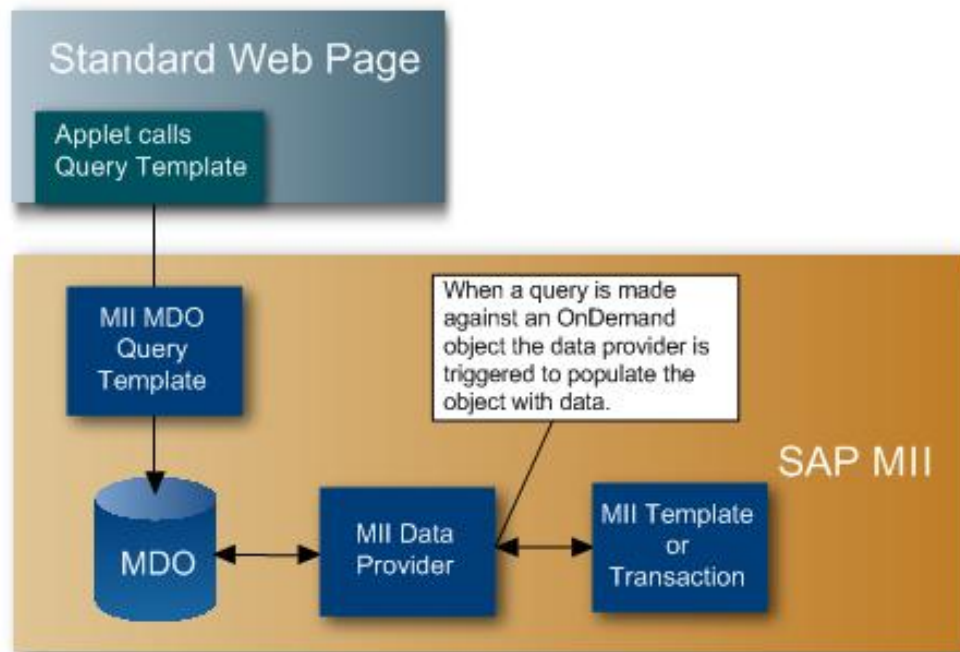
The Persistent mode was designed to stage data in the predefined MDO datasource for local reporting or for asynchronous transactional integration with other systems. The content stored in these objects can be queried and then reported from and is ideal for local survivability scenarios. In many cases, where custom database tables were previously required for the development of a composite application, you can now use the modeling and persistence layer available in MDO. This interface can be beneficial over custom database tables since the objects are database independent and transport with the project content. The lifecycle of the objects can be configured and managed through the application content. It is important to note that you can impact the performance of your objects via the following implementation practices relative to the usage requirements and hardware available.

- The MDO Datasource database is used by multiple MII instances (Not Supported)
- The Datasource target is on a database that is across the network which can increase query and command response times due to network latency and data round trip times.
- OnDemand & Persistent objects are required to store and/or process a large number of records; for example if there are 50,000 rows or more in an object then you may experience long running queries (The upper limit varies from system to system and from database to database).

The most common usage scenarios are addressed in the following examples in order to help familiarize you with the implementation of MDO to support your MII applications.

## OnDemand

As previously mentioned the OnDemand mode for MDO means that the data is retrieved from the defined Data Provider when a query request is issued to the object. This data can be all of the object data or a filtered subset of the data provider content depending on your reporting needs. The high level architecture of this type of object looks like this:



## Reporting Interface

The purpose of this scenario is to demonstrate the capabilities of an OnDemand object for flexibly reporting data to a pre-defined UI. An example is when you design a standard UI or report that will be used at several plants. However, each plant has different underlying data sources and means of retrieving data; MDO can be used as the abstraction layer to fit this use case. A common use case arises when deploying content from a central location to a distributed location where the structure of the systems & data is unknown. This object can be used to define the interface where sample data from Simulator or other test databases can be used to develop against. The data provider for the object can then be defined at the local site in order to "Wire it into your plant", and populate data into the predefined report. This approach will cut down on the content development and cross-site training time since the same user interface will exist at each of the distributed sites. Development using this interface fits into the use of NWDI and deployment of content since the DataProvider is simply a reference it can point to a transaction outside of the deployed project content that the local plant can define to fit their needs. See the [Expression Functions & Types](#) section for additional possibilities and use cases for deployed content.

The example provided demonstrates how to generate this type of generic report fir by creating the object defined in the Object tab here:

Examples 12.2/MIIOBJ/LineReporting

The first point to observe is that there is a data provider specified that points to the following Content:

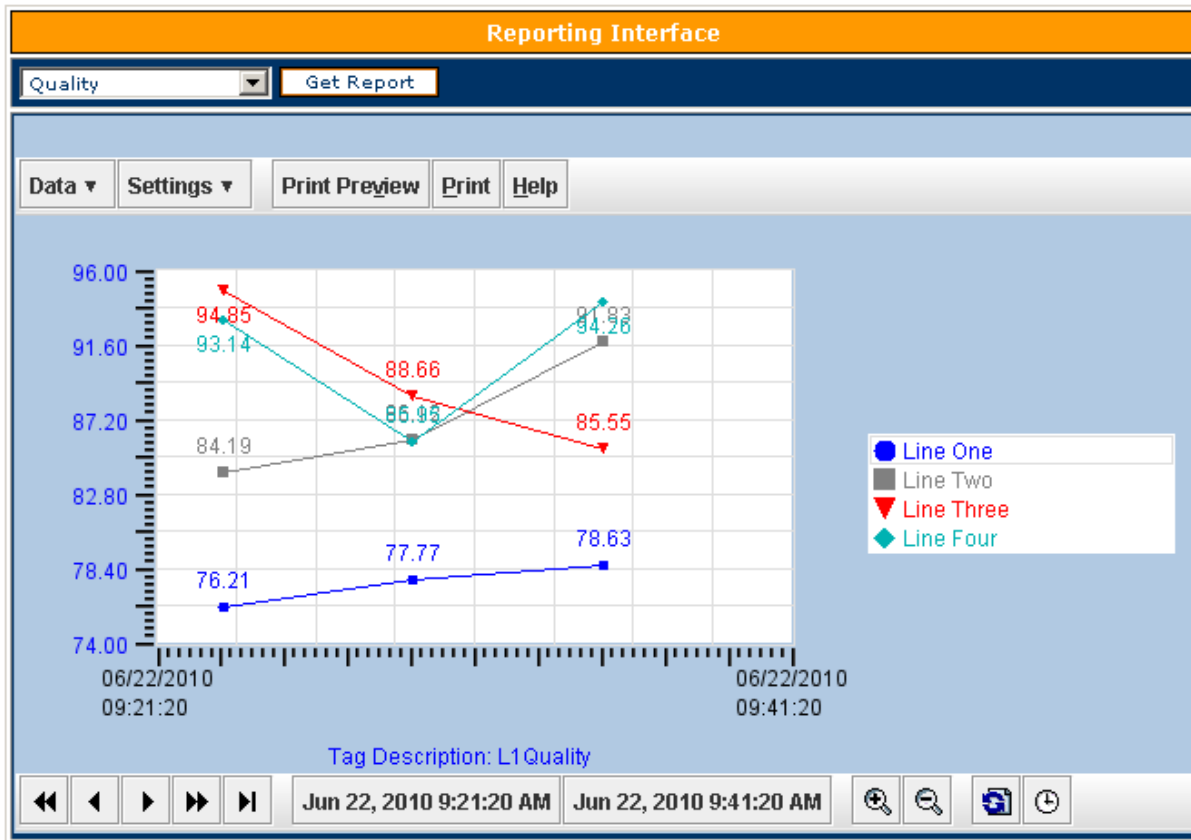
Examples 12.2/MDO/OnDemand/ReportingInterface/LineReporting

This transaction will return different Illuminator Document XML columns based on the type of report the user wishes to see. This is done in the transaction by modifying the query template path at runtime and changing the structure of the XML data returned to the object. The object definition in the Attributes Template Category has five columns defined and it is set for position mapping. Position mapping will overlay the column names defined in the attributes over the top of the columns and data returned by the data provider. This ensures that mappings for the UI objects will remain consistent even if the data returned from the data

provider is different. There is a sample web report that is delivered as the front-end for the object is located here on the Web tab:

Examples 12.2/WEB/MDO/OnDemand/ReportingInterface/OnDemandReportingInterface.html

When you test the page it should look similar to this:



There are a couple of things to notice about this web page in order to fully understand the benefit of the OnDemand objects. The first is that the description for the tags is not modified and therefore still appears in the response so that it is clear what data is being presented to the user. In this case the description is set as the tag name since this is the default behavior for the tag query. Also, there is only one display template and one web page but there are three different tag queries and one object query involved in building this report. The object query is what is used by the iChart applet on the page to retrieve the data through the OnDemand object interface which presents the data from one of the three tag queries. This flow of data is outlined in the architecture diagram shown in the previous section and should help to conceptualize how this type of abstracted interface can be used to roll-out standard reports to various different locations.

## Persistent

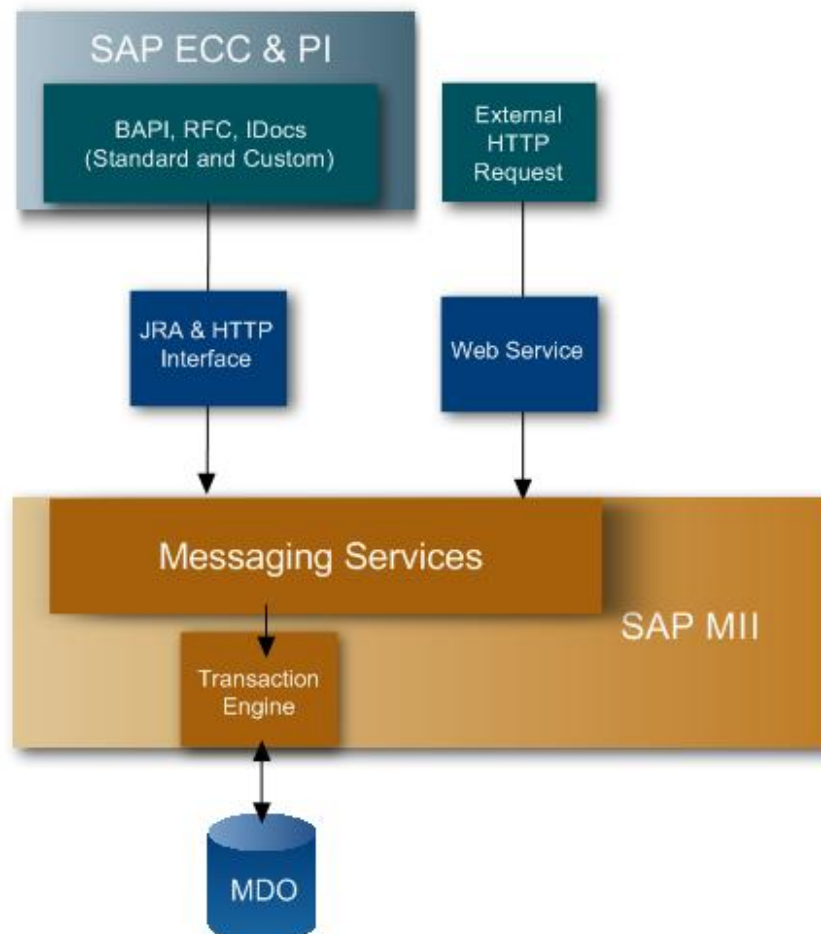
Persistent MDO definitions allow data to be staged in a local repository for use in limited, local survivability and local reporting scenarios. This allows data that exists in another system to be kept locally for quick and reliable reporting without having to query the remote system each time. Persistent objects are also a good choice when you create data (such as a metric) and want to store it, although it is not recommended for long term historical values, since the MDO object will not be as efficient as a well indexed and tuned database.

