Novell exteNd Composer™
CICS RPC Connect

5.0

USER'S GUIDE

Novell®

Legal Notices

Novell, Inc.

1800 South Novell Place

Provo, UT 85606

www.novell.com

exteNd Composer CICS-RPC Connect ***User's Guide***

January 2004

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

eDirectory is a trademark of Novell, Inc.
exteNd is a trademark of Novell, Inc.
exteNd Composer is a trademark of Novell, Inc.
exteNd Director is a trademark of Novell, Inc.
jBroker is a trademark of Novell, Inc.
NetWare is a registered trademark of Novell, Inc.
Novell is a registered trademark of Novell, Inc.

SilverStream Trademarks

SilverStream is a registered trademark of SilverStream Software, LLC.

Third-Party Trademarks

All third-party trademarks are the property of their respective owners.

Third-Party Software Legal Notices

Jakarta-Regexp Copyright ©1999 The Apache Software Foundation. All rights reserved. Xalan Copyright ©1999 The Apache Software Foundation. All rights reserved. Xerces Copyright ©1999-2000 The Apache Software Foundation. All rights reserved. Jakarta-Regexp , Xalan and Xerces software is licensed by The Apache Software Foundation and redistribution and use of Jakarta-Regexp, Xalan and Xerces in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notices, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "The Jakarta Project", "Jakarta-Regexp", "Xerces", "Xalan" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache" nor may "Apache" appear in their name, without prior written permission of The Apache Software Foundation. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright ©1996-2000 Autonomy, Inc.

Copyright ©2000 Brett McLaughlin & Jason Hunter. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution. 3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@jdom.org. 4. Products derived from this software may

not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management (pm@jdom.org). THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This Software is derived in part from the SSLava™ Toolkit, which is Copyright ©1996-1998 by Phaos Technology Corporation. All Rights Reserved. Customer is prohibited from accessing the functionality of the Phaos software.

The code of this project is released under a BSD-like license [license.txt]: Copyright 2000-2002 (C) Intalio Inc. All Rights Reserved. Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain copyright statements and notices. Redistributions must also contain a copy of this document. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The name "ExoLab" must not be used to endorse or promote products derived from this Software without prior written permission of Intalio Inc. For written permission, please contact info@exolab.org. 4. Products derived from this Software may not be called "Castor" nor may "Castor" appear in their names without prior written permission of Intalio Inc. Exolab, Castor, and Intalio are trademarks of Intalio Inc. 5. Due credit should be given to the ExoLab Project (http://www.exolab.org/). THIS SOFTWARE IS PROVIDED BY INTALIO AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE DISCLAIMED. IN NO EVENT SHALL INTALIO OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contents

About This Guide

Purpose

This guide describes how to use the CICS RPC Component Editor, which is the design-time portion of the exteNd Composer CICS RPC Connect.

Audience

This book is for systems analysts, programmers, and others who intend to build applications or services that require a CICS-facing component.

Prerequisites

This book assumes prior familiarity with the exteNd Composer design-time environment and Composer application-building metaphors. You should also be familiar with CICS RPC concepts.

Additional documentation

For the complete set of Novell exteNd Director documentation, see the Novell Documentation Web Site (http://www.novell.com/documentation-index/index.jsp).

1 Welcome to exteNd Composer and CICS RPC

Before You Begin

Welcome to the *CICS RPC Connect Guide*. This Guide is a companion to the *exteNd Composer User's Guide*, which details how to use all the features of Composer, except the Connect Component Editors. So, if you haven't looked at the Composer User's Guide yet, please familiarize yourself with it before using this Guide.

Novell exteNd Composer provides separate Component Editors for each Connect, like CICS RPC. The special features of each component editor are described in separate Guides like this one.

If you have been using exteNd Composer, and are familiar with the core component editor, the XML Map Component Editor, then this Guide should get you started with the CICS RPC Component Editor.

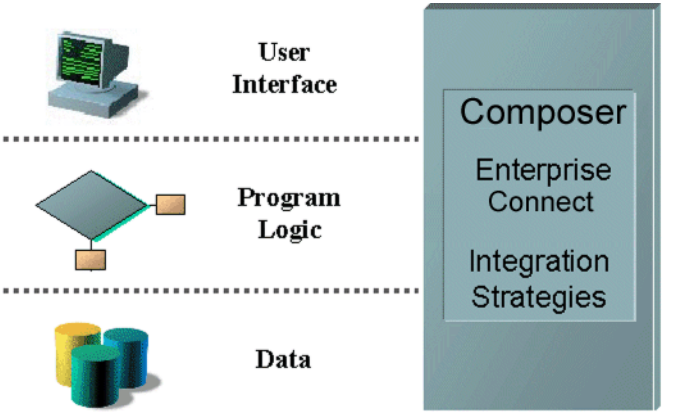
Before you begin working with the CICS RPC Connect, you must have it installed into your existing exteNd Composer. Likewise, before you can run any Services built with this Connection the Composer Enterprise Server environment, you must have already installed the Server software for this Connect into Novell exteNd Enterprise Server.

NOTE: To be successful with this Component Editor, you must be familiar with the basics of CICS and the COBOL programming language as well as the programs with which you wish to interface.

About CICS RPC Connect

Novell exteNd Composer is built upon a simple hub and spoke architecture. The hub is a robust XML transformation engine that accepts requests via XML documents, performs transformation processes on those documents, and interfaces with XML enabled applications which returns a XML response document. The spokes or Connects are plug-in modules that “XML enable” sources of data that are not XML aware, bringing their data into the hub for processing as XML. These data sources can be anything from legacy COBOL applications to Message Queues to HTML pages.

CICS RPC Connect can be categorized by the integration strategy each one employs to XML enable an information source. The integration strategies are a reflection of the major divisions used in modern systems designs for Internet-based computing architectures. Depending on your B2B needs, and architecture of your legacy applications, exteNd can integrate your business systems at the User Interface, Logic, or Data Levels.



