How to Use xMII to Report an SAP Query (Includes the Solution for Implementing a Generic Interface)

Applies to:

xApp Manufacturing Integration & Intelligence version 11.5.x

Summary

The purpose of this document is to provide a high level overview of the steps that need to be taken to create and report from an SAP Query using xMII.

Author(s): Salvatore Castro

Company: SAP Labs, LLC

Created on: 23 October 2006

Author Bio

Salvatore Castro of SAP Labs has a Bachelors Degree in Computer Engineering and is currently working on completing his Masters Degree in Computer Science both through the Rochester Institute of Technology. He is a member of the Engineering and Field Enablement Services group of xMII under Mo Ghanem.

Table of Contents

Overview 3

Process Flow 3

xMII & SAP Query Architecture 5

Building an xMII Generic Reporting Transaction 8

Appendix 11

SAP Query Transform to xMII Rowset Transaction 11

SAP Query Report Transaction 20

SAP Query Web Page 30

Related Content 34

Copyright 35

Overview

In the ERP system there are many ways to retrieve data but none are as flexible or as specific to your company as using the SAP Query capabilities. The SAP Query allows access to a variety of data sources within the ERP system such as Logical Databases, Table(s), and Custom ABAP reports. These data stores and ABAP reports can then be accessed via queries and filtered using different variant settings. An xMII interface can be generically defined to report from any query results using the AQRC Function Module RFC calls as the interface for requesting the data. There are five primary RFC calls that are used in this document so be sure that you have the proper rights and permissions to access the following:

RSAQ_REMOTE_USERGROUP_CATALOG

RSAQ_REMOTE_FUNCAREA_CATALOG

RSAQ_REMOTE_QUERY_CALL_CATALOG

RSAQ_REMOTE_QUERY_CATALOG

RSAQ_REMOTE_QUERY_CALL

It is also possible to create and display these queries through the SAP GUI Interface and it is recommended that their operation is tested ahead of time. The SAP GUI transactions that pertain to this type of reporting are: SQ03, SQ02, and SQ01. However, it is only necessary to have access to SQ01 for SAP query development and modification. Each of the transactions corresponds to creating a logical component for organizing access and permissions for performing the Query. The SQ03 transaction is used to create a Group, the SQ02 transaction is used for creating a Functional Area which will be referred to as an InfoSet, and the SQ01 transaction is used to create queries, quick views of queries, and defining query variants. When creating any of these fields be sure to add meaningful descriptions as they will show up in the interface. There are additional RFC calls located in the AQRC function group however their use is not supported via the xMII JCO interface.

Process Flow

The SAP Query mechanism allows a user to define columns of data from various available data sources and can allow for very company specific data to be retrieved with minimal effort. The creation of the queries themselves is out of the scope of this document. For more information on how to create Groups (SQ03), Info-Sets (Functional Areas, SQ02), and Queries (SQ01) please refer to the following document: http://searchsap.techtarget.com/searchSAP/downloads/Teach_yourself_SAP_C20.pdf and the title of this document is “Reporting Tools in SAP (SAP Query, InfoSet Query, Ad Hoc Query, and QuickViewer)” By SAMS Publishing. From the xMII perspective looking up a query can be done following the hierarchical process outlined in Diagram 1.

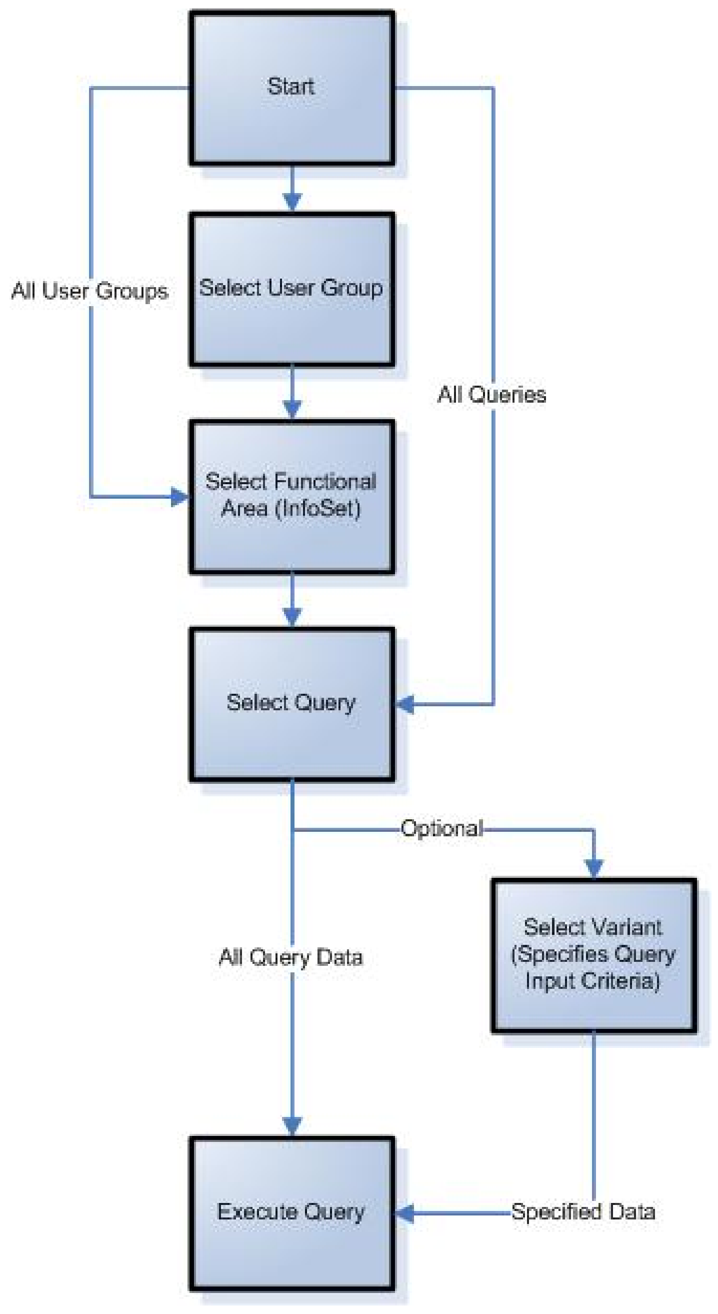
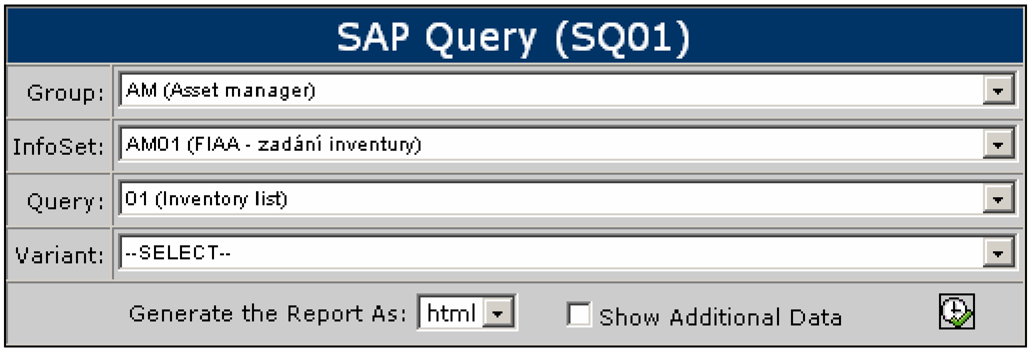


Diagram 1: Process Flow for Finding and Executing a Query

Specifying all input parameters is the recommended method for locating specific queries in the ERP system in order to limit the number of available choices. It is possible to specify a wildcard, *, in the input fields to widen the search as shown in Diagram 1 by the multiple starting points. The variant selection in this process is an optional field value that allows for returning a subset of the selected query by setting input field values. The variant definitions can only be defined via the SAP GUI interface and are specific to each of the queries.

xMII & SAP Query Architecture

The xMII system architecture and process flow can be matched up to the following screen:


SAP Query (SQ01)

Group: AM (Asset manager)

InfoSet: AM01 (FIAA - zadání inventury)

Query: 01 (Inventory list)

Variant: --SELECT--

Generate the Report As: html

☐ Show Additional Data

Image 1: Example Page for SAP Query Reporting

When this page loads the Group, InfoSet (Functional Area), and Query drop down lists populate with the names and data-linked descriptions that correspond with the values defined in the ERP system. The Variant drop down list box only updates when a query selection has been made since the variants are query specific. It is possible to browse them by Group and InfoSet but it is not practical for this application since variants are Query specific. When the "execute" button is clicked a page is opened that displays the results of the report in HTML, XML or CSV format. If the "Show Additional Data" checkbox is selected then the additional data item fields returned by the Query will be displayed in separate rowsets otherwise only the first set of data is displayed. This interface allows for access to various levels of possibly sensitive information so your page may wish to lock down the available capabilities.

The RFC calls used to populate the drop down list fields contain both the Technical name and Description fields for that specific component which are used to feed filtering parameters for the subsequent drop downs or to populate the div values. These fields will be identified later along with the required input fields and their input field values. It is recommended that some standard ERP to xMII XML conversion utility be used; the recommended utility is referenced in the "Related Content" section of this document.