Another good use case is when you need to collect plant data, merge it with other data that has business or ERP context, and you will write it to ERP on a timed or event basis. An example of this use case is taking data from an incoming IDOC and storing it in an MDO object (such as product order number). You may then collect material consumption data and wish to do goods movement or material consumption against that order. You can collect the data into the MDO object, and write back to ERP when the operation is complete.

Data can be populated into this type of object through two different mechanisms depending on how you intend to use it. The object can represent the result of a query (Replace) where the contents of the object are completely replaced when the replace task is run. This object behaves similar to the query cache feature but with a higher level of control over the data. If a Data Provider is specified for the object then its response will define the contents of the object. The object can behave similar to a database table (Insert) by defining only custom attributes for the object and using MDO Insert commands to write data into the object and no defining a Data Provider. Both of these MDO design approaches are illustrated in this example project.

## Asynchronous Reporting

The purpose of this scenario is to outline one example on how the Asynchronous Messaging Services interface can be used to populate data into an object for end-user reporting. The example demonstrates the use of an object like a database table. This is useful when data is sent asynchronously to MII from another system and needs to be displayed to an end user. The architecture diagram for the way data is loaded into the MDO Object for this scenario is as follows:





## Configuration

The following section explains how an asynchronous message from an external system, such as an IDOC from ERP, can be captured and its contents can be loaded into an MDO object for user reporting as shown in this example. The detailed steps to achieve the Asynchronous Messaging scenario a Message Processing Rule needs to be configured as shown below:

**Details for Examples12\_2**

Settings	Processing												
Name: Examples12_2	Transaction: Examples 12.2/MDO/Persistent/AsynchronousDataBuffer/LoadLOIP...												
Description: Processing Async Message to Object	Persist Transaction: On Error												
Message Listener: XMIMESSEAGELISTENER	Log Level: Info												
Message Type: Web Service	Parameters												
Message Name: LOIPRO	<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/> LOIPRO</td> <td>ReceivedMessageXML</td> </tr> <tr><td> </td><td> </td></tr> <tr><td> </td><td> </td></tr> <tr><td> </td><td> </td></tr> <tr><td> </td><td> </td></tr> </tbody> </table>	Name	Value	<input checked="" type="checkbox"/> LOIPRO	ReceivedMessageXML								
Name	Value												
<input checked="" type="checkbox"/> LOIPRO	ReceivedMessageXML												
Processing Type: transaction													

The transaction set for processing this message is located here:

Examples 12.2/MDO/Persistent/AsynchronousDataBuffer/LoadLOIPROIDocIntoAsyncObject

Once this processing rule is configured and enabled it's a good idea to also define a Message Cleanup rule that will remove messages older than N hours from the MII Message Services history. An example might look something like this:

**Details for XMIMESSEAGELISTENER\_DELETE**

Name:	XMIMESSEAGELISTENER_DELETE
Description:	Remove all messages older than 2 days
Message Listener:	XMIMESSEAGELISTENER
Message Type:	Web Service
Message Name:	*
Messages Older Than:	48 Hours
Processing Status:	Any
Enabled:	<input checked="" type="checkbox"/>

Once the previously mentioned steps are complete you can test this operation by running the following transaction from the Workbench:

Examples 12.2/MDO/Persistent/AsynchronousDataBuffer/Usage/SimulateIDocPushFromERP

Once the simulation transaction runs you'll notice the following entry in the Message Monitor:

SAP MII: Message Monitor						
<b>Messages</b>						
Listener	All	Delete	Reprocess			
Find		With Status	Any	From	Today	Go
Advanced						
Received	Status	Name	Type	Server	Category	Processed
<div> </div>						
April 27, 2010 7:38:12 PM CEST	<a href="#">Success</a>	LOIPRO	Web S...	XMIMESSEAGELISTENER		April 27, 2010 7:49:52 PM CEST

The Message Monitor image above shows that the message was successfully received and sent to the processing transaction and that processing transaction finished successfully.

Next navigate to the following Web page for an example on how to view the contents of the defined MDO:

Examples 12.2/WEB/MDO/Persistent/AsynchronousReporting/AsynchronousReporting.html

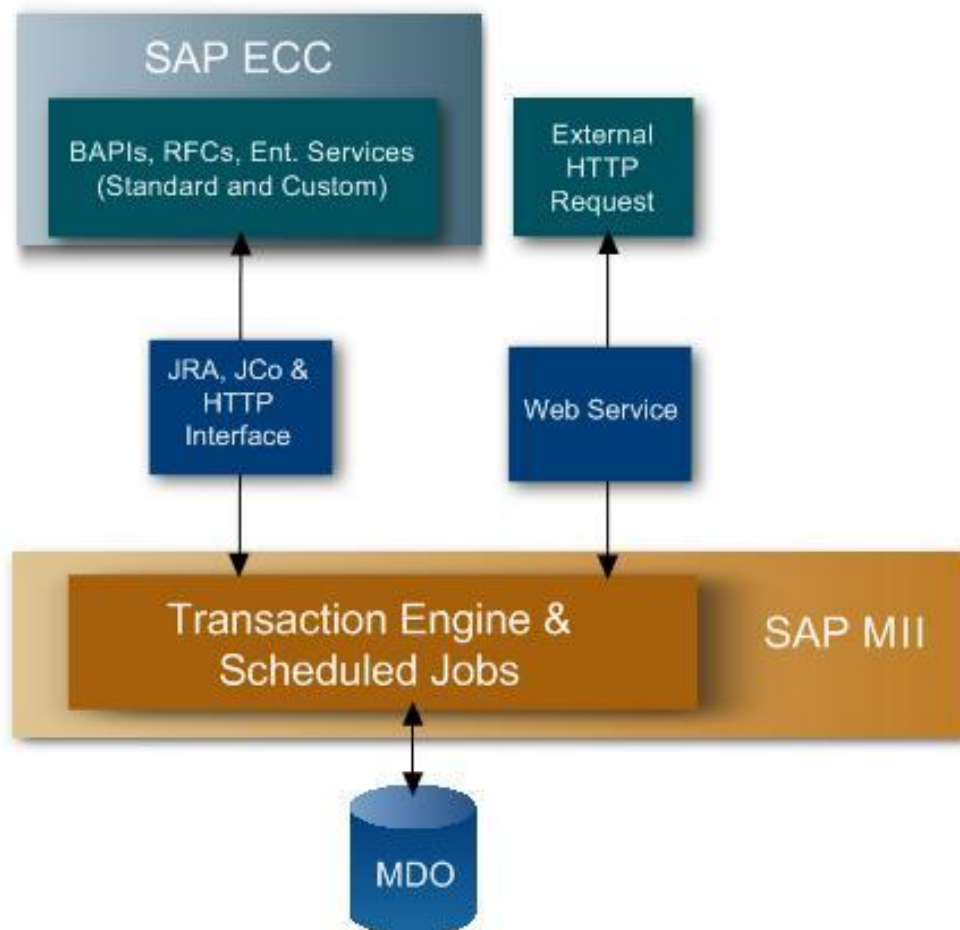
This page will look similar to this and it displays the contents of the Persistent MDO object:

Asynchronous Reporting								
Last Update: Apr 27, 2010 10:11:10 PM								
ProductionOrder	OrderType	Material	Plant	BaseQuantity	BaseUOM	TotalOrderQuantity	ScheduledFinish	BillOfMaterial
70001176	PI01	Y-300	1100	0.000	KGM	10000.000	Apr 12, 2006 12:19:28 PM	00000122

The above web page is simply a MDO query populating an iGrid and the object contains production order information based on LOIPRO IDocs received by the Messaging Services interface. The object can have a lifecycle task configured in order to ensure that old production order information is removed so only current/relevant data is displayed to the end user. There is no Delete task configured in this example as the record would be removed since its scheduled finish date is for April of 2006.

### Synchronous Reporting

The purpose of this scenario is to outline how a scheduled lifecycle transaction can be used to synchronously query data and populate it into a local object for reporting. This is similar to a query cache except that it supports caching of time based queries and allows the user to query for subsets of the data in the object using SQL like syntax. The architecture diagram for the way data is loaded into the MDO Object for this scenario is as follows:



## Configuration

Before the object will populate with any data the lifecycle task that is defined for it must first be deployed and enabled. The object needs to be opened from the workbench and can be found here:

Examples 12.2/MIIOBJ/SynchronousReportingBuffer

To view configuration, follow these steps from the MDO Template:

- Select the Lifecycle category
  - View the task named “Replace Data - Synchronous Reporting”
  - Press the “Deploy All” button to load the task into the SAP MII Scheduler
  - Press Run in order to execute the task and populate it with initial data.
- From the MII Menu, navigate to the System Management -> Scheduler screen
  - Enable the task.

The task in the scheduler screen should look similar to this:

The screenshot shows the 'Details for CachedDataQueryTest' window in the SAP MII Scheduler. The 'MDO Scheduler' tab is active. The configuration details are as follows:

- Name:** Replace Data - Synchron...
- Description:** Perform a wholesale update of
- Enabled:** ☒
- MDO:** Examples 12.2/MIIOBJ/SynchronousReportingE
- User Name:** I808627
- Password:** [Masked]
- Pattern:** 00 00 00 \*\* 2-6 (Midnight between Monday and Friday)
- Mode:** Replace

On the right, the 'Inputs' tab is active, showing a table of input values:

Key	Value
Plant	1000
MATLOW	A
MATHIGH	Z
MaxRows	50
Debug	0

Buttons for 'Add' and 'Delete' are located below the input table.