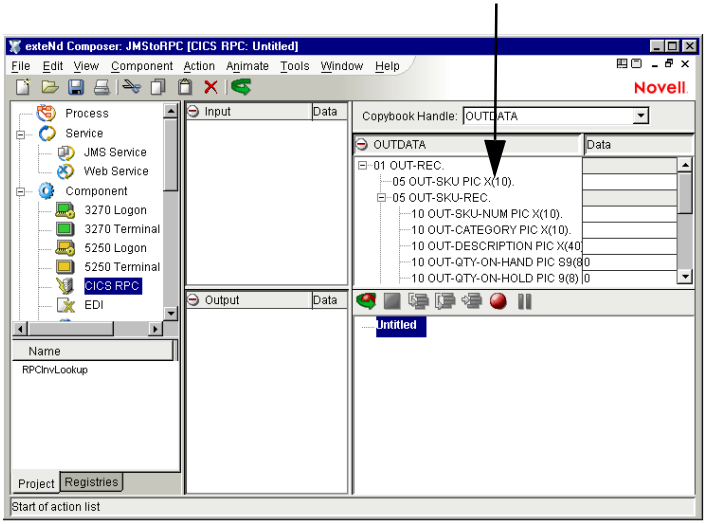
What is CICS RPC?

CICS RPC is an acronym for Customer Information Control System - Remote Procedure Call. The CICS RPC Connect XML enables legacy system data using the Logic integration strategy. The CICS RPC component provides the ability to interface with CICS managed programs using CICS’ External Call Interface (ECI) and exchange data directly with the programs running in a CICS Region using the standard communication area commonly referred to as the DFHCOMMAREA. Novell exteNd relies on being provided a representation of the communication area as is often defined in applications via a copybook.

The Novell exteNd CICS RPC Connect was designed specifically to work in the CICS environment through IBM's CICS Java Gateway. The Java gateway can reside on the same platform as the CICS region or on a separate platform depending on your system architecture. CICS is an IBM transaction-processing environment that often resides on an IBM mainframe using IBM's proprietary EBCDIC character-encoding scheme, but can also be run under IBM's TXSeries product line on other platforms that support other character encoding schemes.

CICS RPC Connect uses copybook representation of the DFHCOMMAREA as the data interface presented to the XML transformation hub. These copybooks appear in the native environment pane of the CICS RPC Component Editor. Once there, the data can be converted, transformed, or transferred the same as any other XML document.

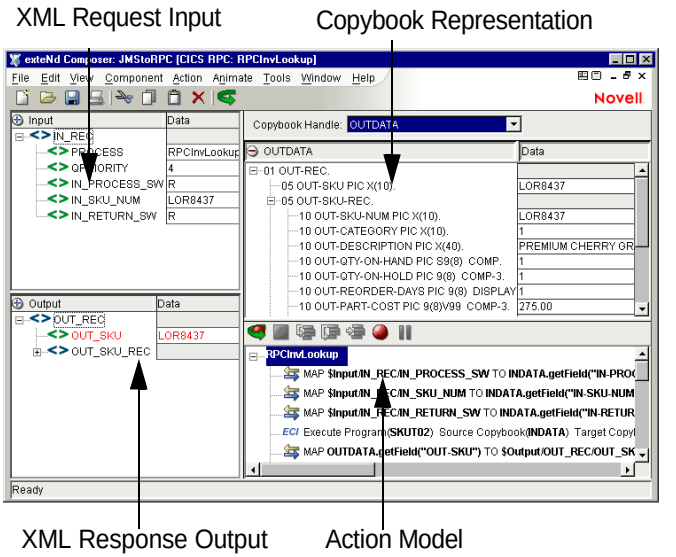
RPC Copybook Interface appears in the Native Environment pane



About exteNd's CICS RPC Component

Much like the XML Map component, the CICS RPC component is designed to map, transform, and transfer data between two different XML templates (i.e., request and response XML documents). However, it is specialized to make a connection to a CICS program via ECI, process the data using elements from a DOM, and then map the results to an output DOM. You can then act upon the output DOM in any way that makes sense for your organization, including displaying the output to a web site. In essence, you're able to push XML data into and/or pull data as XML from a legacy system without ever having to alter the legacy system itself. A CICS RPC Component can perform simple data manipulations, such as mapping and transferring data from one XML document to another, or from an XML document to a program. It can also perform sophisticated manipulations, such as performing multi row mapping (in reference to an OCCURS clause). The CICS RPC Component can also perform all the actions available in a XML Map Component including; process XSL, send mail, and post and receive XML documents using the HTTP protocol.

The following illustration shows how a CICS RPC component uses an ECI connection to interact with data on the mainframe.



What Can You Build Using the CICS RPC Component Editor?

The CICS RPC Component Editor allows you to extend any XML integration application you are building (refer to the *exteNd Composer User's Guide*) to include any CICS business application that supports ECI interactions. For example, you may have an application that retrieves the description, picture, price and inventory status of a product from the database(s) that are updated on a scheduled basis and displays in a Web browser. By using the CICS RPC Component Editor, you can now get the current product information from the operational systems and the static information (e.g., the picture) from the database(s) and merge the information from these separate sources before it displays to a user. This provides the same current information to both internal and external users.

2 Getting Started with the CICS RPC Component Editor

The Sample Transaction

Throughout this guide a prototypical COBOL program called SKUT02 will be used to illustrate the most interesting features of the CICS RPC Connect. SKUT02 is designed to perform read-only operations that query an Inventory system. A query is defined with three parameters. The first is a process switch identifying the type of operation to be performed (READ, UPDATE, etc.) For our purposes, this switch will always be set to R for READ. The second parameter is the specific SKU or product ID to obtain information about it. The third parameter is a switch specifying the information returned from the inquiry to be either a text message or an inventory record. The message will indicate that the item exists or doesn't exist. The record will contain data or be returned blank.