The RFC calls that were made along with their inputs in order to populate the list boxes are as follows:

Field Name	RFC Call	Inputs	Name Field	Description Field

Group	RSAQ_REMOTE_USERGROUP_CATALOG	GENERIC_USERGROUP = “*” (Group Name)
		
		WITH_SYSTEM_OBJECT = “X”
			NUM	UTEXT

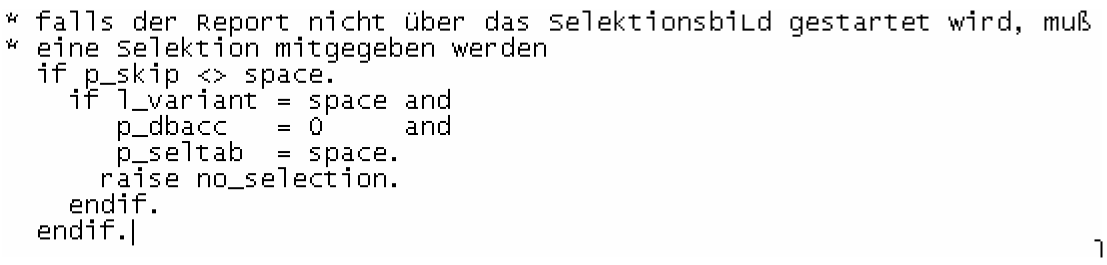
InfoSet	RSAQ_REMOTE_FUNCAREA_CATALOG	SAME AS GROUP GENERIC_FUNCAREA = “*” (InfoSet Name)	CLAS	UTEXT
Query Catalog	RSAQ_REMOTE_QUERY_CATALOG	SAME AS INFOSET GENERIC_QUERYNAME = “*” (Query Name)	QUERY	QTEXT
Variant	RSAQ_REMOTE_QUERY_CALL_CATALOG	SAME AS QUERY CATALOG	VARIANT	VTEXT

Query Call	RSAQ_REMOTE_QUERY_CALL	SAME AS QUERY CATALOG
		
		SKIP_SELSCREEN = “X”
		
		EXTERNAL_PRESENTATION = “X”
		
		DATA_TO_MEMORY = “X”
		
		*DBACC = stringif(variant == “”, “9”, “0”)
			LDATA	LISTDESC

Table 1: RFC calls used to populate the various input fields

The input JCO response XML structure requires some manipulation to convert the itemized structure to xMII document format (See the Related Content section for additional information on developing a conversion utility transaction). The actual JCO Query call requires a couple more input fields but is still fundamentally the same as the Query Catalog above, with the exception of a couple of fields for how the data is returned by the call.

The trickiest field associated with the query call is the DBACC field and how it affects the results. This field is described as the Database access field and will specify the total number of database accesses the query is allowed to make. The field size is defined as INTEGER 11 (default is 0), but when testing the BAPI call only a single integer value was accepted as the input to this field, consequently only values from 0-9 are valid inputs to this field. As a result of this you will see the conditional if no variant is defined to set this parameter to 9 otherwise set it to the default value. In the ABAP code for the call the following code snippet was found that will help to explain why this is the case:

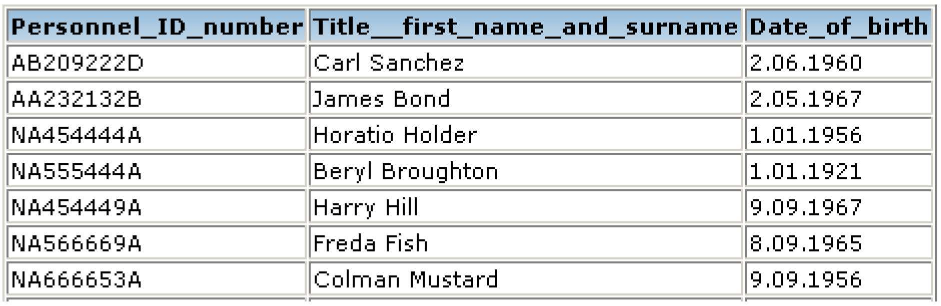


```
* falls der Report nicht über das Selektionsbild gestartet wird, muß
* eine Selektion mitgegeben werden
  if p_skip <> space.
    if l_variant = space and
       p_dbacc   = 0     and
       p_seltab  = space.
      raise no_selection.
    endif.
  endif.
```

Image 2: Code snippet from the RSAQ_REMOTE_QUERY_CALL RFC

What this code snippet is indicating is that if no filtering criteria are specified for the query then there must be a limit on the number of accesses to the database in order to prevent the user from executing a bad query. (The “no_selection” exception is not returned by the RFC in the response XML but rather all of the nodes are NULL values).

The data returned by the Query Call can be handled in a generic fashion and logically separated based upon the order of the column descriptions returned and the sequence of the data. The column definitions are returned in the following location of the JCO Response XML: “/RSAQ_REMOTE_QUERY_CALL/TABLES/LISTDESC/item” where each item returned represents a new column. In order to ensure that the columns are compatible with a standard XML format the names are encoded using the following function xmlencodename(/item/FDESC) and the column description field is “/item/FNAMENEW”. The FDESC node under each item contains the column name used when generating the return xMII XML doc. Once the columns are created in the xMII Doc the row data can now be added and this is located under “/RSAQ_REMOTE_QUERY_CALL/TABLES/LDATA/item”. The data may or may not contain additional items but from the referenced interface, Image 1, it is optional to return them by checking the “Show Additional Data” box. The data format comes back all in a single LINE node for each item where each row is semicolon delimited and each column in that row is comma delimited. In addition to this each column value is prefixed with a three digit numerical value that describes the length of the data value for that column. These two values, the length and value, are colon delimited so an example LINE node value looks like this: 001:0,022:0 - No time evaluation,010:01.04.1998,001:0; and may or may not end with a slash after the semicolon.



Personnel_ID_number	Title__first_name_and_surname	Date_of_birth
AB209222D	Carl Sanchez	2.06.1960
AA232132B	James Bond	2.05.1967
NA454444A	Horatio Holder	1.01.1956
NA555444A	Beryl Broughton	1.01.1921
NA454449A	Harry Hill	9.09.1967
NA566669A	Freda Fish	8.09.1965
NA666653A	Colman Mustard	9.09.1956

Image 3: Screen shot of sample query results in HTML format

Building an xMII Generic Reporting Transaction

The business logic environment provided by the xMII product allows for the creation of a completely generic utility that will allow the user to call any ERP Query and have the data returned to their web browser. The process is which this can be accomplished is shown by the Pseudo code below:

```
XML Call_Query( string QueryName, bool showAdditional, string UserGroup, string Variant )
    XML returnXML = “”
    XML resp = JCO_Call.RSAQ_REMOTE_QUERY_CALL( QueryName, UserGroup, Varaint)
    For( x=1; x<=DataItemsCount; x++ )
        returnXML += call( TransformToxMII( resp[x] ) )
    If( NOT( showAdditional ) )
        BreakLoop
    RETURN returnXML
XML TransformToxMII( XML DataItem )
    XML returnXML = “”
    String primaryParse = item/LINE
    String currentRow = “”
    XML currentValues = “”;
    For( x=1; x<=ColumnCount; x++ )
        returnXML.addColumn( item/FDESC )
    whileIndex = 0;
    While( primaryParse != NULL )
        whileIndex++
        currentRow = primaryParse( whileIndex )
        currentValues = StringListToXML( currentRow )
        returnXML.insertRow()
        For( x=1; x<=currentValuesCount; x++ )
            returnXML.addData( column[x], stringright(/Row/Item, columnValueLength )
        if( primaryParse == “” )
            BreakWhileLoop
    RETURN returnXML
```

Here is the process flow for the above pseudo code:

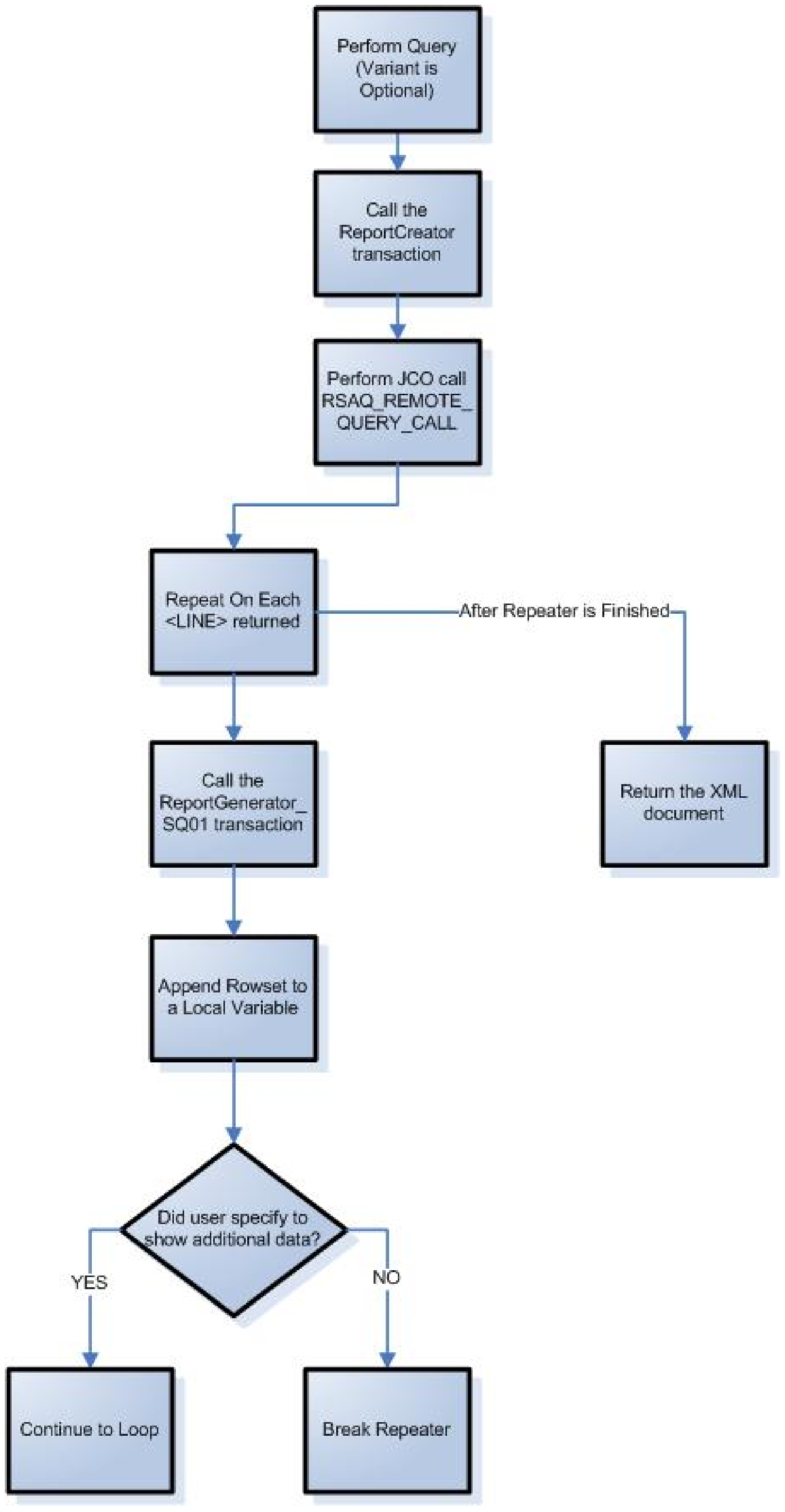

Perform Query (Variant is Optional)
Call the ReportCreator transaction
Perform JCO call RSAQ_REMOTE_QUERY_CALL
Repeat On Each <LINE> returned
After Repeater is Finished
Return the XML document
Call the ReportGenerator_SQ01 transaction
Append Rowset to a Local Variable
Did user specify to show additional data?
YES
NO
Continue to Loop
Break Repeater

Diagram 2: The process flow for the Report Creator Transaction

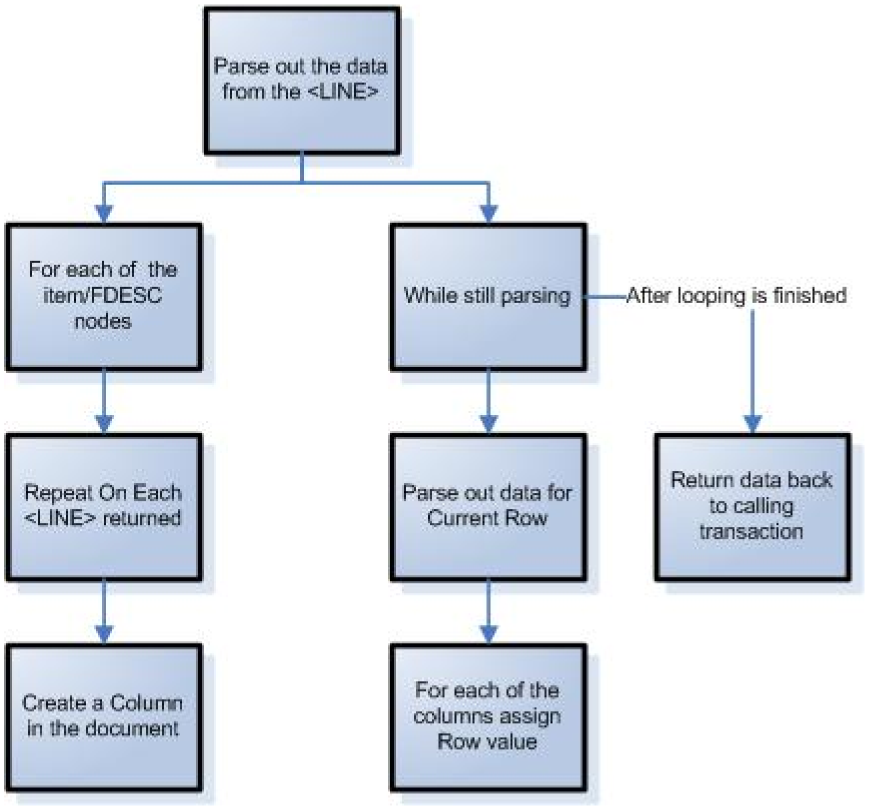

Parse out the data from the <LINE>
For each of the item/FDESC nodes
While still parsing
After looping is finished
Repeat On Each <LINE> returned
Parse out data for Current Row
Return data back to calling transaction
Create a Column in the document
For each of the columns assign Row value

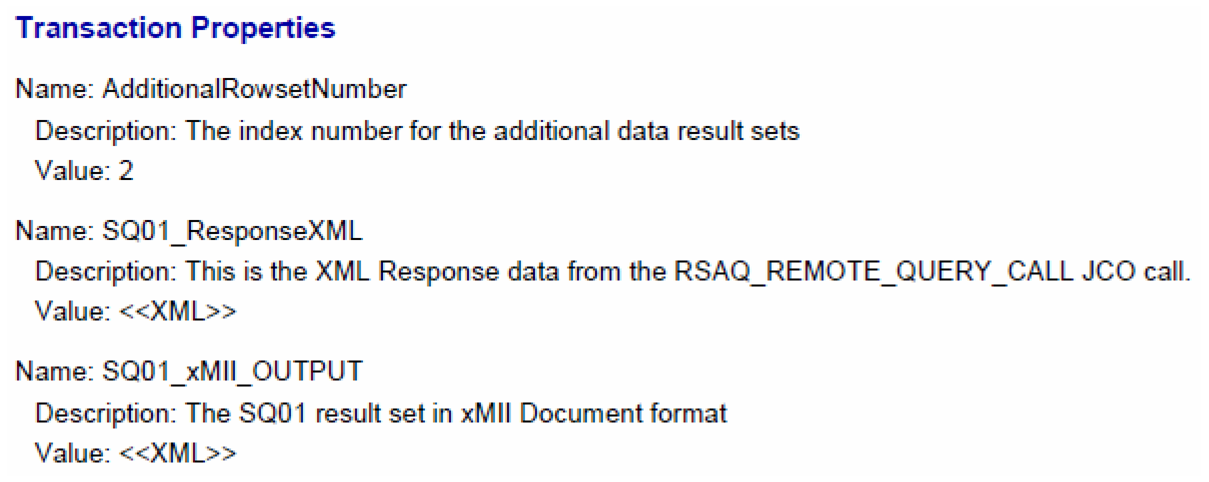
Diagram 3: The conversion utility used by the Report Creator Transaction

More documentation on the configuration of each of these transactions is located in the appendix portion of this document and can be used as a reference in creating your own ERP Query handling transactions.