The default value for the CRON pattern, <http://en.wikipedia.org/wiki/Cron>, indicates the following schedule “Midnight between Monday and Friday” for the task to run but can be adjusted in order to fit your business needs.

You can now navigate to the following page for an example on how to view the contents of the defined MDO:

Examples 12.2/WEB/MDO/Persistent/SynchronousReporting/SynchronousReporting.html

The page will look similar to this:

Synchronous Reporting							
Last Update: May 3, 2010 7:15:21 PM							
Plant	Material	Description	Version	Type	Group	BaseUOM	
1000	A-100	Finished good		FERT	00101	EA	
1000	A-110	semi finished material		HALB	00101	EA	
1000	A-111	raw material		BOH	00101	EA	

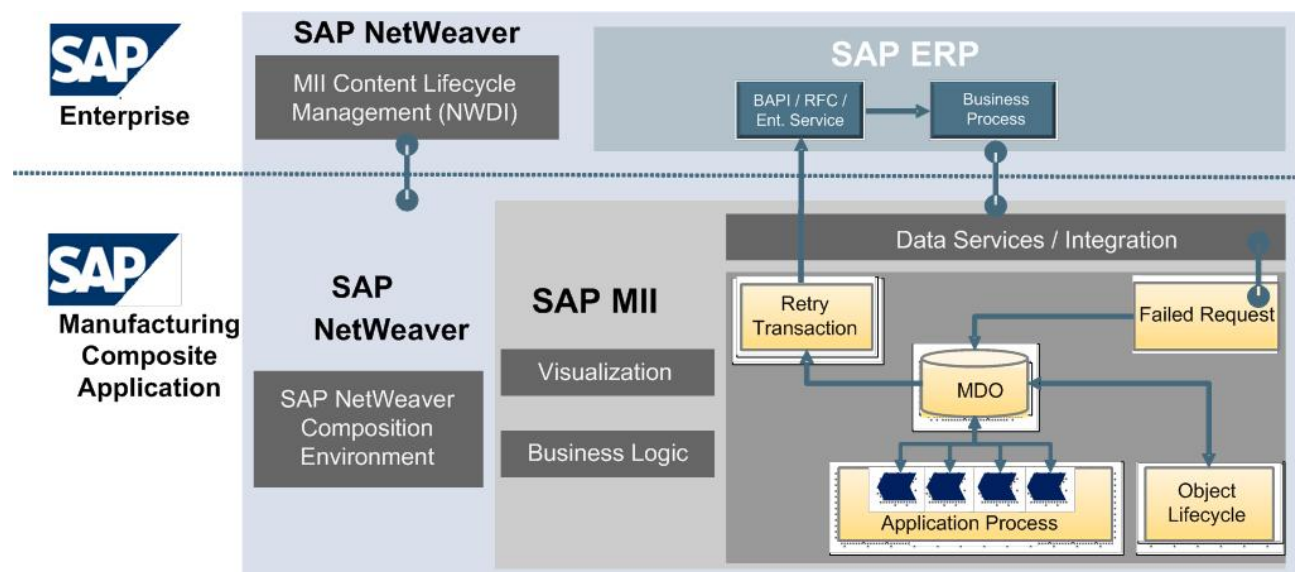
It displays the results of the Material Get List and Details transaction which combines the results from two different BAPIs and then via a “Replace” task refreshes the data in the object.

## Communication Buffer

The SAP MII Data-buffering capabilities can be used to ensure data integrity between plant and corporate systems while ensuring a stable and reliable end-user experience. The presence of MDO provides the ability to take buffering one step further and this section explains how to do this. The purpose of this scenario is to demonstrate how MDO can be used as a communication data buffer and how it can be configured to process messages according to your business requirements. There are a variety of ways to handle these messages and this example is designed to handle the most popular ones. The features that this example incorporates are:

- Retry of the following XML request types: BAPI/RFC/Web Service
- Ability to edit a buffered message via a web page
- Ability to manually retry or delete a message via a web page
- Enable or Disable the in order processing of the messages (order is based on time the request failed, but this can be modified to work in any way that suites your scenario).
- Shows the retry count of how many times message delivery was attempted and the last error message as to why the attempt failed.
- Data buffering for loading failed messages into object for guaranteed delivery

The architecture diagram for the way data is loaded into the MDO Object for this scenario is as follows:



This diagram outlines the various systems that can be involved in the scenario where data integrity and system availability are key factors in your shop-floor application. It also outlines the asynchronous nature of the buffering capabilities and the ability to plug this component into your existing application.







## Configuration

Before the object is populated with any data, the lifecycle task that is defined for it must first be deployed and enabled in your environment. The object needs to be opened from the MII Workbench and can be found here on the Object tab:

Examples 12.2/MIIOBJ/CommunicationBuffer

From the Lifecycle Template Category view the task named "CleanupBuffer" and Press the "Deploy All" button to load the task into the scheduler. Then navigate to the Menu -> System Management -> Scheduler screen and select and Enable the CleanupBuffer task.

There is also a separate task that must be defined in the Meta-Inf -> jobsconfig.xml editor that is similar to:

Name	ProcessCommunicationBufferEntrie		
Description	Process the entries in the Examples 12.2/MIIOBJ/CommunicationBuffer		
File	essCommunicationBufferEntries.trx		
Credential Alias	SAPJCO		
Pattern	0 **/4 ***		
Persist Transaction	ONERROR 		
Log Level	ERROR 		
<b>Properties</b>			
Name	Description	Type	Value
 InOrderProcessing	Indicate if communicaiton buffer entries should be sequentially processed	Input	false

The defined file that is scheduled to run is:

Examples 12.2/MDO/Persistent/CommunicationBuffer/ProcessCommunicationBufferEntries

This task defines how and at what interval each of the entries in the Communication Buffer object is processed. The default pattern is set for "On 0<sup>th</sup> Second at every Minute at every 4 Hours" this can be modified depending on your specific business needs.

In order to load an example failure entry into the Communication buffer and to see how to interact with the buffer open and run the following transaction:

Examples

12.2/MDO/Persistent/CommunicationBuffer/Usage/ExampleForInterfacingToMDOCommunicationBuffer

You can now navigate to the following page for an example on how to view the contents of the defined MDO:

Examples 12.2/WEB/MDO/Persistent/CommunicationBuffer/BufferManager.html

The page will look similar to this:

Communication Buffer Entries						
Last Update: May 4, 2010 9:25:56 PM						
CreatedOn	Name	Type	RetryCount	LastRetry	LastErrorMessage	Status
Apr 12, 2010 9:35:04 PM	BAPL_PRODCONF_CREATE_TT	BAPL	0	TimeUnavailable	---	NEW

View Entry
Delete Entry
Refresh

This page provides an administration interface for viewing, editing, and removing entries in the Communication Buffer object. If you select the entry and press the “View Entry” button you will see a page that looks similar to this:

Buffer Entry Details		
Name: BAPL_PRODCONF_CREATE_TT	Type: BAPL	Created On: 2010-04-12T21:35:04
Retry Count: 0	Status: NEW	Last Retry: TimeUnavailable
Last Error Message: ---		
Parameter1: <input type="text"/>		
Parameter2: <input type="text"/>		
<pre>&lt;?xml version="1.0" encoding="utf-8"?&gt; &lt;ProductionConfirmationCreateRequest_sync xmlns="http://sap.com/xi/SAPGlobal20/Global"&gt; &lt;MessageHeader xmlns=""&gt; &lt;ID schemeAgencyID="" schemeAgencySchemeAgencyID="" schemeID="" /&gt; &lt;UUID /&gt; &lt;ReferenceID schemeAgencyID="" schemeAgencySchemeAgencyID="" schemeID="" /&gt; &lt;ReferenceUUID /&gt; &lt;/MessageHeader&gt; &lt;ProductionConfirmation xmlns=""&gt; &lt;GroupID /&gt; &lt;ExternalSystemConfirmationID /&gt; &lt;ProductionOrderID&gt;000001004000&lt;/ProductionOrderID&gt; &lt;ProductionOrderSequenceID /&gt; &lt;ProductionOrderOperationID /&gt; &lt;PlantID schemeAgencyID=""&gt;0001&lt;/PlantID&gt; &lt;WorkCentreID schemeAgencyID="" /&gt; &lt;ExternalSystemUserAccountID schemeAgencyID="" schemeAgencySchemeAgencyID=""&gt;Sam&lt;/ExternalSystemUserAccountID&gt; &lt;RecordTypeCode&gt;R20&lt;/RecordTypeCode&gt; &lt;PostingDate&gt;2009-06- 28&lt;/PostingDate&gt; &lt;OpenReservationCompletedIndicator&gt;true&lt;/OpenReservationCompletedIndicator&gt; &lt;ConfirmQuantity unitCode="EA"&gt;1&lt;/ConfirmQuantity&gt; &lt;ScrapQuantity unitCode="EA"&gt;0&lt;/ScrapQuantity&gt; &lt;ReworkQuantity unitCode="EA"&gt;0&lt;/ReworkQuantity&gt; &lt;Description languageCode=""&gt;My Confirmation&lt;/Description&gt; &lt;GoodsMovementInformation&gt; &lt;MaterialInternalID schemeAgencyID="" schemeID=""&gt;SC- PRODUCT&lt;/MaterialInternalID&gt; &lt;PlantID schemeAgencyID=""&gt;0001&lt;/PlantID&gt; &lt;InventoryManagedLocationInternalID schemeAgencyID="" schemeID="" /&gt; &lt;BatchID /&gt; &lt;ConsignmentStockVendorID schemeAgencyID="" schemeID="" /&gt; &lt;ConsignmentStockCustomerID /&gt; &lt;SalesOrderID schemeAgencyID="" /&gt; &lt;SalesOrderItemID /&gt;</pre>		
Update Request		

This page allows the user to update the contents of the buffered XML Request and write it back into the object. This can come in handy if the reason for the message failure is because of a typo or other incorrect data content. You could also provide a more intuitive UI if you wished, by parsing the document in a BLS transaction and showing a more standard web form for a user to edit data. For simplicity, this was not done in this example.

## Miscellaneous

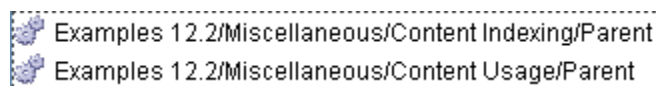
The miscellaneous content defined in this project outlines content development changes that have been implemented in order to reduce implementation overhead and improve the overall understanding of the SAP MII product within the implementation community. This section contains background information about newer features along with the setup required and where to find the corresponding example in the project.