➢ **Steps commonly used to create a new CICS RPC Component**

1. Obtain or create copies of the host program DFHCOMMAREA definitions as copybooks and make them accessible from your local machine.
2. Create a CICS RPC Component.
3. Create an Input XML Template and Sample Documents (or you can optionally use the Auto Map feature to do this for you).
4. Create Map Actions moving data from the Input DOM into the Copybook (or you can optionally use the Auto Map feature to do this for you).
5. Create an Execute ECI Action.
6. Repeat steps 3 and 4 for Output.
7. Test the Component.

Creating a CICS RPC Connection Resource

Before you create a CICS RPC Component, you need to create a Connection Resource to access the mainframe program.

About Connection Resources

When you create a Connection Resource for the CICS RPC Connect, select ECI CICS RPC Connection. The ECI CICS RPC Connection is used when you want to connect to a CICS TCP protocol application. This connection allows you to get through the TCP/IP port on the host to the CICS gateway.

Performance Parameters and Connection Resources

When using the CICS Transaction Gateway with exteNd Composer, you want to be sure to tune the gateway for the applications you will be using. Doing this requires knowledge of how many simultaneous users are expected to be using the Composer service. With this knowledge there are two gateway parameters which can be set to help tune the connections. The parameters are: *initworker* and *initconnection.*

Initworker represents the initialized number of worker threads. *Initconnection* represents the initialized number of connections. These two parameters can be set as command line parameters on startup of the gateway or in the ctg.ini file for the 3.1.x version of the gateway or in the gateway.properties file for the 3.0 version of the gateway.

As mentioned above, knowledge of the workload for the application is needed to set these parameters appropriately. For example, if you are expecting 50 concurrent users for the application, then you would set both the *initworker* and *initconnection* to 50. The examples below demonstrate how to set the parameters via a command line in the ini files.

To set them as command line parameters, use the switches -initconnect=50 -initworker=50 on startup of the gateway.

For version 3.1.x of the gateway, the *initconnect* and *initworker* parameters are set in the ctg.ini file. Below is a sample of the SECTION GATEWAY from the ctg.ini file. showing the init lines.

```
SECTION GATEWAY
        closetimeout=20000
        ecigenericreplies=on
        initconnect=50
        initworker=50
        maxconnect=100
        maxworker=100
        noinput=off
        nonames=off
        notime=off
        tfile=ctg.trc
        workertimeout=20000

        protocol@tcp.handler=com.ibm.ctg.server.TCPHandler

    protocol@tcp.parameters=connecttimeout=5000;idletimeout=600000;pingfrequency=600
    00;port=2006;solinger=0;sotimeout=1000;
    ENDSECTION
```

For version 3.0 of the gateway, the *initconnect* and *initworker* are set in the Gateway.properties file. Below is a sample of the Gateway.properties files showing the init lines. It is recommended that you upgrade to the version 3.1 since IBM is dropping support for version 3.0

```
# Initial number of ConnectionManager threads
# initconnect=50

# Maximum number of ConnectionManager threads
# maxconnect=100

# Initial number of Worker threads
# initworker=50

# Maximum number of Worker threads
# maxworker=100
```

About Constant and Expression Driven Connections

You can specify Connection parameter values in one of two ways: as Constants or as Expressions. A constant based parameter uses the value you enter in the Connection dialog every time the Connection is used. An expression based parameter allows you to set the value using a programmatic expression, which can result in a different value each time the connection is used at runtime. This allows the Connection's behavior to be flexible and vary based on runtime conditions each time it is used.

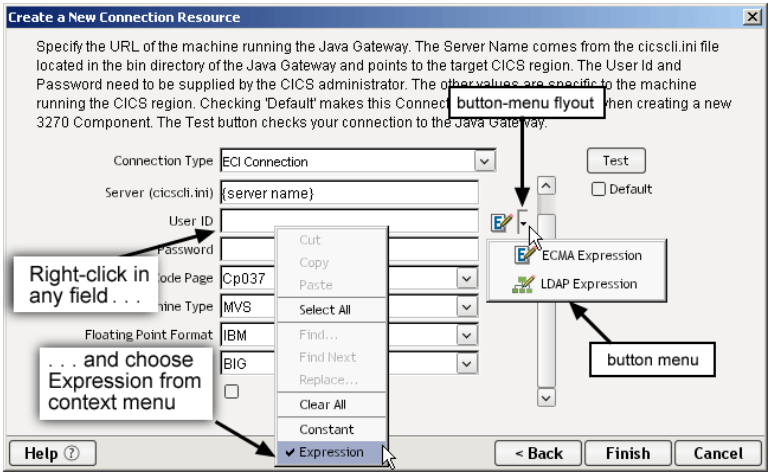
For instance, one very simple use of an expression driven parameter in a ECI CICS Connection would be to define the User ID and Password as PROJECT Variables (e.g. PROJECT.XPath("USERCONFIG/MyDeployUser"). This way when you deploy the project, you can update the PROJECT Variables in the Deployment Wizard to values appropriate for the final deployment environment. At the other extreme, you could have a custom script that queries a Java business object in the Application Server to determine what User ID and Password to use.

➢ **To switch a parameter from Constant driven to Expression driven:**

1 Click the right mouse button in the parameter field you are interested in changing.

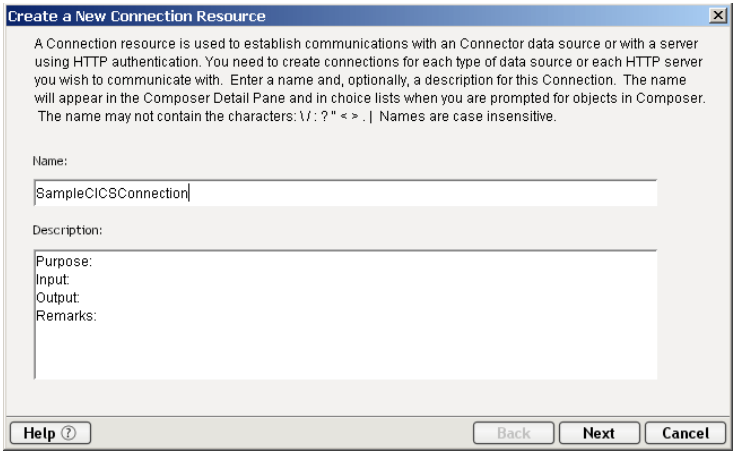
2 Select **Expression** from the context menu (as shown below) and the editor button will appear or become enabled.