Appendix

SAP Query Transform to xMII Rowset Transaction

Create the following transaction and save it as “ReportGenerator_SQ01”

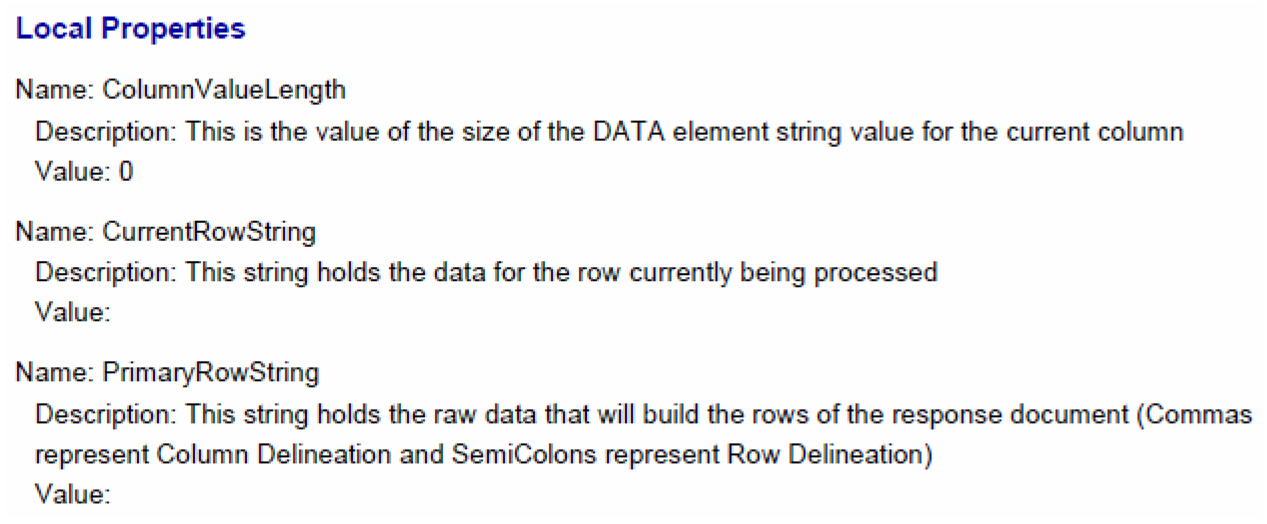


Transaction Properties

Name: AdditionalRowsetNumber
  Description: The index number for the additional data result sets
  Value: 2

Name: SQ01_ResponseXML
  Description: This is the XML Response data from the RSAQ_REMOTE_QUERY_CALL JCO call.
  Value: <<XML>>

Name: SQ01_xMII_OUTPUT
  Description: The SQ01 result set in xMII Document format
  Value: <<XML>>

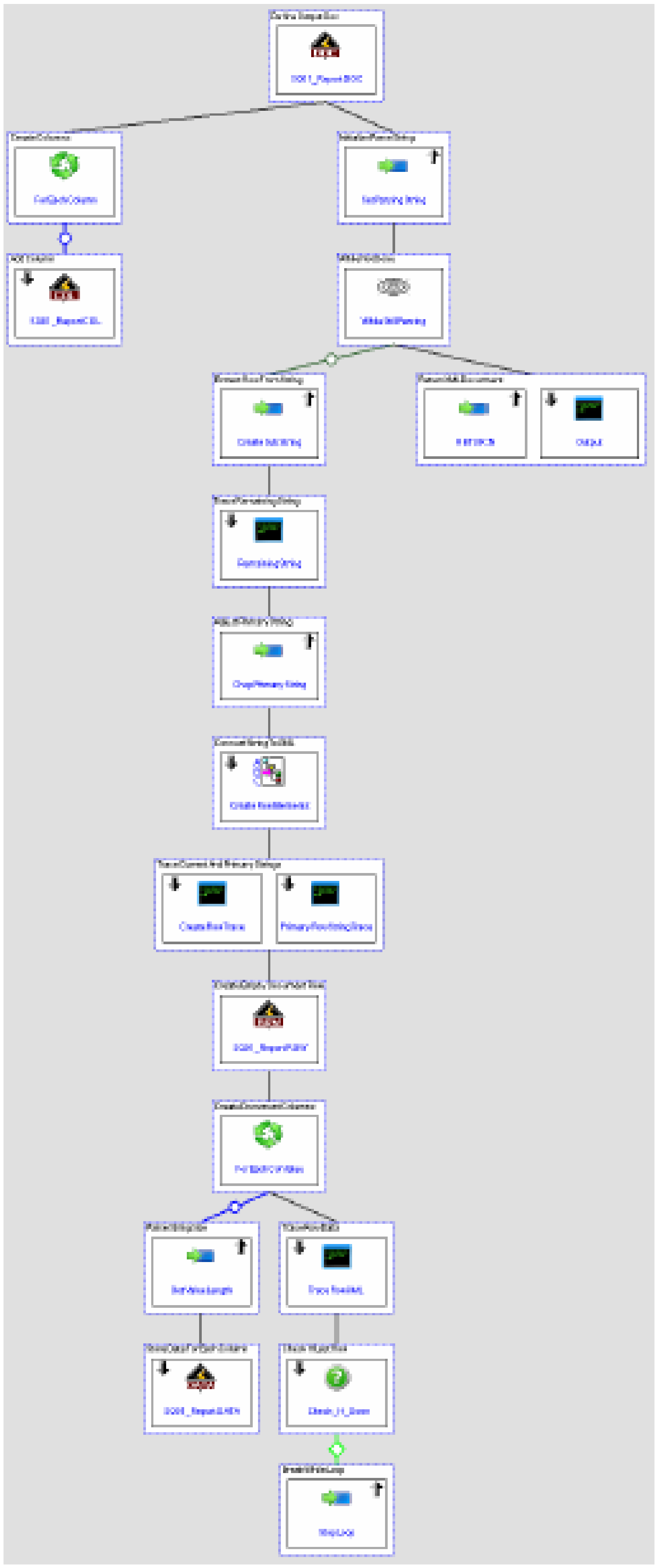


Local Properties

Name: ColumnValueLength
  Description: This is the value of the size of the DATA element string value for the current column
  Value: 0

Name: CurrentRowString
  Description: This string holds the data for the row currently being processed
  Value:

Name: PrimaryRowString
  Description: This string holds the raw data that will build the rows of the response document (Commas represent Column Delineation and SemiColons represent Row Delineation)
  Value:



SQ01_ReportDOC - IlluminatorDocument

Description: Define the return document structure.

Inputs:

ColumnSpecifications	
StartDate	2006-08-09T14:31:14
EndDate	2006-08-09T14:31:14
Output	<<XML>>

ForEachColumn - Repeater

Description: Repeat on each column value in the input XML, /RSAQ_REMOTE_QUERY_CALL/TABLES/LISTDESC/item.

Inputs:

Break	false
CurrentItem	0
ItemCount	0
Source	Transaction.SQ01_ResponseXML{/RSAQ_REMOTE_QUERY_CALL/TABLES/LISTDESC/item}
Output	<<XML>>

SQ01_ReportCOL - IlluminatorColumn

Description: Based off of the input XML FDESC value define a column for the XML report document for data to be stored against.

Incoming Links:

Type: Assign

From: ForEachColumn.Output{/item/FNAMENEW}

To: SQ01_ReportCOL.Description

Type: Assign

From: xmlencodename(ForEachColumn.Output{/item/FDESC})

To: SQ01_ReportCOL.Name

Inputs:

MinRange	0
MaxRange	100
IlluminatorDocument	SQ01_ReportDOC.Output
SQLDataType	1
Name	SQ01_COL
Description	

SetParsingString - Assignment

Description: Set the Local property Primary Data String to the value of the desired Line data for parsing and manipulation.

Outgoing Links:

Type: Assign

From: Transaction.SQ01_ResponseXML{/RSAQ_REMOTE_QUERY_CALL/TABLES/LDATA/item[#Transaction.AdditionalRowsetNumber#]/LINE}

To: Local.PrimaryRowString

WhileStillParsing - WhileRepeater

Description: While not done processing all of the string data for the rows in the return document.

Inputs:

Break	false
CurrentItem	0
MaxIterations	9000

CreateSubString - Assignment

Description: Each row in the LINE data is semicolon delimited, so get the first semicolon segment and set the row data to a local property for parsing.

Outgoing Links:

Type: Assign

From: stringif(stringindexof(Local.PrimaryRowString , ";") < 1, Local.PrimaryRowString, stringleft(Local.PrimaryRowString, stringindexof(Local.PrimaryRowString , ";")))

To: Local.CurrentRowString

RemainingString - Tracer

Description: This will show the user what data is left after the current row data has been removed.

Incoming Links:

Type: Assign

From: "SemiColon Index: " & stringindexof(Local.PrimaryRowString , ";") & "   String Length Left: " & (stringlength(Local.PrimaryRowString) - stringindexof(Local.PrimaryRowString , ";"))

To: RemainingString.Message

Inputs:

Message	
Level	INFO

ChopPrimaryString - Assignment

Description: Remove the current row data from the primary string along with the semicolon delimiter.

Outgoing Links:

Type: Assign

From: stringif(stringindexof(Local.PrimaryRowString, ";") < 1, "/", stringpart(Local.PrimaryRowString, stringindexof(Local.PrimaryRowString , ";") + 1, (stringlength(Local.PrimaryRowString) - stringindexof(Local.PrimaryRowString , ";"))))

To: Local.PrimaryRowString

CreateRowElements - StringListToXml

Description: Since the row data is comma delimited the Stringlist to XML action block can be used to convert the data into XML format for easier reference.

Incoming Links:

Type: Assign

From: Local.CurrentRowString

To: CreateRowElements.Input

Inputs:

TrimWhitespace	false
StripQuotes	true
Input	
Output	<<XML>>

CreateRowTrace - Tracer

Description: Trace out the XML created by the string list to XML action block for error checking in the execution logger.

Incoming Links:

Type: Assign

From: CreateRowElements.Output

To: CreateRowTrace.Message

Inputs:

Message	
Level	INFO

PrimaryRowStringTrace - Tracer

Description: Trace out to the execution logger the remaining primary row string trace.