### Content Indexing

Content indexing only works for content that is defined in the Catalog tab of the Workbench, and it does not apply to Web content. Indexing does not identify content referenced only by web content such as query and display templates that are used together on web pages. It does allow you to see what templates are used in other BLS transactions. To utilize Content Indexing right-click on content you wish to check and select the "Usage" option.

In the following project location right-click on the transaction named "Child":  
Examples 12.2/Miscellaneous/Content Indexing/Child

You should see the following window:



This indicates that the Child transaction is used in the Parent transaction of both the Content Indexing and Content Usage folders.

### Content Usage

This feature allows an MII user to view how frequently MII content is utilized along with basic statistical information about the performance of the object. To view this feature simply run the Parent transaction in the Content Usage folder and navigate to the Menu -> System Management -> Content Usage page and view the previously run content. You may want to filter the Project column with the following entry "Examples 12.2\*" and the grid could look like this:

Child.trx	Transaction	Examples 12.2/Miscellaneous/Content I...	1	52	52	52
GetSynchronousReportingDataQuery.t...	Query	Examples 12.2/MDO/Persistent/Synchr...	1	2,056	2,056	2,056

This is useful in identifying bottlenecks in the application due to long running transactions and it also helps to identify frequently used content which can then be targeted for optimization efforts.

### Transaction Performance Statistics

A very granular report on the execution times of the individual actions for a given transaction can be captured for detailed performance analysis information for a given execution of a transaction. This can be done by running a transaction via the Runner Servlet:

[http://help.sap.com/saphelp\\_mii122/helpdata/en/4a/287c3dd30242ade10000000a421138/content.htm](http://help.sap.com/saphelp_mii122/helpdata/en/4a/287c3dd30242ade10000000a421138/content.htm)

The parameter **LogStatisticsToDB** must be set to true as shown here in this example URL:

<http://<server>:<port>/XMII/Runner?Transaction=<Project>/<Path>/<Name>&LogStatisticsToDB=true>

The execution of this transaction then writes out the runtimes of each action and the links associated with each action, making it easier to identify potential bottlenecks in the transaction. It will also return the Transaction ID associated with the recorded values and is required to retrieve them. To retrieve the detailed performance information from the database using the following URL:

<http://<server>:<port>/XMII/Illuminator?service=BLSManager&Mode=Stats&ID=<TRXID>&content-type=text/csv>

This will open the results from the transaction execution into your CSV editor, typically Excel, where the data can be analyzed for performance bottlenecks. There is an example transaction that demonstrates how to call a transaction via the performance logging URL and how to retrieve the recorded performance here:

Examples 12.2/Miscellaneous/TransactionPerformanceStatistics/ExecuteAndStoreTransactionExecution



## Custom Actions

The Custom Action SDK has also been improved in this release to include access to the Credential and Connection stores. This allows the custom action to better integrate with the environment so that it can use previously defined credential and connection alias information. This section requires that the JAR in the following location in the Web tab:

Examples 12.2/WEB/Miscellaneous/CustomAction/SAPMIISDKTest.jar

Is exported from the Workbench and imported and deployed to MII from the Menu -> System Resources -> Custom Actions page. Once this is done the workbench must be restarted in order to properly load the custom action. Open the following transaction:

Examples 12.2/Miscellaneous/CustomAction/CustomActionTest

View the action dialogs for each of the defined actions in the SDK Test action category. For more details on how these actions were created see the source code in Appendix A -> Miscellaneous Custom Action section.

## Data Buffering

Communication buffering for various MII actions can be configured in the event of a communication failure between the MII and target system. An example of how this buffering works is located here:

Examples 12.2/Miscellaneous/Databuffering/DataBufferPOSTAction

Simply open this transaction and view the configuration for the POST action. This defines a configuration that points to a system that should never exist in your environment and as a result when the transaction runs an entry is made into the MII Communication Data Buffer. The entry generated by this can be viewed and administrated from the Menu -> Data Services -> Data Buffer page.

Also notice that if you select the entry the contents of the message are also shown.

## Enterprise Service Repository (ESR)

The enterprise service repository is a centrally defined system that holds WSDL URL references to various web service operations across the enterprise. There is a new action block available in the transaction environment that is specifically designed to browse this repository to simplify retrieval of various service URLs. This makes browsing of available services, and consuming them within an MII BLS transaction, a simple process.

The transaction demonstrates the usage of this action:

Examples 12.2/Miscellaneous/ESR/DemonstrateESRIntegration

Simply open up the configuration dialog for the ESR\_SimpleEchoBean action and press the back button to navigate back to the ESR search dialog. Select one of the various searches and provide the filter criteria and press the Search button. Select a service that fits your needs and press the Next button. The rest of the action configuration is the same as the generic web service action.

## Expression Functions & Types

The new release also has some additional miscellaneous additions to the expression function list in the link editor. The best way to understand some of the expression and data type controls is to simply open each of these transactions, view their operation and test their execution.



## Grid

There was a small change in the way that the SAP MII Grid applet behaves; this is the way that Boolean values can be displayed to the user. In order to view this simply open up the grid display template here:

Examples 12.2/Miscellaneous/Grid/BooleanGridTest

Then press the test button and you will see a page that looks similar to this:

Last Update: May 3, 2010 2:38:10 PM			
Data ▼	Settings ▼	Print Preview	Print Help
DateTime	Fan1	Fan2	
May 3, 2010 1:14:06 PM	<input type="checkbox"/>	0	
May 3, 2010 1:15:06 PM	<input checked="" type="checkbox"/>	1	
May 3, 2010 1:16:06 PM	<input type="checkbox"/>	0	

## JCo Timeout

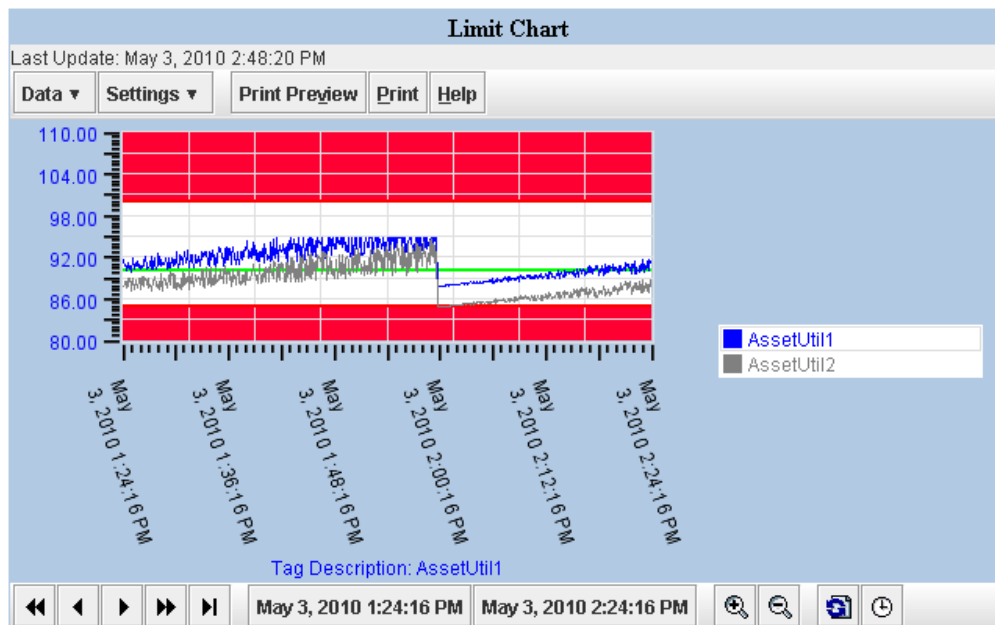
The SAP JCo Interface and SAP JCo Start Session action blocks now allow for more granular control over the cleanup interval of the various connections that are made to the target ERP system. The default value for this connection is also the maximum value of 10 minutes; which may be higher than what is desired to fit your application needs. In the configuration dialogs for these actions the "Timeouts" field can be used to specify the maximum time to keep an idle connection and its associated resources alive for.

## Limit Chart

There is a new iChart type called a Limit Chart which allows for regions to be displayed on a line chart in order to help identify good and bad regions for values. This works similar to the way that regions on the iSPCChart do but without the additional SPC calculation overhead and it also supports the display of multiple lines on the same chart. To test this feature simply open up the chart display template here:

Examples 12.2/Miscellaneous/Limit Chart/DemonstrateLimitChart

Then test the template, the page that appears should be similar to:



## Localization

Localization of messages and strings are defined in the workbench environment and now supports via the use of Java Properties files that are contained in the Meta-Inf -> Bundles for each project. The properties files behave like standard java properties files, , and require the entry of name value pairs and also utilize the Language code of the user to identify which file is to be used for the given language. The naming convention for the properties files follows the java ResourceBundle standard definition which is as follows:

<ProjectName>\_<LanguageCode>.properties

For example you can have the following properties files and values:

**Examples 12.2.properties**

Language=Default

**Examples 12.2\_en.properties**

Language=English

**Examples 12.2\_de.properties**

Language=German

## Logic Editor

Each of these entries in the resource bundle resolves differently for each locale with the following expression in the link editor:

localize("Language")

There are multiple locales that are involved in the determination of which properties file is used and this is determined in the following order: User Language -> System Language -> Default.

The localization feature is useful for returning localized error messages from within the application to an end user. The way this would commonly work would be a set of Keys would be maintained which would then have a value specific to the language of the properties file. For example:

**Examples 12.2\_en.properties**

EXCEPTION=Exception

**Examples 12.2\_de.properties**

EXCEPTION=Ausnahme

## Web Pages

The SAP MII product has support web page localization for JSP and IRPT via the use of a tag library since the 12.1 release. Since it is still somewhat new examples of this are included in this project. For IRPT & JSP pages the same precedence rules apply as in the logic engine and the locale is defined in the user's session. The following example pages demonstrate how to utilize this feature to localize your web pages:

Examples 12.2/WEB/Miscellaneous/Localize/Sample.irpt

Examples 12.2/WEB/Miscellaneous/Localize/Sample1.jsp

## XSLT & Navigation

It is also possible to localize values for a role/user homepage through the user and also in any XSL using the resource bundles of a project based. In order to localize a value in an XSLT the following must be setup in the header of the XSLT:

```
xmlns:java=http://xml.apache.org/xslt/java exclude-result-prefixes="java"
```

Next setup a custom theme for a role/user. This document provides more details on to accomplish this:

<https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/c004bc2a-013b-2c10-f2ab-a1e2f61e025f?overridelayout=true>

Create a custom attribute to specify the name of the default project and map it to a role/user (For Example "**LocalizationDefaultProject**"). The resource bundles specified in this default project are what will be used to specify the localization keys. Then in the previously specified default project, on the Meta-Inf tag create your resource bundles and localization key/value pairs for the various user languages.