3 Click on the button and then create an ECMAScript expression that will evaluate to a valid parameter value at runtime.



➢ **To Create a CICS RPC Connection Resource:**

1 From the **File** menu in exteNd Composer, select **New > xObject,** then open the **Resource** tab and select **Connection**. The Create a New Connection Resource Wizard appears.



NOTE: Alternately, you can highlight **Connection** in the Composer window category pane, click the right mouse button, then select **New**.

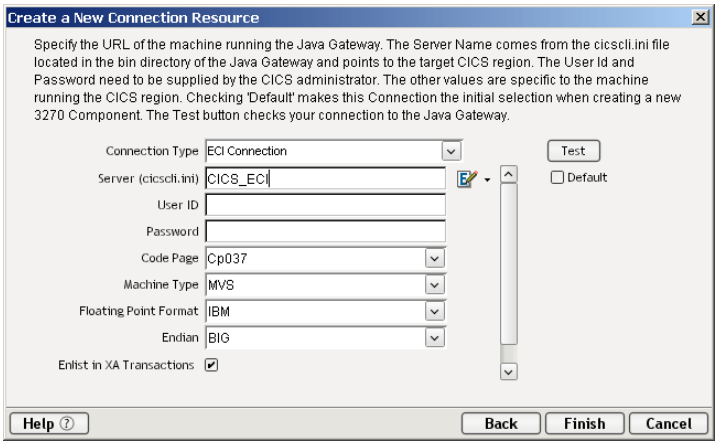
2 Enter a **Name** for the connection object.

3 Optionally, enter **Description** text.

4 Click **Next**.

5 Select **ECI Connection** from the **Connection Type** drop down menu.

NOTE: Use the **ECI Connection** type if you are connecting to a CICS application.



6 In the **Server (cicscli.ini)** field, enter the server name, **CICS_ECI** in our example.

7 In the **User ID** and **Password** fields, your system administrator will provide you with this name and password, which is defined in a separate client initialization file.

8 In the **Code Page** field, from the drop down menu, select a code page dependent upon the remote machine operating system character data standard.(i.e. CP037 for EBCDIC or 8859_1 for ASCII).

9 In the **Machine Type** field, from the drop down menu, select the target platform of your CICS Region/Server (MVS, OS2, NT, AIX).

10 In the **Floating Point Format** field, from the drop down menu, select a name dependent upon the machine type selected: IBM or IEEE .

11 In the **Endian** field, from the drop down menu, select the order of the most/least significant bytes in integers (BIG if the most significant byte precedes the least significant byte in memory, otherwise select LITTLE).

12 If a component that uses this connection will be running under XA transaction control, check the **Enlist in XA transactions** checkbox.

NOTE: This option has no effect at design time. It affects runtime behavior only.

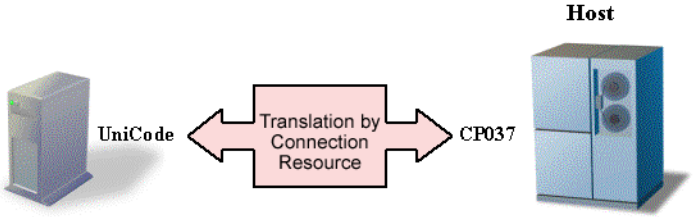
13 Optionally click the **Test** button to verify that the connection is valid. (A live connection will be attempted.)

14 Optionally click on the **Default** checkbox if you wish the connection to be saved as a default connection setting for CICS RPC Components.

15 Click **Finish** to save your connection.

About Code Page Support

Code Page support in exteNd Composer's Connection Resources allows you to specify which character encoding scheme to use when translating characters sent between exteNd Composer and other host systems. Novell exteNd uses Unicode character encoding (the Java and XML standard). Existing legacy and other host systems use a variety of character encoding schemes (e.g., Code Pages) specific to their language or usage. Therefore, a mechanism is needed to translate the character encoding between these systems if they are to communicate with each other. This is handled in exteNd Composer by specifying the Code Page used by a host system in the Connection Resource used to access that system. Refer to Appendix E for list of Code Pages supported by Java.



Creating XML Templates for Your Component

Unique to CICS RPC, you can automatically create XML templates and sample documents based on the copybook(s) you associate with the CICS RPC Component through the Auto Map selection in the Component Menu. This feature should be used especially when the information from the application can be used by multiple services or the CICS program is a multi use program (e.g., a program that performs all types of information access: inquiry, update, insert and delete).

However, you can create your own XML templates prior to designing your component. See Chapter 5, *Creating XML Templates* in the *exteNd Composer User's Guide* for more information. Use this option when the component uses only a portion of data available in the Copybook.

Also, if your component design calls for any other xObject resources such as custom scripts or code table maps, it is best to create these before creating the CICS RPC Component. For more information, see the *exteNd Composer User's Guide*.

3 Creating a CICS RPC Component

Before Creating a CICS RPC Component

As with all Novell exteNd components, the first step in creating a CICS RPC component is to specify the XML templates needed. For more information, see *Creating a New XML Template* in the *Composer User's Guide.*

If you decide to use the Auto Map feature to create your XML templates and sample documents, then you will use the {System} and {ANY} choices for the Input and Output templates when initially creating the CICS RPC component.