Incoming Links:

Type: Assign

From: "Primary Row String: " & Local.PrimaryRowString & "   Index: " & stringindexof(Local.PrimaryRowString, ";")

To: PrimaryRowStringTrace.Message

Inputs:

Message	
Level	INFO

SQ01_ReportROW - IlluminatorRow

Description: Add and empty row as a place holder for data values, this way data can be specified without knowing the column names.

Inputs:

IlluminatorDocument	SQ01_ReportDOC.Output

ForEachColValue - Repeater

Description: Repeat on each column to add data values into the newly defined row.

Inputs:

Break	false
CurrentItem	0
ItemCount	0
Source	CreateRowElements.Output{/Rowsets/Rowset/Row }
Output	<<XML>>

GetValueLength - Assignment

Description: Since the data is returned as: StringSize:Value
Use the size of the Value must be parsed and stored for reference.

Outgoing Links:

Type: Assign

From: number(stringleft(ForEachColValue.Output{/Row/Item}, stringindexof(ForEachColValue.Output{/Row/Item}, ":")))

To: Local.ColumnValueLength

SQ01_ReportDATA - IlluminatorDataItem

Description: Store the data value to the specified column based on the size of the data value and the desired column name.

Incoming Links:

Type: Assign

From: xmlencodename(Transaction.SQ01_ResponseXML{/RSAQ_REMOTE_QUERY_CALL/TABLES/LISTDESC/item[#ForEachColValue.CurrentItem#]/FDESC})

To: SQ01_ReportDATA.Name

Type: Assign

From: stringreplace(stringright(ForEachColValue.Output{/Row/Item}, Local.ColumnValueLength), ";", "")

To: SQ01_ReportDATA.Value

Inputs:

IlluminatorDocument	SQ01_ReportDOC.Output
Name	TEST
Value	SQ01

TraceRowXML - Tracer

Description: Output the XML data used to makup the row data from the current string parse.

Incoming Links:

Type: Assign

From: "After Row " & WhileStillParsing.CurrentItem & ": " & CreateRowElements.Output

To: TraceRowXML.Message

Inputs:

Message	
Level	INFO

Check_If_Done - Conditional

Description: Check to see if there is any data left in the primary string.

Incoming Links:

Type: Assign

From: if(stringindexof(Local.PrimaryRowString, ";") < 1, 1, 0)

To: Check_If_Done.Input1

Type: Assign

From: if(Local.PrimaryRowString == "/", 1, 0)

To: Check_If_Done.Input2

Inputs:

Input1	false
Input2	false
Output	false
InputCount	2
LogicalAnd	true

StopLoop - Assignment

Description: Check to see if there is any data left in the primary string. If no data is left then break out of the loop.

Outgoing Links:

Type: Assign

From: 1

To: WhileStillParsing.Break

RETURN - Assignment

Description: Return the constructed xMII rowset document to the calling transaction.

Outgoing Links:

Type: AssignXml

From: SQ01_ReportDOC.Output

To: Transaction.SQ01_xMII_OUTPUT

Type: Assign

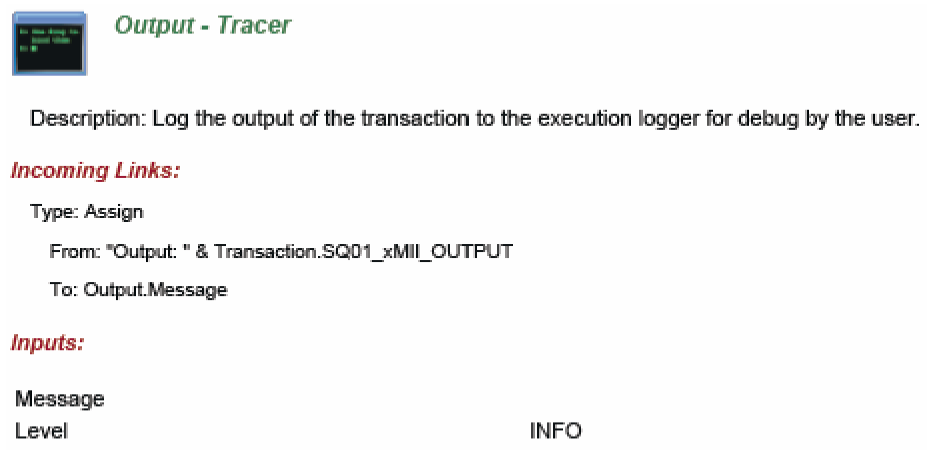
From: 1

To: WhileStillParsing.CurrentItem

Type: Assign

From: 0

To: WhileStillParsing.Break



Output - Tracer

Description: Log the output of the transaction to the execution logger for debug by the user.

Incoming Links:

Type: Assign

From: "Output: " & Transaction.SQ01_xMII_OUTPUT

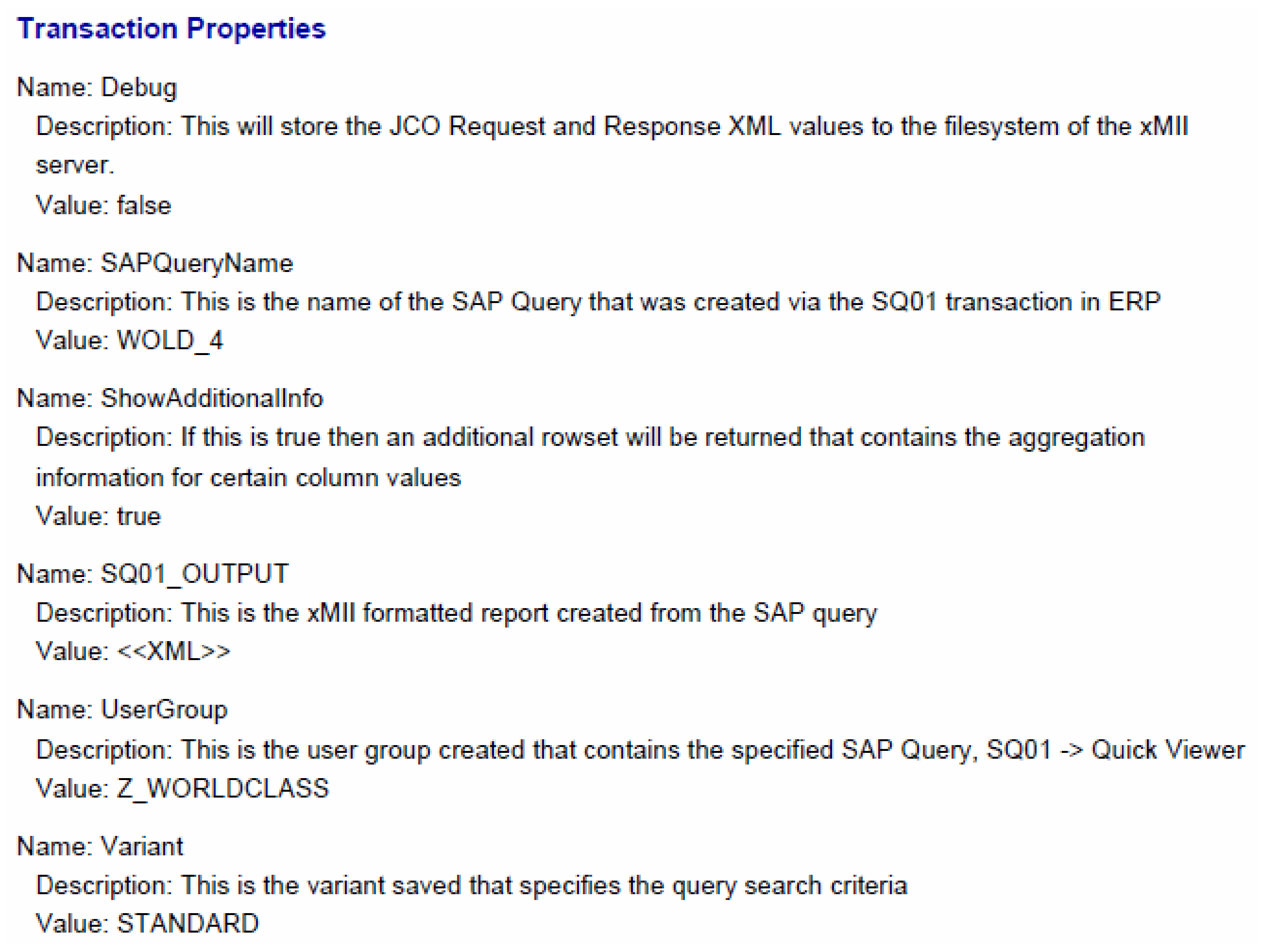
To: Output.Message

Inputs:

Message	
Level	INFO

SAP Query Report Transaction

Create the following transaction and save it as “ReportCreator”



Transaction Properties

Name: Debug
  Description: This will store the JCO Request and Response XML values to the filesystem of the xMII server.
  Value: false