In the DynamicHomePageLibrary.xsl for your custom theme create the following variable:

```
<!-- Retrieve the custom attribute DefaultProject from the user session -->
<xsl:variable name="LocalizationProject">
  <xsl:choose>
    <!-- When the value is set use it, otherwise use the hard-coded default project -->
    <xsl:when test="/Profile/@LocalizationDefaultProject != " ">
      <xsl:value-of select="/Profile/@LocalizationDefaultProject" />
    </xsl:when>
    <!-- If the DefaultProject session property isn't set then use this default value -->
    <xsl:otherwise>
      <xsl:value-of select=" 'SAP' " />
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
```

Next in the template here, <xsl:template name="writeTabItems">, modify the label so that it matches the following:

```
<!-- Previously here before localization -->
<!-- <xsl:value-of select="@label" /> -->
<!-- This is where the localization for the user language happens />-->
<xsl:value-of
select="java:com.sap.xmii.Illuminator.localization.WebLocalizer.getLocalizedString($LocalizationProject,
$Language, @label)" />
```

Then in the template here, <xsl:template name="writeValue">, modify the label so that it matches the following:

```
<!-- This is what used to be here prior to the localization call being added -->
<!-- <xsl:value-of select="$label" /> -->
<!-- This is where the localization for the user language happens />-->
<xsl:value-of
select="java:com.sap.xmii.Illuminator.localization.WebLocalizer.getLocalizedString($LocalizationProject,
$Language, $label)" />
```

If you want to view the localization key as hover over text you can always add in the following to your transform:

```
<xsl:attribute name="title">
  <xsl:value-of select="@label" /> <!-- may be $label depending on where in the xsl you are -->
</xsl:attribute>
```

To view the working example of this included in the project content the XSL is defined here:

Examples 12.2/WEB/Miscellaneous/Localize/Theme/DynamicHomePageLibrary.xsl

This XSL can be used as a theme for a user and you can define localized navigation trees for a project that match keys defined in the project resource bundles for a given language.

## PCo Query

The SAP MII product supports the ability to natively communicate with the SAP Plant Connectivity product. This interface has the same capabilities as the legacy tag query, but additionally it allows for retrieving tag meta-data and aggregation information from the historian that was not supported by the standard UDS queries. If the PCo Agent supports it, the PCo query template also supports the ability to make SQL like calls against the historian database using OLEDB query syntax. These queries are sometimes required in order to construct complex data reporting queries that are too complex for a standard tag query, and provide a complete range of reporting capabilities to the developer. Examples of these queries are provided here, but in general you will need to construct your own queries for the PCo Agents that exist in your landscape. The PCo Help documentation can be found online here:

[http://help.sap.com/content/documentation/sbs/docu\\_sbs\\_mii.htm](http://help.sap.com/content/documentation/sbs/docu_sbs_mii.htm)

There is more about the data server in the SAP MII 12.2 help documentation here:

[http://help.sap.com/saphelp\\_mii122/helpdata/en/4a/adcf2a4e4b21aae10000000a421138/content.htm](http://help.sap.com/saphelp_mii122/helpdata/en/4a/adcf2a4e4b21aae10000000a421138/content.htm)

## Peer to Peer

The SAP MII product has had the ability to communicate with other MII instances for multiple releases but the ability to leverage binary for remote transactions calls is part of the 12.1 and newer releases. The benefit of the binary approach is a reduction in the amount of data transmitted across the network and therefore generates lower loads on your networking infrastructure.

Synchronous communication between two MII instances is typically done via the user of Virtual Data server connections to access Data Server connections on remote MII instances to query their data. This interface has always leveraged binary communication and is useful in distributed architectures. It is more efficient than a direct connection to the remote data source because it uses a binary stream instead of a text based stream that something like a JDBC driver uses. The Virtual Connections are not shown here in the example as they are clearly outlined in the help documentation. In some cases it is necessary to communicate with a transaction on a remote system and it is possible to leverage binary communication in this case. Calling a remote transaction is not as well known and is provided here as one of the examples.

## Asynchronous via Messaging Services

To call a transaction asynchronously from another MII instance the recommended approach is to use the HTTP Message Listener interface provided by the Messaging Services. This interface provides message tracking capabilities and standard asynchronous processing functionality for immediate or buffered message processing. More information about this interface can be found here in the help documentation:

[http://help.sap.com/saphelp\\_mii122/helpdata/en/45/6a862988130dece10000000a11466f/content.htm](http://help.sap.com/saphelp_mii122/helpdata/en/45/6a862988130dece10000000a11466f/content.htm)

In order to properly configure this scenario go to the MII Menu and navigate to Message Services -> Message Processing Rules page. Then create a new rule with the following configuration values:

Details for BinaryMessage	
<b>Settings</b>	
Name:	BinaryMessage
Description:	Demonstrate how to process a binary message
Message Listener:	XMIIMESSEAGELISTENER
Message Type:	Web Service
Message Name:	BinaryMessage
Processing Type:	transaction

The next step is to define how the received message is processed which can be immediately triggered or sent to a category which will queue the messages in the order they were received. Then a scheduled transaction can read messages from this queue and process them accordingly. This example routes the messages directly to a transaction and this configuration is shown here:

Processing					
Transaction:	/Asynchronous/ProcessIncomingBinaryMessageFromMsgServices				
Persist Transaction:	On Error				
Log Level:	Error				
Parameters	<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/> InputXML</td> <td>ReceivedMessageXML</td> </tr> </tbody> </table>	Name	Value	<input checked="" type="checkbox"/> InputXML	ReceivedMessageXML
Name	Value				
<input checked="" type="checkbox"/> InputXML	ReceivedMessageXML				

The full transaction path is:

Examples 12.2/Miscellaneous/Peer2Peer/Asynchronous/ProcessIncomingBinaryMessageFromMsgServices

Once the rule is configured for your instance you can test the operation of this rule by opening the MII Workbench and navigating to the following location on the Catalog tab:

Examples 12.2/Miscellaneous/Peer2Peer/Asynchronous/PostBinaryXMLMessageToMsgServices

When this transaction is run a binary message is sent to the interface which then calls the Processing transaction. This transaction saves the received XML message to the following location on the Web tab:

Examples 12.2/WEB/Miscellaneous/Peer2Peer/Asynchronous/Test.xml

You can verify that the message was properly received via the XMIIMESSAGELISTENER by navigating from the MII Menu to Message Services -> Message Monitor and viewing the messages received by this listener.

#### Direct Call via Runner Servlet

Sometimes in a distributed MII environment it is necessary to synchronously call a transaction on a remote instance in order to pass along information between the systems. When there are concerns over network performance and/or the payload to be sent, you may wish to communicate through a binary interface. This can be done via the SAP MII Post action block which allows you to specify a binary content type in order to reduce the total load on the network. This action block makes use of the SAP MII Transaction Servlet interface which is defined here in the help documentation:

[http://help.sap.com/saphelp\\_mii122/helpdata/en/44/c5e701054c388ce10000000a11466f/frameset.htm](http://help.sap.com/saphelp_mii122/helpdata/en/44/c5e701054c388ce10000000a11466f/frameset.htm).

The calling transaction uses the post action to synchronously call another transaction which echoes back what it received and also saves the data as XML to the following location on the Web tab:

Examples 12.2/WEB/Miscellaneous/Peer2Peer/Synchronous/Test.xml

The calling transaction can be found here on the Catalog tab:

Examples 12.2/Miscellaneous/Peer2Peer/Synchronous/PostBinaryXMLMessage

To test this scenario simply run the calling transaction and verify that the XML message was returned in the response and that the Test.xml file was properly written.

## Query Caching

The query caching for a query template can now be managed in that it can be updated at a scheduled interval using a background task defined in the template. This task will update the cached results each time it runs so that the front-end will always hit a cached value instead of a user periodically having to “take the hit” due to an expired cache value. See the query template defined here for details:

Examples 12.2/Miscellaneous/Query Caching/CachedDataQueryTest

In the jobsconfig.xml in the Meta-Inf tab for the Examples 12.2 project, there is a task defined for it that looks similar to this:

The screenshot shows a job configuration form with the following fields:

- Name:** CachedDataQueryTest
- Description:** The schedule for demonstrating how the jobs scheduling works
- File:** Caching/CachedDataQueryTest.tsq (with a 'Load' button)
- Credential Alias:** SAPJCO
- Pattern:** 00 00 00 \*\* 2-6 (with a clock icon and an 'Example' button)

## SPC Chart

The SPC Chart now supports additional security controls for role based security to control which roles can view/edit/delete comments and which roles can suppress and un-suppress points on the SPC Chart. There is an example here that can be used as an example:

Examples 12.2/Miscellaneous/SPC Chart/DemonstrateUIRoleControls

The configuration for this capability is defined in the SPC Chart Display template in the Context Menu Security category and this editor looks like this:

The screenshot shows the SPC Chart configuration editor with the following sections:

- Allow Point Suppression:** ☒ (with a 'Refresh Available Roles' button)
- Allow Comment Editing:** ☒
- Point Suppression Roles:**
  - SAP\_XMII\_Administrator
  - SAP\_XMII\_Developer
  - SAP\_XMII\_User
- Comment Editing Roles:**
  - SAP\_XMII\_Administrator
  - SAP\_XMII\_Developer
  - SAP\_XMII\_User
- Available Roles (for Point Suppression):**
  - admin
  - Administrator
  - com.sap.caf.eu.gp.roles.superuser
  - DSS\_SYSTEM\_INFO
  - EDMAdmin
  - EDMEmployee
  - EDMProjectManager
- Available Roles (for Comment Editing):**
  - admin
  - Administrator
  - com.sap.caf.eu.gp.roles.superuser