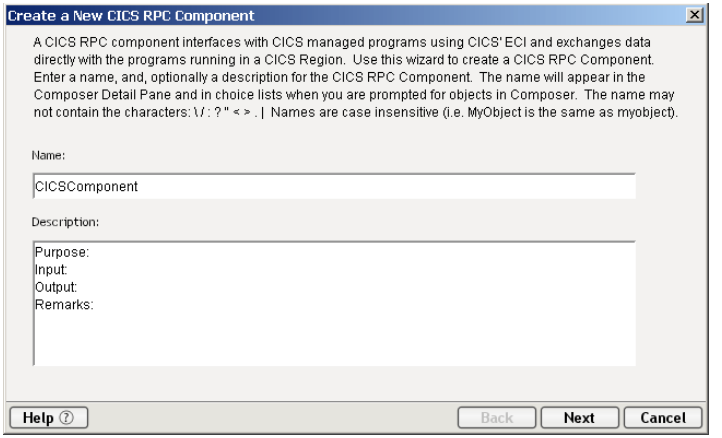
Also, as part of the process of creating a CICS RPC component, you can select a CICS RPC connection or you can create a new one. If you create the connection beforehand, then it is available to all new CICS RPC components as a selection.

To create a new CICS RPC Component:

1. Select **File>New > xObject**, then open the **Component** tab and select **CICS RPC** in exteNd Composer.

   NOTE: Alternatively, under **Component** in the Composer window category pane you can highlight **CICS RPC**, click the right mouse button, then select **New**.

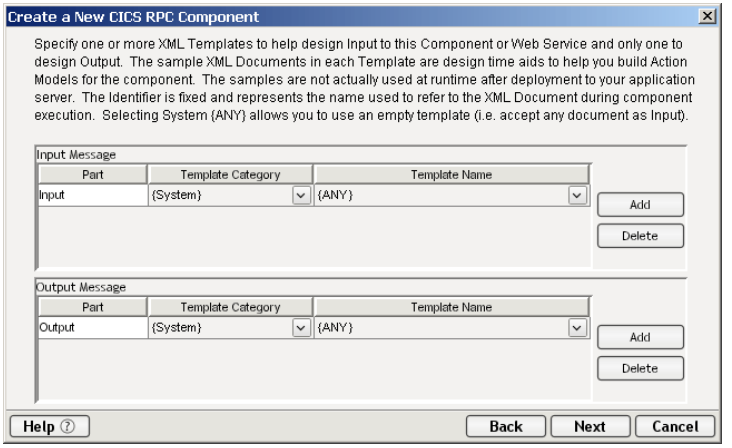
2. The Create a New CICS RPC Component Wizard appears.



3 Enter a **Name** for the new CICS RPC Component.

4 Optionally, enter **Description** text.

5 Click **Next**. The XML Input/Output Property Info panel of the New CICS RPC Component Wizard appears.



6 Specify the Input template as follows.

7 Specify the Input and Output templates as follows.

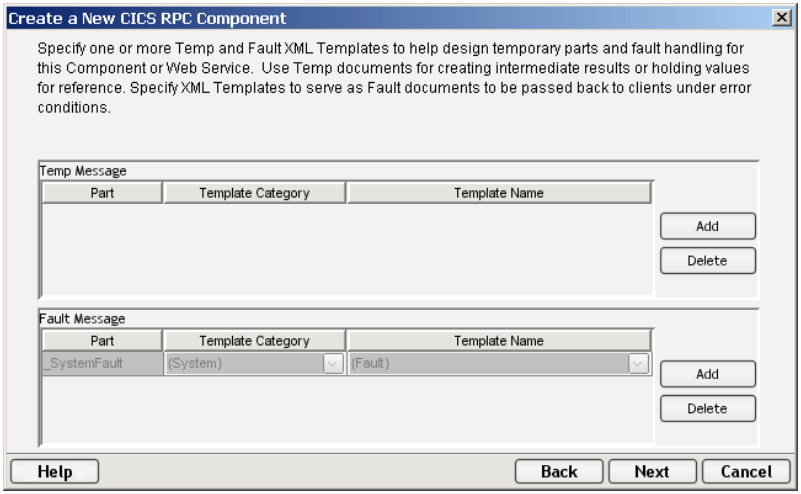
- If you don't plan to use the Auto Map feature, type in a name for the template under **Part** if you wish the name to appear in the XML Editor as something other than "Input".
- If you don't plan to use the Auto Map feature, select a **Template Category** if it is different than the default category.

- If you don't plan to use the Auto Map feature, select a **Template Name** from the list of XML templates in the selected **Template Category**.
- To add additional input XML templates, click **Add** and choose a **Template Category** and **Template Name** for each.
- To remove an input XML template, select an entry and click **Delete**.

8 Select an XML template for use as an Output Part using the same steps outlined above.

NOTE: For most new CICS RPC components, you will specify an input or output XML template that contains no structure by selecting {System}{ANY} as the input and output template and then use the auto-map tool. For more information, see "Creating an Output Part without Using a Template" in the User's Guide.

9 Click **Next**. The Temp and Fault XML Template panel appears.

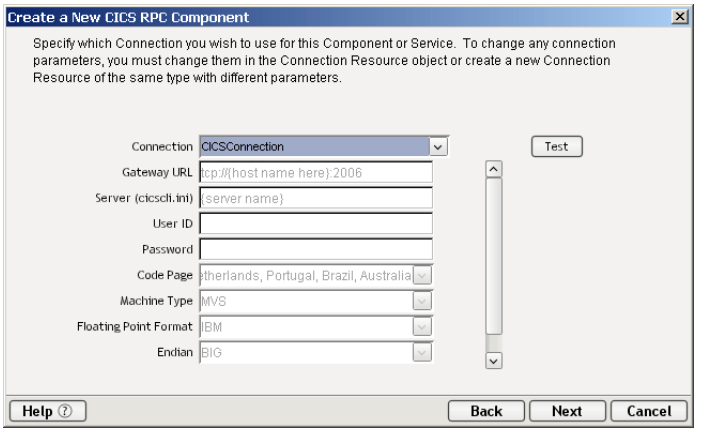


10 If desired, specify a template to be used as a scratchpad under the "Temp Message" pane of the dialog window. This can be useful if you need a place to hold values that will only be used temporarily during the execution of your component or are for reference only. Select a Template Category if it is different than the default category. Then select a Template Name from the list of XML templates in the selected Template Category.