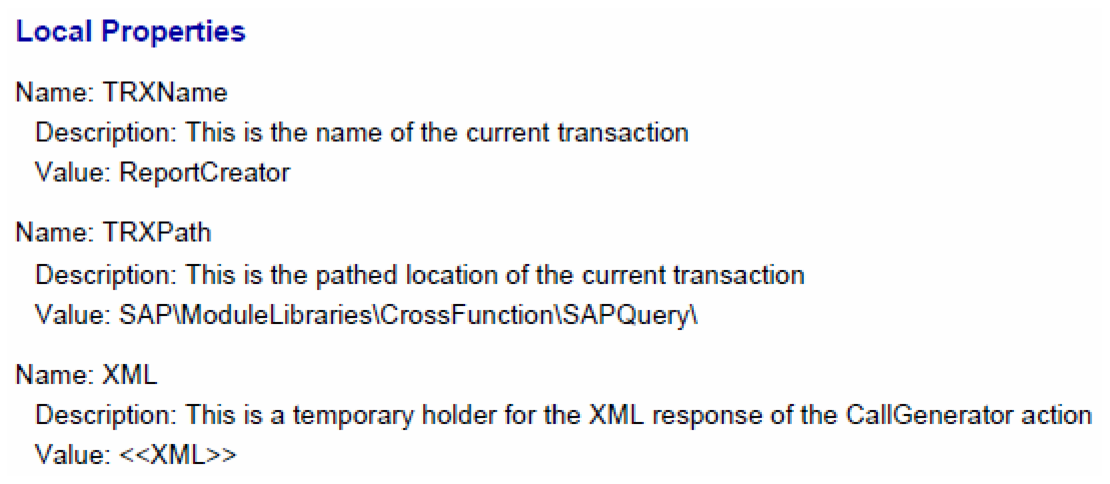
Name: SAPQueryName
  Description: This is the name of the SAP Query that was created via the SQ01 transaction in ERP
  Value: WOLD_4

Name: ShowAdditionalInfo
  Description: If this is true then an additional rowset will be returned that contains the aggregation information for certain column values
  Value: true

Name: SQ01_OUTPUT
  Description: This is the xMII formatted report created from the SAP query
  Value: <<XML>>

Name: UserGroup
  Description: This is the user group created that contains the specified SAP Query, SQ01 -> Quick Viewer
  Value: Z_WORLDCLASS

Name: Variant
  Description: This is the variant saved that specifies the query search criteria
  Value: STANDARD

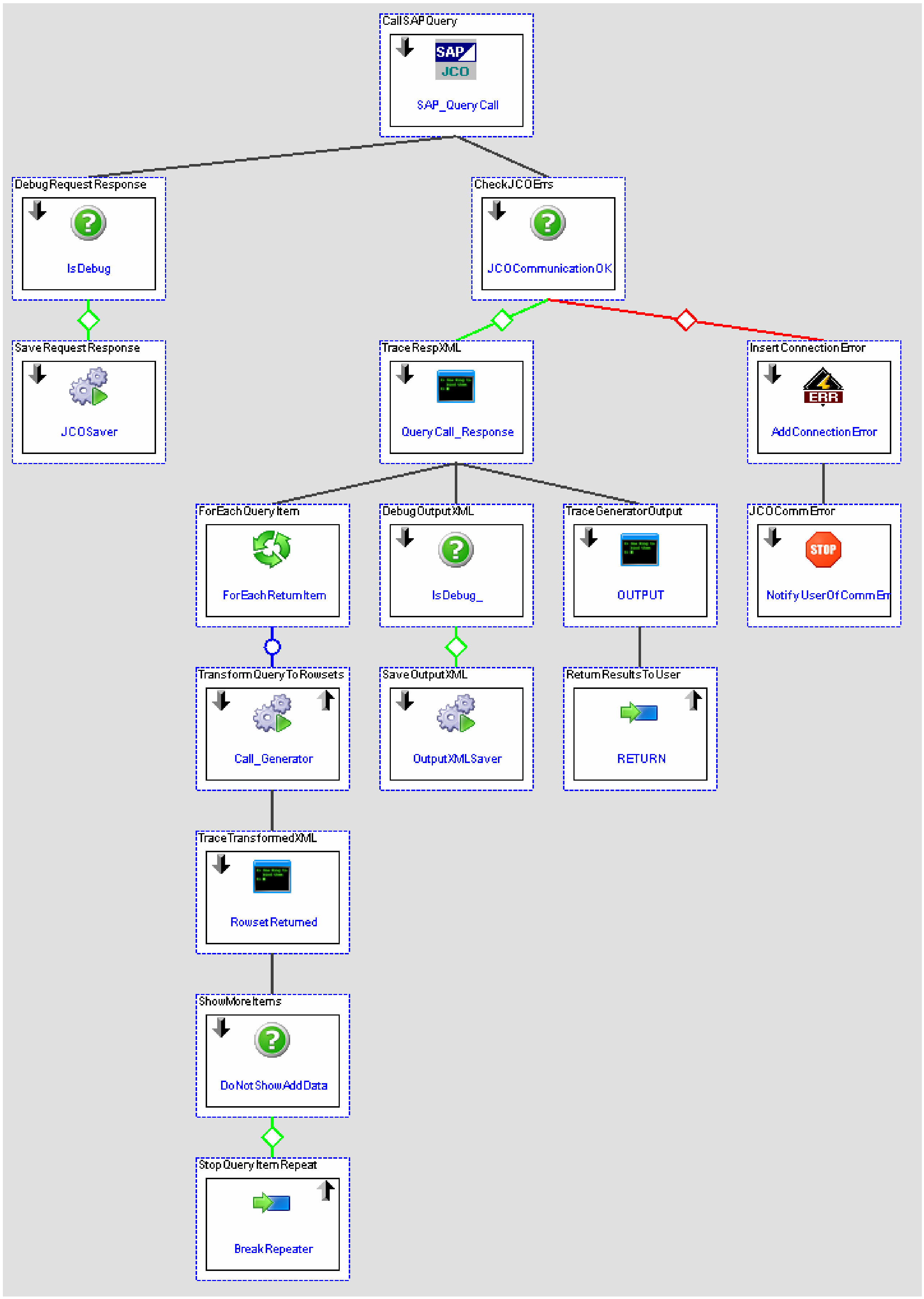


Local Properties

Name: TRXName
  Description: This is the name of the current transaction
  Value: ReportCreator

Name: TRXPath
  Description: This is the pathed location of the current transaction
  Value: SAP\ModuleLibraries\CrossFunction\SAPQuery\

Name: XML
  Description: This is a temporary holder for the XML response of the CallGenerator action
  Value: <<XML>>


CallSAPQuery
SAP_QueryCall
DebugRequestResponse
IsDebug
CheckJCOErrs
JCOCommunicationOK
SaveRequestResponse
JCOSaver
TraceRespXML
QueryCall_Response
InsertConnectionError
AddConnectionError
ForEachQueryItem
ForEachReturnItem
DebugOutputXML
IsDebug_
TraceGeneratorOutput
OUTPUT
JCOCommError
NotifyUserOfCommErr
TransformQueryToRowsets
Call_Generator
SaveOutputXML
OutputXMLSaver
ReturnResultsToUser
RETURN
TraceTransformedXML
RowsetReturned
ShowMoreItems
DoNotShowAddData
StopQueryItemRepeat
BreakRepeater

SAP_QueryCall - SAPJCOInterface

Description: Use JCO to connect to ERP and perform the RSAQ_REMOTE_QUERY_CALL call.

Incoming Links:

Type: Assign

From: Transaction.UserGroup

To: SAP_QueryCall.Request{/RSAQ_REMOTE_QUERY_CALL/INPUT/USERGROUP}

Type: Assign

From: Transaction.SAPQueryName

To: SAP_QueryCall.Request{/RSAQ_REMOTE_QUERY_CALL/INPUT/QUERY}

Type: Assign

From: Transaction.Variant

To: SAP_QueryCall.Request{/RSAQ_REMOTE_QUERY_CALL/INPUT/VARIANT}

Type: Assign

From: "X"

To: SAP_QueryCall.Request{/RSAQ_REMOTE_QUERY_CALL/INPUT/SKIP_SELSCREEN}

Type: Assign

From: "X"

To: SAP_QueryCall.Request{/RSAQ_REMOTE_QUERY_CALL/INPUT/EXTERNAL_PRESENTATION}

Type: Assign

From: "X"