## Step-Through Debug

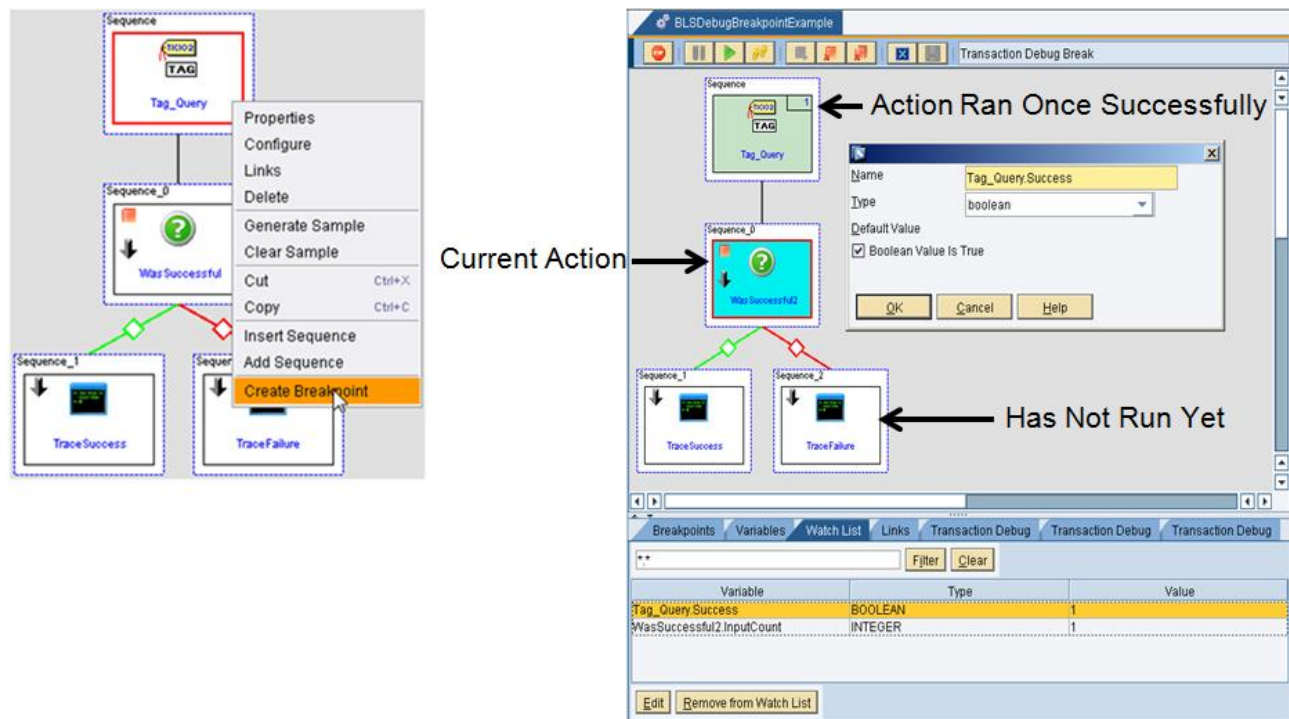
The transaction engine development environment now supports the ability to set break-points, watch variables, and control the operation of a given transaction from the workbench design time environment. When preparing to run the transaction it is possible to add in important values to a watch list for runtime transaction analysis. This feature also allows for viewing and setting variables mid-execution to enhance testing and debugging. The actions are color coded based on their execution state. The help documentation around this feature is located here:

[http://help.sap.com/saphelp\\_mii122/helpdata/en/4a/c0bf9750e821aae10000000a421138/content.htm](http://help.sap.com/saphelp_mii122/helpdata/en/4a/c0bf9750e821aae10000000a421138/content.htm)

An example transaction can be found here:

Examples 12.2/Miscellaneous/StepThroughDebug/BLSDDebugBreakpointExample

If the transaction is executed it will break at the conditional action. To test setting a variable, the input expression value can be set to true or false before continuing the transaction. Here is an example of what the Debug execution looks like:



## Template Security

There is a new feature for the query templates that has been implemented to reduce the vulnerability of an MII application from being accessed in a manner that might allow an advanced user control over sensitive data. The Query Template -> General -> Restrict Property Override checkbox will control what parameters can be overridden from the web at runtime and the parameters affected by this are shown in this example page:

Examples 12.2/WEB/Miscellaneous/Security/TemplateOverrides.jsp

There is an example query template with this feature enabled here:

Examples 12.2/Miscellaneous/Template Security/PropertyOverrideRestricted

Which you can attempt to run by entering the following URL in your browser:

<http://<Server>:<port>/XMII/Illuminator?QueryTemplate=Examples%2012.2/Miscellaneous/Template%20Security/PropertyOverrideRestricted&Method=HistoryEvent>



## Time zone calculations

There are new expression functions which allow for the calculation of time zone differences between the MII server and incoming timestamps. Each of the functions and its operation is demonstrated in the transaction:

Examples 12.2/Miscellaneous/TimeZone/TimeZoneTester

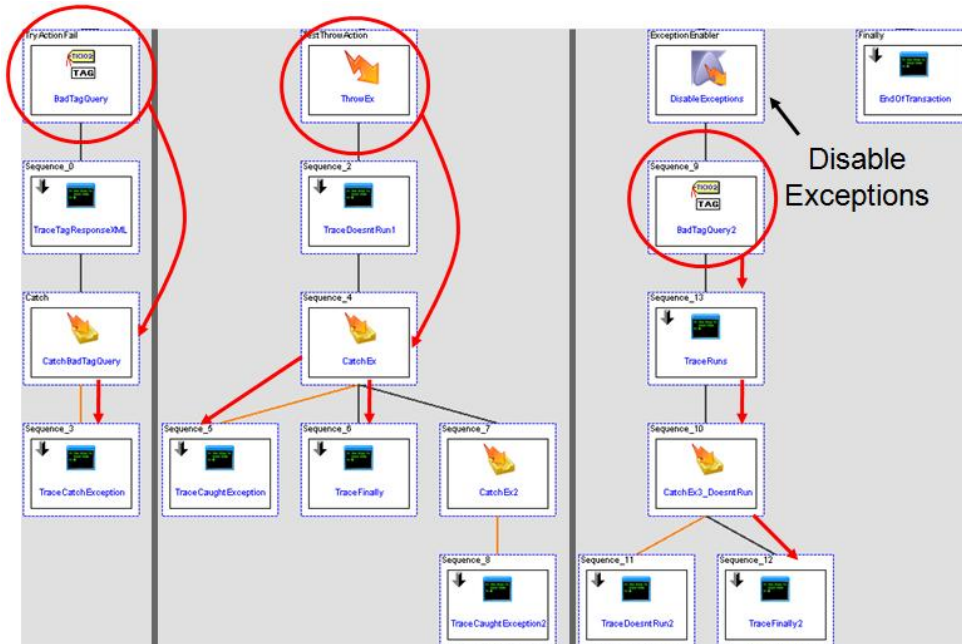
To see how each function is used open up the link editor for each of the tracer actions and press the "Evaluate" button or simply run the transaction.

## Try/Throw/Catch/Finally

The transaction engine now supports the programming concept of Try/Throw/Catch/Finally into the transaction design. If you are not familiar with the Try/Throw/Catch/Finally programming concept you can refer to the following link for more information: [http://en.wikipedia.org/wiki/Exception\\_handling\\_syntax](http://en.wikipedia.org/wiki/Exception_handling_syntax). There are two example transactions which help to demonstrate the usage of this feature and they are both located here:

Examples 12.2/Miscellaneous/TryCatch

It is recommended that each transaction execution is reviewed in detail in order to fully understand the capabilities of the throw, catch, and enable/disable action blocks.



It is also important to note that you can disable the throw on action and link errors for a given transaction by modifying the transaction attributes ThrowOnActionError & ThrowOnLinkError directly.



## Value Mapping

The value mapping feature of MII supports a basic value replacement of values in an XML document and also the ability to lookup a set of values for a given input set. Each scenario is outlined in the examples contained in the project here:

Examples 12.2/Miscellaneous/Value Mapping

Before these examples will work each rule defined in Appendix A of this document in the Value Mapping section may need to be imported. Save each off to a .csv file and import them separately via the Workbench -> Meta-Inf -> Examples 12.2 -> ValueMap.vmap editor if they do not already exist in your project (Note the delimiter character used is a semi-colon and not the default of comma).

## XML Document

To give an example on how the value mapping rules work open the XMLDocValueReplacement transaction and view the BAPI\_EQUI\_GETLIST transaction property. The first equipment listed in the XML is 000000000010002906 which conform to a rule defined in the value map. The rule will take the original equipment number found and remove the leading zeros from it. It is important to note the use of the #SourceValue# keyword in the expression which indicates that the original value is an input into the expression. To view the XML Document value replacement example which uses the Maintenance rule, open and run the transaction here:

Examples 12.2/Miscellaneous/Value Mapping/XMLDocValueReplacement

The transaction will modify the incoming XML document as shown in the image below:

### Original Document

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<BAPI_EQUI_GETLIST>
  <TABLES>
    <EQUIPMENT_LIST>
      <item>
        <EQUIPMENT>000000000010002906</EQUIPMENT>
        <DESCRIPT>AGG: CC superior equipment</DESCRIPT>
        <EQUICATGRY>M</EQUICATGRY>
        <PLANPLANT>0001</PLANPLANT>
        <MAINTPLANT>0001</MAINTPLANT>
        <PLANGROUP>
        <SORTFIELD>
      </item>
      <item>
        <EQUIPMENT>000000000010002907</EQUIPMENT>
        <DESCRIPT>AGH: CC superior equipment</DESCRIPT>
        <EQUICATGRY>M</EQUICATGRY>
        <PLANPLANT>0001</PLANPLANT>
        <MAINTPLANT>0001</MAINTPLANT>
        <PLANGROUP>
        <SORTFIELD>
      </item>
      <item>
        <EQUIPMENT>EHK-FHM003</EQUIPMENT>
        <DESCRIPT>EQUIPMENT(MASCHINE)</DESCRIPT>
        <EQUICATGRY>M</EQUICATGRY>
        <PLANPLANT>0001</PLANPLANT>
        <MAINTPLANT>0001</MAINTPLANT>
        <PLANGROUP>
        <SORTFIELD>
      </item>
      <item>
        <EQUIPMENT>EHK-FHM004</EQUIPMENT>
        <DESCRIPT>EQUIPMENT(Fertigungshilfsmittel)</DESCRIPT>
        <EQUICATGRY>P</EQUICATGRY>
        <PLANPLANT>0001</PLANPLANT>
        <MAINTPLANT>0001</MAINTPLANT>
        <PLANGROUP>
        <SORTFIELD>
      </item>
    </EQUIPMENT_LIST>
  </TABLES>
</BAPI_EQUI_GETLIST>
```

### Mapped Document