11 Under the "Fault Message" pane, select an XML template to be used to pass back to clients when an error condition occurs.

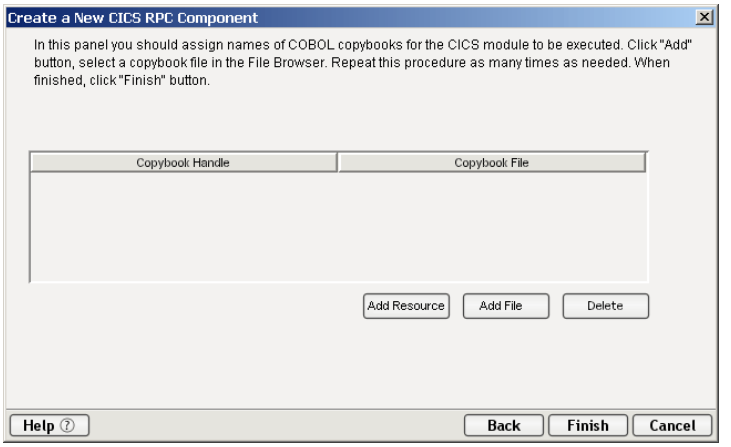
12 As above, to add additional input XML templates, click **Add** and choose a Template Category and Template Name for each. Repeat as many times as desired. To *remove* an input XML template, select an entry and click **Delete**.

13 Click **Next.** The Connection Info panel of the Create a New CICS RPC Component wizard appears.



14 Select a **Connection** type from the pull down list.

15 Click **Next**. The Copybook Info panel of the Create a New CICS RPC Component wizard appears.

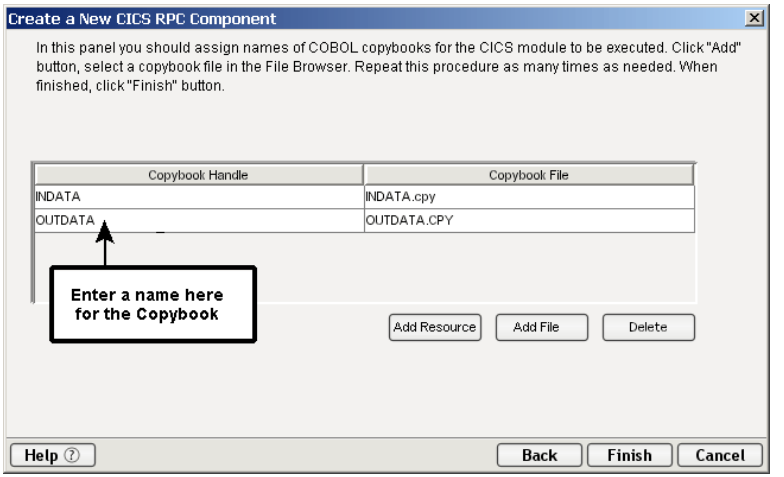


16 Select a Copybook by clicking on **Add Resource** or **Add File**.

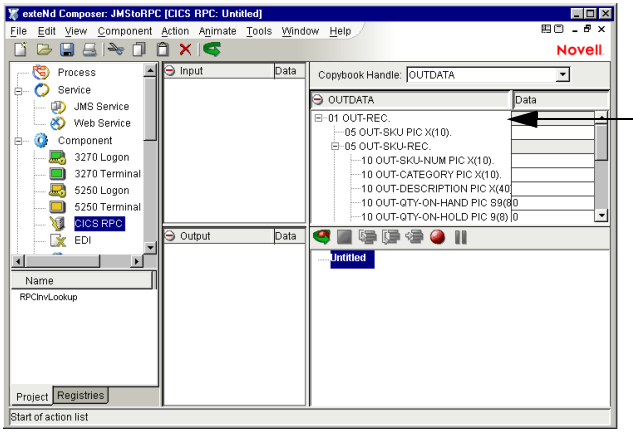
- Click on **Add Resource** to select a previously created Copybook Resource (see Chapter 9 in the *Composer User's Guide* for instructions on how to add a Copybook Resource).
- Click on **Add File** to browse and select a COBOL copybook for the program to be executed.

- By default, the Copybook file name is assigned as its handle inside the component. You may optionally assign other names to the copybooks by typing over the names in the Copybook Handle column.
- To remove a file, select that entry and click **Delete**.

NOTE: The CICS RPC Component Editor is designed to handle both multiple ECI requests and/or host programs that use a single copybook as the interface for both Input and Output for the program or require different input and output definitions. All these styles are completely compatible with Composer.



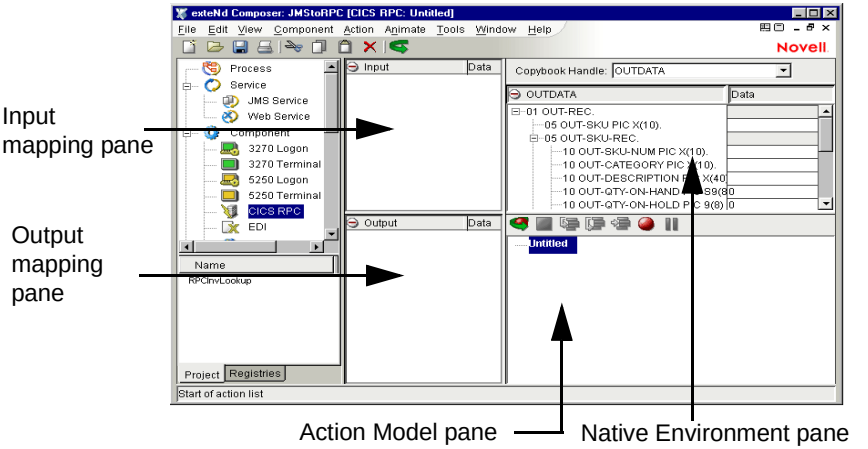
17 Click **Finish**. The component is created and the CICS RPC Component Editor appears.