To: SAP_QueryCall.Request{/RSAQ_REMOTE_QUERY_CALL/INPUT/DATA_TO_MEMORY}

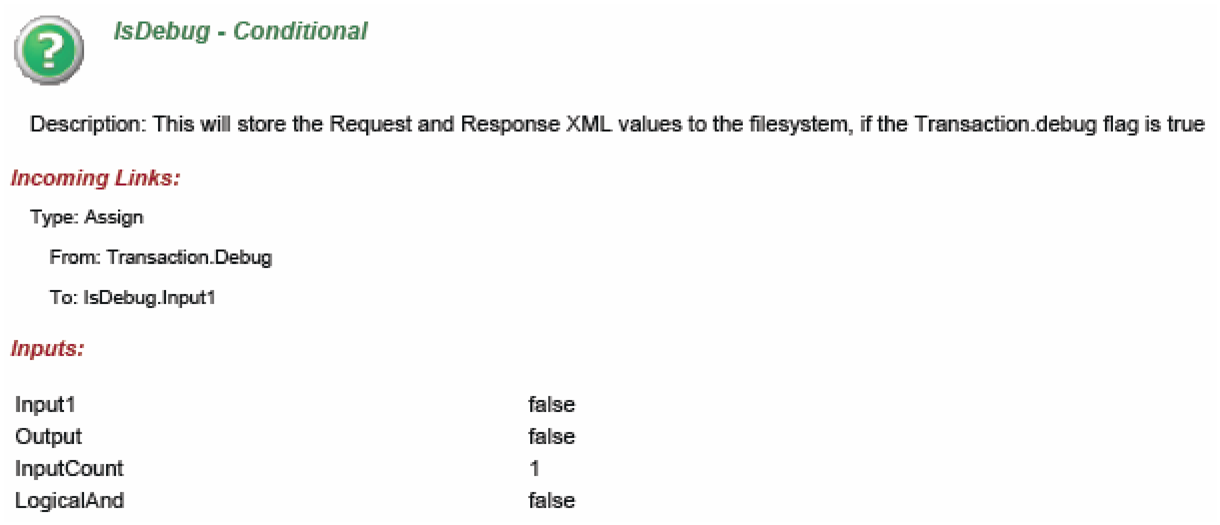
Type: Assign

From: stringif(Transaction.Variant == "", "9", "0")

To: SAP_QueryCall.Request{/RSAQ_REMOTE_QUERY_CALL/INPUT/DBACC}

Inputs:

Request	<<XML>>
Response	<<XML>>
AutoCommit	false
SAPRFC	RSAQ_REMOTE_QUERY_CALL
ExecuteFunction	true
Language	EN
SAPSystemAlias	
SAPServerName	
SAPClient	
SAPUserName	
SAPPassword	
SAPSSO2Ticket	
SAPSystemNumber	



IsDebug - Conditional

Description: This will store the Request and Response XML values to the filesystem, if the Transaction.debug flag is true

Incoming Links:

Type: Assign

From: Transaction.Debug

To: IsDebug.Input1

Inputs:

Input1	false
Output	false
InputCount	1
LogicalAnd	false

JCOSaver - Transaction

Description: Call a TRX that will store the Request and Response XML data to a file for easy reference and debug on the operation of the query.

Incoming Links:

Type: Assign

From: SAP_QueryCall.SAPRFC

To: JCOSaver.RFCName

Type: AssignXml

From: SAP_QueryCall.Response

To: JCOSaver.ResponseXML

Type: AssignXml

From: SAP_QueryCall.Request

To: JCOSaver.RequestXML

Type: Assign

From: "AllSAPQueries"

To: JCOSaver.SpecificName

Type: Assign

From: Local.TRXName

To: JCOSaver.TRXName

Type: Assign

From: Local.TRXPath

To: JCOSaver.TRXPath

Inputs:

ResetState	false
RFCName	BAPI_PRODORD_GET_LIST
RequestXML	<<XML>>
ResponseXML	<<XML>>
SpecificName	AllOrders
TRXName	
TRXPath	SAP\ModuleLibraries\PP\

JCOCommunicationOK - Conditional

Description: Will pass if the JCO Communication with ERP was successful. This means that the JCO action block successfully connected and performed the request. However the request itself still may fail even though the communication is successful.

Incoming Links:

Type: Assign

From: SAP_QueryCall.Success

To: JCOCommunicationOK.Input1

Inputs:

Input1	false
Output	false
InputCount	1
LogicalAnd	false

QueryCall_Response - Tracer

Description: Store the XML reponse from the JCO block to the execution logger so that the user can debug its operation.

Incoming Links:

Type: Assign

From: "Response (" & SAP_QueryCall.Response{count(/RSAQ_REMOTE_QUERY_CALL/TABLES/LDATA/item)} & " items): " & SAP_QueryCall.Response

To: QueryCall_Response.Message

Inputs:

Message	
Level	INFO

ForEachReturnItem - Repeater

Description: Repeat on each LINE node located at: /RSAQ_REMOTE_QUERY_CALL/TABLES/LDATA/item/LINE

Inputs:

Break	false
CurrentItem	0
ItemCount	0
Source	SAP_QueryCall.Response{/RSAQ_REMOTE_QUERY_CALL/TABLES/LDATA/item}
Output	<<XML>>

Call_Generator - Transaction

Description: SAP Query data conversion utility for creating an xMII Rowset from the queried data.

Incoming Links:

Type: AssignXml

From: SAP_QueryCall.Response

To: Call_Generator.SQ01_ResponseXML

Type: Assign

From: ForEachReturnItem.CurrentItem

To: Call_Generator.AdditionalRowsetNumber

Outgoing Links:

Type: AppendXml

From: Call_Generator.SQ01_xMII_OUTPUT{/Rowsets/Rowset}

To: Local.XML{/Rowsets}

Inputs:

ResetState	false
AdditionalRowsetNumber	2
SQ01_ResponseXML	<<XML>>

RowsetReturned - Tracer

Description: Trace out to the execution logger the converted query data for debug.

Incoming Links:

Type: Assign

From: ForEachReturnItem.CurrentItem & ") " & Call_Generator.SQ01_xMII_OUTPUT

To: RowsetReturned.Message

Inputs:

Message	
Level	INFO

DoNotShowAddData - Conditional

Description: Did the user specify to see the additional query data?

Incoming Links:

Type: Assign

From: !Transaction.ShowAdditionalInfo

To: DoNotShowAddData.Input1

Inputs:

Input1	false
Output	false
InputCount	1
LogicalAnd	false

BreakRepeater - Assignment

Description: Signal to the Repeater.ForEachReturnItem to break its loop. This will cause the transaction to return the data from the first query line only.

Outgoing Links:

Type: Assign

From: true

To: ForEachReturnItem.Break

IsDebug_ - Conditional

Description: If debug is enabled then store the Response XML from the JCO Block to the filesystem. This will allow for debug of the converted XML data.

Incoming Links:

Type: Assign

From: Transaction.Debug

To: IsDebug_.Input1

Inputs:

Input1	false
Output	false
InputCount	1
LogicalAnd	false

OutputXMLSaver - Transaction

Description: Save OutputXML document, this XML data is post conversion and will be the XML data returned to the user as the output of the transaction.

Incoming Links:

Type: Assign

From: SAP_QueryCall.SAPRFC

To: OutputXMLSaver.RFCName

Type: Assign

From: "AllQueries"

To: OutputXMLSaver.SpecificName

Type: Assign

From: Local.TRXName

To: OutputXMLSaver.TRXName

Type: Assign

From: Local.TRXPath

To: OutputXMLSaver.TRXPath

Type: AssignXml

From: Local.XML

To: OutputXMLSaver.OutputXML

Inputs:

ResetState	false
OutputXML	<<XML>>
RFCName	BAPI_PRODORD_GET_LIST
SpecificName	
TRXName	ProdOrdList
TRXPath	SAP\ModuleLibraries\PP\

OUTPUT - Tracer

Description: Display the resulting XML document from the Query Conversion, this is the XML data that is returned to the Query Template.

Incoming Links:

Type: Assign

From: "OUTPUT: " & Local.XML

To: OUTPUT.Message

Inputs:

Message	
Level	INFO

RETURN - Assignment

Description: Return the converted SAP Query data as the output XML data for the transaction.

Outgoing Links:

Type: AssignXml

From: Local.XML

To: Transaction.SQ01_OUTPUT

AddConnectionError - IlluminatorFatalError

Description: Add the error message returned from the JCO API to the document to be returned to the user.

Incoming Links:

Type: Assign

From: SAP_QueryCall.LastErrorMessage

To: AddConnectionError.Message

Outgoing Links:

Type: AssignXml

From: Local.XML

To: Transaction.SQ01_OUTPUT

Inputs:

IlluminatorDocument	Local.XML
Message	Connection Error

NotifyUserOfCommErr - TerminateTransaction

Description: The error message returned from the JCO API to the user and stop the execution of the transaction.