```
<?xml version="1.0" encoding="UTF-8"?>
<BAPI_EQUI_GETLIST>
  <TABLES>
    <EQUIPMENT_LIST>
      <item>
        <EQUIPMENT>10002906</EQUIPMENT>
        <DESCRIPT>AGG: CC superior equipment</DESCRIPT>
        <EQUICATGRY>M</EQUICATGRY>
        <PLANPLANT>0001</PLANPLANT>
        <MAINTPLANT>0001</MAINTPLANT>
        <PLANGROUP>
        <SORTFIELD>
      </item>
      <item>
        <EQUIPMENT>abc1234</EQUIPMENT>
        <DESCRIPT>AGH: CC superior equipment</DESCRIPT>
        <EQUICATGRY>M</EQUICATGRY>
        <PLANPLANT>0001</PLANPLANT>
        <MAINTPLANT>0001</MAINTPLANT>
        <PLANGROUP>
        <SORTFIELD>
      </item>
      <item>
        <EQUIPMENT>Machinel - Formerly (EHK-FHM003)</EQUIPMENT>
        <DESCRIPT>EQUIPMENT(MASCHINE)</DESCRIPT>
        <EQUICATGRY>M</EQUICATGRY>
        <PLANPLANT>0001</PLANPLANT>
        <MAINTPLANT>0001</MAINTPLANT>
        <PLANGROUP>
        <SORTFIELD>
      </item>
      <item>
        <EQUIPMENT>EHK-FHM004</EQUIPMENT>
        <DESCRIPT>EQUIPMENT(Fertigungshilfsmittel)</DESCRIPT>
        <EQUICATGRY>P</EQUICATGRY>
        <PLANPLANT>0001</PLANPLANT>
        <MAINTPLANT>0001</MAINTPLANT>
        <PLANGROUP>
        <SORTFIELD>
      </item>
    </EQUIPMENT_LIST>
  </TABLES>
</BAPI_EQUI_GETLIST>
```

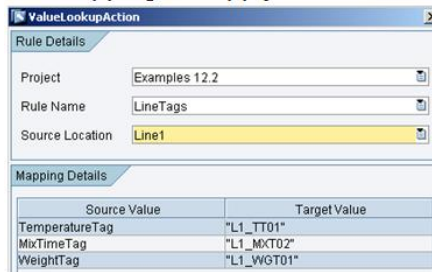
## Value Lookup

Value mapping also supports looking up a set of values at runtime in order to simplify the translation of data values from one system to another so this approach provides a namespace to these lookup values. A working example of this scenario can be found here:

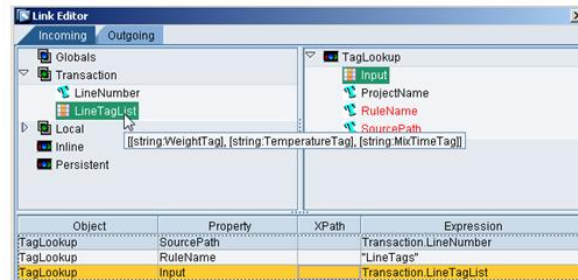
Examples 12.2/Miscellaneous/Value Mapping/ValueMapLookup

The example works as follows; there are multiple production lines that operate in a similar fashion however the details in how the data is retrieved from the data systems vary slightly. In the configuration of the action the following is performed:

The action configuration defines the “namespace” used to identify the mappings to apply.



A list of inputs are passed into the action in order to lookup their mapped values.



A map of name/value pairs is returned:

**{MixTimeTag=L1\_MXT02, TemperatureTag=L1\_TT01, WeightTag=L1\_WGT01}**

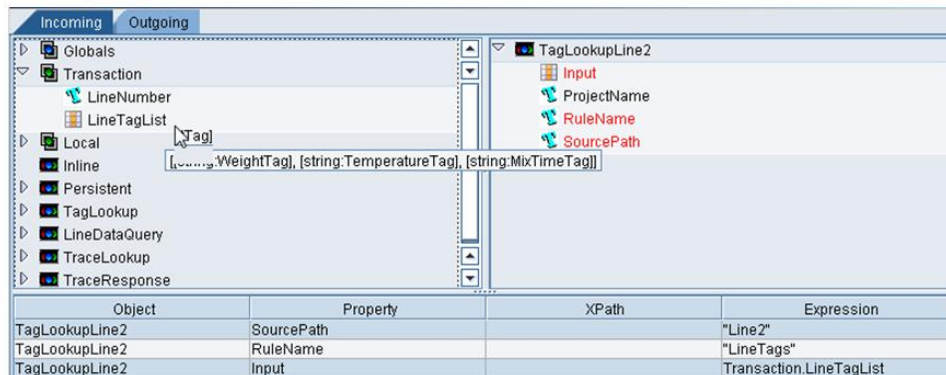
The values can be accessed via the following expression:

**TagLookup.Output{"WeightTag"}**

Which returns the value “L1\_WGT01”

However, when the parameter for Source path is linked to the action with a different value, Line 2, different values are returned:

The action configuration defines the “namespace” used to identify the mappings to apply.



A map of name/value pairs is returned:

**{MixTimeTag=L2\_MXT02, TemperatureTag=L2\_TT01, WeightTag=L2\_WGT01}**

The values can be accessed via the following expression:

**TagLookup.Output{"WeightTag"}**

Which returns the value “L2\_WGT01”

## Appendix A

### Miscellaneous Custom Action

Here is the source code used to create the SAPMIISDKTest.jar action example.

#### [InterfaceTestActions.java](#)

```
package com.sap.mii.custom.actions.credquery;

import java.util.HashMap;
import java.util.Map;

import com.sap.lhcommon.common.LogLevel;
import com.sap.lhcommon.common.NamedVariantData;
import com.sap.lhcommon.common.VariantData;
import com.sap.lhcommon.common.VariantDataTypes;
import com.sap.lhcommon.exceptions.DataConversionException;
import com.sap.lhcommon.xml.XMLDataType;
import com.sap.xml.bls.sdk.*; // This is the main SDK for custom actions.
import com.sap.xml.storage.connections.RemoteConnectionType;

public class InterfaceTestActions {

    /**
     * Run a Illuminator query and return the result using the query
     * interface.
     */
    @Action(name = "Query")
    @Outputs(names = { "Result", "LastErrorMessage" },
             types = { VariantDataTypes.XML, VariantDataTypes.STRING })
    @ConfigurationDialog(dialogClass = QueryDialog.class, localizationType =
ConfigurationDialogLabelLocalizationType.BUILT_IN)
    public static boolean query(
        IActionInstance action,
        IQueryInstance query,
        @Input(name = "QueryTemplate") String templateName,
        @Input(name = "Timeout", defaultValue = "60") int timeout)
        throws InvalidVariableException {

        boolean succeeded = true;
        XMLDataType result = null;
        String errorMessage = "";

        try {
            Map<String,String> params = new HashMap<String,String>();

            // execute the query
            result = query.execute(templateName, params, timeout * 1000);

        } catch (Exception e) {
            action.log(LogLevel.ERROR, "Error running query " + templateName + ", " +
e.getMessage());

            errorMessage = e.getLocalizedMessage();
            succeeded = false;
        }

        action.setActionResult("Result", result);
    }
}
```

```

        action.setActionResult("LastErrorMessage", errorMessage);

        return succeeded;
    }

    /**
     * Run a Illuminator query and return the result using the query
     * interface.
     * @throws DataConversionException
     */
    @Action(name = "Credential")
    @Outputs(names = { "Username", "Password", "LastErrorMessage" },
              types = { VariantDataTypes.STRING, VariantDataTypes.STRING,
VariantDataTypes.STRING })
    @ConfigurationDialog(dialogClass = CredentialAliasDialog.class, localizationType =
ConfigurationDialogLabelLocalizationType.BUILT_IN)
    public static boolean credential(
        IActionInstance action,
        IAdminInstance helper,
        @Input(name = "Name") String name)
        throws InvalidVariableException, DataConversionException {

        boolean succeeded = true;
        NamedVariantData cred = null;
        String errorMessage = "";

        try {
            cred = helper.getCredential(name);

        } catch (Exception e) {
            action.log(LogLevel.ERROR, "Error looking up credential " + name + ", " +
e.getMessage());

            errorMessage = e.getLocalizedMessage();
            succeeded = false;
        }

        action.setActionResult("Username", cred.getName());
        action.setActionResult("Password", cred.getObject().toString());
        action.setActionResult("LastErrorMessage", errorMessage);

        return succeeded;
    }

    /**
     * Run a Illuminator query and return the result using the query
     * interface.
     * @throws DataConversionException
     */
    @Action(name = "Connection")
    @Outputs(names = { "Properties", "LastErrorMessage" },
              types = { VariantDataTypes.MAP, VariantDataTypes.STRING })
    @ConfigurationDialog(dialogClass = ConnectionAliasDialog.class, localizationType =
ConfigurationDialogLabelLocalizationType.BUILT_IN)
    public static boolean connection(
        IActionInstance action,
        IAdminInstance helper,
        @Input(name = "Name") String name,

```

```

        @Input(name = "Type") int type)
        throws InvalidVariableException, DataConversionException {

    boolean succeeded = true;
    VariantData result = null;
    String errorMessage = "";

    try {
        action.log(LogLevel.INFO, "Name: "+name);
        action.log(LogLevel.INFO, "Type: "+type);

        RemoteConnectionType rtype = RemoteConnectionType.lookup(type);

        Map<String,String> props = helper.getConnection(name, rtype);
        result = new VariantData(props);
        action.log(LogLevel.INFO, "Properties: "+result);

    } catch (Exception e) {
        action.log(LogLevel.ERROR, "Error looking up connection " + name + ", " +
e.getMessage());

        errorMessage = e.getLocalizedMessage();
        succeeded = false;
    }

    action.setActionResult("Properties", result);
    action.setActionResult("LastErrorMessage", errorMessage);

    return succeeded;
}
}

```

#### ConnectionAliasDialog.java

```

package com.sap.mii.custom.actions.credquery;

import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;

import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JPanel;

import com.sap.lhcommon.common.VariantData;
import com.sap.lhcommon.exceptions.DataConversionException;
import com.sap.xmii.bls.sdk.BaseConfigurationDialog;
import com.sap.xmii.bls.sdk.ConfigurationDialogUtils;
import com.sap.xmii.bls.sdk.IActionConfiguration;
import com.sap.xmii.storage.connections.RemoteConnectionType;

public class ConnectionAliasDialog extends BaseConfigurationDialog {

    private static final long serialVersionUID = 1L;

    protected JComboBox connectionAliasBox;

    public ConnectionAliasDialog() {
        super(400, 400, "Connection Alias Configuration (WAS)");
    }
}

```

```

        connectionAliasBox = new JComboBox();

        try {
            ConfigurationDialogUtils.populateConnectionAlias(connectionAliasBox,
RemoteConnectionType.WAS);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    protected void load() {