About the CICS RPC Component Editor Window

The CICS RPC Component Editor includes all the functionality of the XML Map Component Editor. It contains mapping panes for Input and Output XML documents as well as an Action Model pane.

The difference, however, is that the CICS RPC Component Editor also includes a Native Environment pane common to all Connects. It is then populated with the initial environment pane, which is specific to the CICS RPC Connect.



About the CICS RPC Native Environment Pane

The CICS RPC Native Environment pane simulates an actual COBOL Copybook. From this pane, you can perform the following:

- Take data from an Input XML document (or other available DOMs) and use it as input for a program you plan to execute by dragging the data into the proper copybook field.
- By clicking the right mouse button in the native environment pane, a list of Menu Options can be accessed. The View selection has three different options for looking at the data. You can select Copybook view or “Hex Data” in addition to the “Copybook + Data” (default). See the *exteNd Composer User’s Guide* to learn more about *Menus and Toolbars*.

NOTE: Data can not be edited in the this pane as it can be done in the Input and Output DOMs.

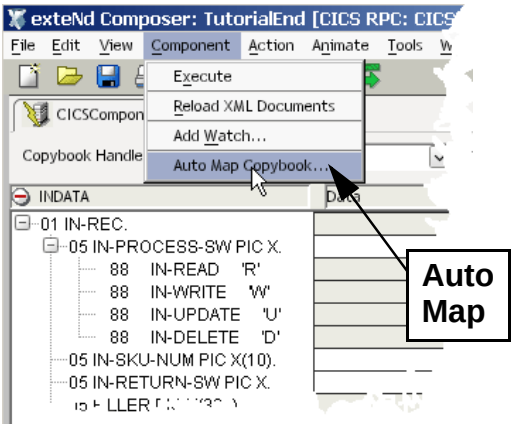
NOTE: In cases where a Copybook is designed without a single root data descriptor (e.g., contains a few 02 level items and no 01 level items), Composer will create a 01 level node at the top of the Copybook data tree displayed in the Native Environment Pane.

About CICS RPC Specific Menu Bar Items

The CICS RPC Connect includes actions that are not included with Composer. One of these actions is Auto Map Copybook. This action allows you to create Input/Output Documents and Mapping Actions automatically.

Auto Map Copybook

From the Menu Bar, select **Component/Auto Map Copybook**. This option allows you to automatically create Input/Output Documents and mapping actions to move data to/from the Documents and Copybooks. Optionally, it allows you to create an XML template based on the generated documents and Auto Map. Refer to the following sections to learn more about Creating Input Documents and Output Documents from a copybook.



About CICS RPC Specific Context Menu Items

A context menu is available whenever you right-click inside the Native Environment Pane. Some commands are available only when you are working with certain line items within this pane. The table below will explain each item on the menu and when they can be enabled.

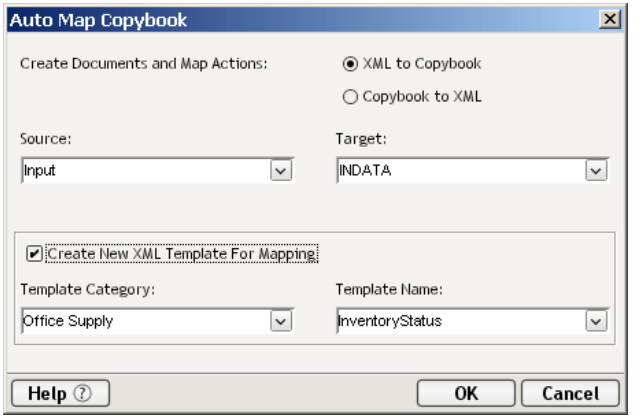
Native Pane Menu	Description
Decision	Enabled when a REDEFINE Statement in the copybook is highlighted. When you click on Decision, a dialog box appears prompting you to enter a Decision expression that determines when to use the Redefined data descriptors.
Repeat	Enabled when an OCCURS statement in the copybook is highlighted. When you click on Repeat, a dialog box appears prompting you to enter information specifying the Target for the Repeat action.
Map	Active in the input or output pane. When you highlight a statement and click on Map, a dialog box appears prompting you to enter information.
View - Copybook	Active in the native environment pane. When you highlight a statement and click on View/Copybook, the pane will display the copybook.
View - Copybook + Data	Active in the native environment pane and is set as the default. When you highlight a statement and click on View/Copybook + Data, the pane will reflect the copybook and any data mapped into the copybook or placed there as output from executing the program.
View Hex Data	Active in the native environment pane. When you highlight a statement and click on View/Hex Data, the pane will display hexadecimal representation of data mapped into the copybook or placed there as an output from executing the program in Hexadecimal format.
Select Occurrence	Enabled when an OCCURS statement in the Copybook is highlighted. Since each occurrence of the data in fields contained in the OCCURS clause is not displayed, when you click on Select Occurrence, a dialog box appears prompting you to select which occurrence of data you wish to see for these fields. Enter a number starting at 1.

Native Pane Menu	Description
Expand Tree	Displays all Copybook nodes beneath the selected data descriptor.
Collapse Tree	Hides Copybook nodes beneath the selected data descriptor.
Copy	Enabled when in the View/Hex Data format. Allows you to highlight a block of text and copy it then paste.
Print Copybook	Allows you to print the copybook.
Find	Allows you to perform a search within the copybook in all data views.
Find next	Allows you to search for the next occurrence of text entered in “Find” dialog in all data views.