Incoming Links:

Type: Assign

From: SAP_QueryCall.LastErrorMessage

To: NotifyUserOfCommErr.TerminationMessage

Inputs:

TerminationMessage	Transaction Execution Terminated Upon Request
TerminateWithError	true

SAP Query Web Page

The web page shown in Image 1 in the xMII & ERP Query Architecture section of this document can be created using the following HTML & JavaScript Code:

```
<html>
<head>
<title>SAP Query Report Creator</title>
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<link type="text/css" rel="stylesheet" href="/SAP/Common/CSS/SAP_BP.css" />
<script type="text/javascript" language="JavaScript">
/*
        PURPOSE:
        Provides a functional example for interfacing with the SAP Queries for the purpose of
        generating web reports.  This page allows for a standard interface to report directly
        from data in the ERP system, similar to the SQ01 Transaction Page.
*/

// Apply filters to the desired browser query
function ApplyBrowserFilter(myApplet, strGroup, strInfoSet, strQuery) {
        var myQuery = myApplet.getQueryObject();
        myQuery.setParam(1,strGroup);
        myQuery.setParam(2,strInfoSet);
        myQuery.setParam(3,strQuery);
        myApplet.updateBrowser(true);
}

// Filters the InfoSet, Query, and Variant browsers
function GroupBrowser_Selected() {
        var strGroup = document.GroupBrowser.getBrowserObject().getSelectedDatalinkValue();
        var strInfoSet = "*";
        var strQuery = "*";
        ApplyBrowserFilter(document.InfoSetBrowser, strGroup, strInfoSet, strQuery);
        ApplyBrowserFilter(document.QueryBrowser, strGroup, strInfoSet, strQuery);
        ApplyBrowserFilter(document.VariantBrowser, strGroup, strInfoSet, strQuery);
}

// Filters the Query and Variant browsers
function InfoSetBrowser_Selected() {
        var strGroup = document.GroupBrowser.getBrowserObject().getSelectedDatalinkValue();
        var strInfoSet = document.InfoSetBrowser.getBrowserObject().getSelectedDatalinkValue();
        var strQuery = "*";
        ApplyBrowserFilter(document.QueryBrowser, strGroup, strInfoSet, strQuery);
        ApplyBrowserFilter(document.VariantBrowser, strGroup, strInfoSet, strQuery);
}

// Filters the Variant browser
function QueryBrowser_Selected() {
        var strGroup = document.GroupBrowser.getBrowserObject().getSelectedDatalinkValue();
        var strInfoSet = document.InfoSetBrowser.getBrowserObject().getSelectedDatalinkValue();
        var strQuery = document.QueryBrowser.getBrowserObject().getSelectedDatalinkValue();
        ApplyBrowserFilter(document.VariantBrowser, strGroup, strInfoSet, strQuery);
}
```

```
// Performs the selected query with or without a variant
function runQuery() {
	var strGroup  = document.GroupBrowser.getBrowserObject().getSelectedDatalinkValue();
	var strInfoSet =
document.QueryBrowser.encodeURLItem(document.InfoSetBrowser.getBrowserObject().getSelectedDatalin
kValue());
	var strQuery = document.QueryBrowser.getBrowserObject().getSelectedDatalinkValue();

	if (strGroup != "*") {
		if (strQuery != "*") {
			strGroup = document.QueryBrowser.encodeURLItem(strGroup);
			strQuery = document.QueryBrowser.encodeURLItem(strQuery);
			var selVariant =
document.VariantBrowser.getBrowserObject().getSelectedDatalinkValue();
			if (selVariant != "") {
				selVariant = document.QueryBrowser.encodeURLItem(selVariant);

			}
			if (document.frmMain.chkShowAdd.checked) {
				showAdd = "true";
			} else {
				showAdd = "false";
			}
			var strContentType = document.getElementById("txtContentType").value;

			// Build the URL & Show the report
			var strURL =
"/Lighthammer/Illuminator?QueryTemplate=SAP/ModuleLibraries/CrossFunction/SAPQuery/ReportQuery";
			strURL += "&Param.1=" + strGroup + "&Param.2=" + strQuery + "&Param.3=" +
selVariant + "&Param.4=" + showAdd;
			strURL += "&RowCount=100&Content-Type=" + strContentType;
			alert(strURL);
			window.open(strURL,"QueryResults","");
		} else {
			alert("Please Select a Query!");
		}
	} else {
		alert("Please Select a Group!");
	}
}
</script>
</head>

<body>
<form name="frmMain">
	<table border="1" cellpadding="2" cellspacing="0" class="SAPTable">
		<tr>
			<td colspan="2" class="PageSmHeader" align="center">
				SAP Query (SQ01)
			</td>
		</tr>
		<tr>
			<td class="tdSubSmHeader" align="right">
```

```
                        Group:
                </td>
                <td class="tdSubSmHeader">
                        <APPLET NAME="GroupBrowser" WIDTH="450" HEIGHT="19"
CODE="iBrowser" CODEBASE="/Illuminator/Classes" ARCHIVE="illum8.zip" MAYSCRIPT>
                        <PARAM NAME="QueryTemplate"
VALUE="SAP/ModuleLibraries/CrossFunction/SAPQuery/GroupCatalogQuery">
                        <PARAM NAME="DisplayTemplate"
VALUE="SAP/Common/DropDownDatalinkBrowser">
                        <PARAM NAME="DisplayColumns" VALUE="Item,NUM">
                        <PARAM NAME="DefaultItem" VALUE="--SELECT--">
                        <PARAM NAME="DefaultItemDatalink" VALUE="*">
                        <PARAM NAME="RowCount" VALUE="250">
                        <PARAM NAME="SelectionEvent" VALUE="GroupBrowser_Selected">
                        </APPLET>
                </td>
        </tr>
        <tr>
                <td class="tdSubSmHeader" align="right">
                        InfoSet:
                </td>
                <td class="tdSubSmHeader">
                        <APPLET NAME="InfoSetBrowser" WIDTH="450" HEIGHT="19"
CODE="iBrowser" CODEBASE="/Illuminator/Classes" ARCHIVE="illum8.zip" MAYSCRIPT>
                        <PARAM NAME="QueryTemplate"
VALUE="SAP/ModuleLibraries/CrossFunction/SAPQuery/InfoSetCatalogQuery">
                        <PARAM NAME="DisplayTemplate"
VALUE="SAP/Common/DropDownDatalinkBrowser">
                        <PARAM NAME="DefaultItem" VALUE="--SELECT--">
                        <PARAM NAME="DefaultItemDatalink" VALUE="*">
                        <PARAM NAME="DisplayColumns" VALUE="Item,CLAS">
                        <PARAM NAME="InitialUpdate" VALUE="false">
                        <PARAM NAME="SelectionEvent" VALUE="InfoSetBrowser_Selected">
                        </APPLET>
                </td>
        </tr>
        <tr>
                <td class="tdSubSmHeader" align="right">
                        Query:
                </td>
                <td class="tdSubSmHeader">
                        <APPLET NAME="QueryBrowser" WIDTH="450" HEIGHT="19"
CODE="iBrowser" CODEBASE="/Illuminator/Classes" ARCHIVE="illum8.zip" MAYSCRIPT>
                        <PARAM NAME="QueryTemplate"
VALUE="SAP/ModuleLibraries/CrossFunction/SAPQuery/QueryCatalogQuery">
                        <PARAM NAME="DisplayTemplate"
VALUE="SAP/Common/DropDownDatalinkBrowser">
                        <PARAM NAME="DefaultItem" VALUE="--SELECT--">
                        <PARAM NAME="DefaultItemDatalink" VALUE="*">
                        <PARAM NAME="DisplayColumns" VALUE="Item,QUERY">
                        <PARAM NAME="InitialUpdate" VALUE="false">
                        <PARAM NAME="SelectionEvent" VALUE="QueryBrowser_Selected">
                        </APPLET>
                </td>
```

```
		</tr>
		<tr>
			<td class="tdSubSmHeader" align="right">
				Variant:
			</td>
			<td class="tdSubSmHeader">
				<APPLET NAME="VariantBrowser" WIDTH="450" HEIGHT="19"
CODE="iBrowser" CODEBASE="/Illuminator/Classes" ARCHIVE="illum8.zip" MAYSCRIPT>
				<PARAM NAME="QueryTemplate"
VALUE="SAP/ModuleLibraries/CrossFunction/SAPQuery/VariantCatalogQuery">
				<PARAM NAME="DisplayTemplate"
VALUE="SAP/Common/DropDownDatalinkBrowser">
				<PARAM NAME="DefaultItem" VALUE="--SELECT--">
				<PARAM NAME="DefaultItemDatalink" VALUE="">
				<PARAM NAME="DisplayColumns" VALUE="Item,VARIANT">
				<PARAM NAME="InitialUpdate" VALUE="false">
				</APPLET>
			</td>
		</tr>
		<tr>
			<td colspan="2" class="tdSubSmHeader" align="center">
				<table border="0" width="95%">
					<tr>
						<td class="tdSubSmHeader" align="right">
							Generate the Report As:
						</td>
						<td align="left" class="tdSubSmHeader">
							<select size="1" id="txtContentType"
name="txtContentType">
								<option value="text/html">html</option>
								<option value="text/xml">xml</option>
								<option value="text/csv">csv</option>
							</select>
						</td>
						<td align="left" class="tdSubSmHeader">
							<input type="checkbox" name="chkShowAdd"
id="chkShowAdd" value="true">Show Additional Data
						</td>
						<td align="left" class="tdSubSmHeader">
							<img src="/SAP/Common/images/s_b_exec.gif"
border="1" width="16" height="16" onclick="runQuery();" alt="Run Query" style="cursor:hand;border-
style:outset;">
						</td>
					</tr>
				</table>
			</td>
		</tr>
	</table>
</form>
</body>
</html>
```

Related Content

Reference 1: How to Create a JCO BAPI/RFC Itemset to xMII Rowset Conversion Utility

Goto: http://sdn.sap.com

Click On: xAPPs

Search For: How to xMII Itemset Rowset Conversion

Reference 2: xMII Best Practices Guide

Goto: http://sdn.sap.com

Click On: xAPPs

Search For: xMII Best Practices

Copyright