        JPanel mainPanel = new JPanel(new GridBagLayout());
        GridBagConstraints c = new GridBagConstraints();

        c.gridy = 0;

        /* Connection Alias */
        c.gridy = c.gridy + 1;
        c.weightx = 0.1;
        mainPanel.add(new JLabel("Name"), c);
        mainPanel.add(connectionAliasBox , c);

        getContentPane().setLayout(new GridBagLayout());
        GridBagConstraints e = new GridBagConstraints();

        e.gridy = 0;
        e.weightx = 1.0;
        e.weighty = 1.0;
        e.fill = GridBagConstraints.BOTH;
        getContentPane().add(mainPanel, e);

        e.gridy = e.gridy + 1;
        e.weightx = 1.0;
        e.weighty = 0.0;
        e.fill = GridBagConstraints.NONE;
        e.anchor = GridBagConstraints.LINE_END;

        super.createOKCancelPanel(this, e);

        // populate fields
        IActionConfiguration config = this.getConfiguration();
        try {
            String alias = config.getStaticInput("Name").stringValue();
            connectionAliasBox.setSelectedItem(alias);

        } catch (DataConversionException ex) {
            System.out.println("Unable to load dialog properties," + ex.getLocalizedMessage());
        }
    }

    @Override
    protected void saveData() {
        String alias = null;
        if(connectionAliasBox.getSelectedIndex() > 0) {
            alias = String.valueOf(connectionAliasBox.getSelectedItem());
        }
    }

```

```

    }
    this.getConfiguration().setStaticInput("Name", new VariantData(alias));
}
}

```

### CredentialAliasDialog.java

```

package com.sap.mii.custom.actions.credquery;

import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;

import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JPanel;

import com.sap.lhcommon.common.VariantData;
import com.sap.lhcommon.exceptions.DataConversionException;
import com.sap.xmii.bls.sdk.BaseConfigurationDialog;
import com.sap.xmii.bls.sdk.ConfigurationDialogUtils;
import com.sap.xmii.bls.sdk.IActionConfiguration;

public class CredentialAliasDialog extends BaseConfigurationDialog {

    private static final long serialVersionUID = 1L;

    protected JComboBox credentialAliasBox;

    public CredentialAliasDialog() {
        super(400, 400, "Credential Alias Configuration");

        credentialAliasBox = new JComboBox();

        try {
            ConfigurationDialogUtils.populateCredentialAlias(credentialAliasBox);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    protected void load() {

        JPanel mainPanel = new JPanel(new GridBagLayout());
        GridBagConstraints c = new GridBagConstraints();

        c.gridy = 0;

        /* Connection Alias */
        c.gridy = c.gridy + 1;
        c.weightx = 0.1;
        mainPanel.add(new JLabel("Name"), c);
        mainPanel.add(credentialAliasBox, c);

        getContentPane().setLayout(new GridBagLayout());
        GridBagConstraints e = new GridBagConstraints();

        e.gridy = 0;
    }
}

```



```

e.weightx = 1.0;
e.weighty = 1.0;
e.fill = GridBagConstraints.BOTH;
getContentPane().add(mainPanel, e);

e.gridy = e.gridy + 1;
e.weightx = 1.0;
e.weighty = 0.0;
e.fill = GridBagConstraints.NONE;
e.anchor = GridBagConstraints.LINE_END;

        super.createOKCancelPanel(this, e);

// populate fields
    IConfiguration config = this.getConfiguration();
    try {
        String alias = config.getStaticInput("Name").stringValue();
        credentialAliasBox.setSelectedItem(alias);

    } catch (DataConversionException ex) {
        System.out.println("Unable to load dialog properties," + ex.getLocalizedMessage());
    }
}

@Override
protected void saveData() {
    String alias = null;
    if(credentialAliasBox.getSelectedIndex() > 0) {
        alias = String.valueOf(credentialAliasBox.getSelectedItem());
    }

    this.getConfiguration().setStaticInput("Name", new VariantData(alias));
}
}

```

#### QueryDialog.java

```

package com.sap.mii.custom.actions.credquery;

import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

import com.sap.lhcommon.common.VariantData;
import com.sap.lhcommon.exceptions.DataConversionException;
import com.sap.xmii.bls.sdk.BaseConfigurationDialog;
import com.sap.xmii.bls.sdk.IActionConfiguration;

public class QueryDialog extends BaseConfigurationDialog {

    private static final long serialVersionUID = 1L;

    private JTextField templateName;
    private JTextField timeout;

```



```

    public QueryDialog() {
        super(400, 400, "Query Configuration");

        templateName = new JTextField();
        templateName.setMinimumSize(new Dimension(80, 22));
        templateName.setPreferredSize(new Dimension(200, 22));

        timeout = new JTextField();
        timeout.setMinimumSize(new Dimension(80, 22));
        timeout.setPreferredSize(new Dimension(200, 22));
    }

    @Override
    protected void load() {
        JPanel mainPanel = new JPanel(new GridBagLayout());
        GridBagConstraints c = new GridBagConstraints();

        c.gridy = 0;

        /* Connection Alias */
        c.gridy = c.gridy + 1;
        c.weightx = 0.1;
        mainPanel.add(new JLabel("Template Name"), c);
        mainPanel.add(templateName, c);

        c.gridy = c.gridy + 1;
        c.gridwidth = 1;
        mainPanel.add(new JLabel("Timeout (seconds)"), c);
        mainPanel.add(timeout, c);

        getContentPane().setLayout(new GridBagLayout());
        GridBagConstraints e = new GridBagConstraints();

        e.gridy = 0;
        e.weightx = 1.0;
        e.weighty = 1.0;
        e.fill = GridBagConstraints.BOTH;
        getContentPane().add(mainPanel, e);

        e.gridy = e.gridy + 1;
        e.weightx = 1.0;
        e.weighty = 0.0;
        e.fill = GridBagConstraints.NONE;
        e.anchor = GridBagConstraints.LINE_END;

        super.createOKCancelPanel(this, e);

        // populate fields
        IActionConfiguration config = this.getConfiguration();
        try {
            templateName.setText(config.getStaticInput("QueryTemplate").stringValue());
            timeout.setText(config.getStaticInput("Timeout").stringValue());
        } catch (DataConversionException ex) {
            System.out.println("Unable to load dialog properties," + ex.getLocalizedMessage());
        }
    }

```

```

    @Override
    protected void saveData() {
        this.getConfiguration().setStaticInput("QueryTemplate", new
VariantData(templateName.getText()));
        this.getConfiguration().setStaticInput("Timeout", new VariantData(timeout.getText()));
    }
}

```

### Catalog.xml

```

<?xml version="1.0" encoding="utf-8"?>
<ComponentCatalog>
  <Category Name="SDK Test" Description="SDK Test">
    <Component Type="Action" Name="Credential" Description="" Label="Credential Alias"
ClassName="com.sap.mii.custom.actions.credquery.InterfaceTestActions"
AssemblyName="SAPMIISDKTest.jar" Dependencies="" HelpFileName="" />
    <Component Type="Action" Name="Connection" Description="" Label="Connection Alias"
ClassName="com.sap.mii.custom.actions.credquery.InterfaceTestActions"
AssemblyName="SAPMIISDKTest.jar" Dependencies="" HelpFileName="" />
    <Component Type="Action" Name="Query" Description="" Label="Query Inheritance"
ClassName="com.sap.mii.custom.actions.credquery.InterfaceTestActions"
AssemblyName="SAPMIISDKTest.jar" Dependencies="" HelpFileName="" />
  </Category>
</ComponentCatalog>

```

### Value Mapping

The following sections contain the raw CSV version of the exported rules for value mapping where a semi-colon was used as the Delimiter character for each of the exports. Save each of these as a .csv file and import them into the ValueMap.vmap editor via the Import button.

#### Maintenance Rule

RuleName;RuleDescription;Source;SourceDocument;SourceValue;TargetValue;MatchType

Maintenance;Replace the asset tag with the equipment  
number;/BAPI\_EQUI\_GETLIST/TABLES/EQUIPMENT\_LIST/item/EQUIPMENT;IC/WEB/ActionUpdates/Valu  
eMapping/BAPI\_EQUI\_GETLIST.xml;000000000010002906;format("00000000000" & #SourceValue#,  
"#####");1

Maintenance;Replace the asset tag with the equipment  
number;/BAPI\_EQUI\_GETLIST/TABLES/EQUIPMENT\_LIST/item/EQUIPMENT;IC/WEB/ActionUpdates/Valu  
eMapping/BAPI\_EQUI\_GETLIST.xml;10002907;"aa10002907";2

Maintenance;Replace the asset tag with the equipment  
number;/BAPI\_EQUI\_GETLIST/TABLES/EQUIPMENT\_LIST/item/EQUIPMENT;IC/WEB/ActionUpdates/Valu  
eMapping/BAPI\_EQUI\_GETLIST.xml;EHK-FHM003;"Machine1 - Formerly (EHK-FHM003)";1

Maintenance;Replace the asset tag with the equipment  
number;/BAPI\_EQUI\_GETLIST/TABLES/EQUIPMENT\_LIST/item/EQUIPMENT;IC/WEB/ActionUpdates/Valu  
eMapping/BAPI\_EQUI\_GETLIST.xml;\*;"RegexComparison";3

#### LineTags Rule

RuleName;RuleDescription;Source;SourceDocument;SourceValue;TargetValue;MatchType

LineTags;Get Tags by line for specific values;Line1;Line1;TemperatureTag;"L1\_TT01";1

LineTags;Get Tags by line for specific values;Line1;Line1;MixTimeTag;"L1\_MXT02";1

LineTags;Get Tags by line for specific values;Line1;Line1;WeightTag;"L1\_WGT01";1

LineTags;Get Tags by line for specific values;Line2;Line2;TemperatureTag;"L2\_TT01";1

LineTags;Get Tags by line for specific values;Line2;Line2;WeightTag;"L2\_WGT01";1

LineTags;Get Tags by line for specific values;Line2;Line2;MixTimeTag;"L2\_MXT02";1

## Related Content

SAP MII Help Documentation: <http://help.sap.com> -> Business Suite -> SAP Manufacturing Integration and Intelligence

SAP Ramp-up Knowledge Transfer: <http://service.sap.com/rkt>

SAP Online Knowledge Products: <http://service.sap.com/okp>

SAP MII Forum: <https://www.sdn.sap.com/irj/sdn/forum?forumID=237>

SAP MII Wiki: <https://wiki.sdn.sap.com/wiki/display/xMII>

## Copyright

© Copyright 2009 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects S.A. in the United States and in other countries. Business Objects is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.