Creating Input Sample Documents from a Copybook with Auto Map

Auto Map Copybook allows you to create Input and Output XML sample Documents for data based on the Copybook interface to the host program. It will also create mapping actions between the sample Documents and Copybook(s). Optionally, it allows you to create an XML template for the sample documents and be saved permanently, in the event you want to reuse it.

➢ **To create a new Input Sample Document:**



1 Select **Component>Auto Map Copybook** from exteNd Composer. The Automap Copybook dialog box appears.

2 Selecting **XML to Copybook** indicates that you wish to create an Input document and Map Actions to move data into the Copybook.

3 Select the **Source (**Input**)** from the pull down list. The source will contain Partss available in the component (e.g., Input, Output etc.)

4 Select the **Target (**Copybook name**)** from the drop down list. The target list will contain Copybooks available in the component.

5 Clicking the checkbox to **Create New XML Template For Mapping** indicates that you wish to create a permanent XML Template with a sample document for use with this and additional components you create. By enabling this feature, the Template Category and Template Name will be activated.

6 Select a **Template Category** if it is different than the default category or enter a New Template Category Name over an existing entry.

7 Select a **Template Name** from the list of XML templates or enter a New Template Name over an existing entry.

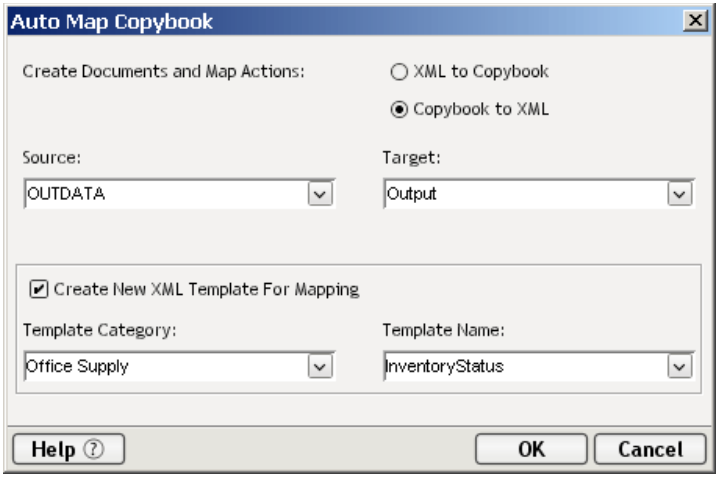
8 Click **OK.** The Component Editor Window appears with the Input Template created and the Map Actions added to the Action Model.

NOTE: If your host program uses only a single copybook for both Input and Output, you can still use the Auto Map feature, but you may wish to delete some of the actions that were created automatically and are not relevant for Input.

Creating Output Sample Documents from a Copybook with Auto Map

This screen allows you to create Output Documents and mapping actions to move data to/from the Copybook(s). Optionally, it allows you to create an XML template based on the generated documents and Auto Map Copybooks.

NOTE: These actions are generally performed after an “Execute ECI” action as described in the next chapter.



➢ **To create a new Output Sample Document:**

1. Select **Component>Auto Map Copybook** from exteNd Composer. The Automap Copybook dialog box appears.
2. Selecting **Copybook to XML** indicates that you want to create an Output document and Map Actions to move host program output data from the Copybook to the Output document.
3. Select the **Source (**Copybook) from the drop down list. The source drop-down list will contain a list of Copybooks.
4. Select the **Target (**Output**)** from the pull down list. The target list will contain Parts available in the component.
5. Clicking the checkbox to **Create New XML Template For Mapping** indicates that you wish to create a permanent XML Template with a sample document for use with this and additional components you create. By enabling this feature, the Template Category and Template Name will be activated.
6. Select a **Template Category** if it is different than the default category or enter a New Template Category Name over an existing entry.
7. Select a **Template Name** from the list of XML templates or enter a New Template Name over an existing entry.
8. Click **OK.** The Component Editor Window appears with the Output Template created and Map Actions added to the Action Model.

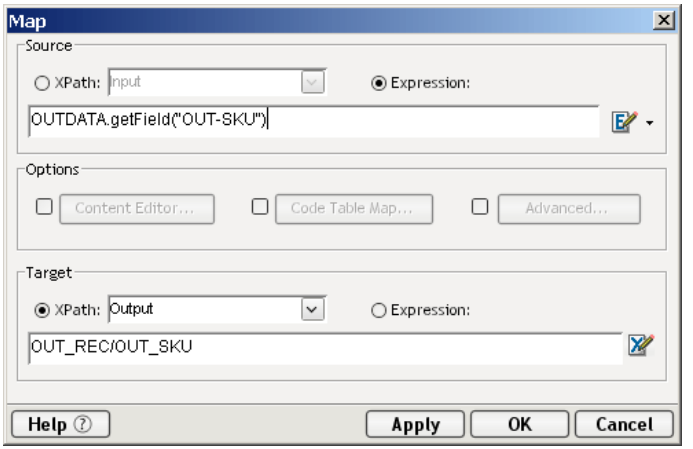
NOTE: If your host program uses only a single copybook for both Input and Output, you can still use the Auto Map feature, but you may wish to delete some of the actions that were created automatically and are not relevant for Output.

Creating an Input and Output XML Template Manually

You may want to create a sample document(s) manually if you are going to build a component that uses only a portion of the data available from the DFHCOMMAREA that is described by the copybook. The interaction of the Native Pane is then the same as with all other component editors. You can choose to use this technique for either the Input and Output documents or both as required by your application.

➢ **To create a XML Template Manually:**

1. Locate an existing XML document or use an external XML editor to create your sample document.
2. Create a new XML template and add the document(s) you created as a sample(s).
3. Create the component as before, but instead of selecting **{System}** and **{Any}** for the Input and/or Output template(s), select the name of the one you created.
4. Add the same Copybooks as before.
5. When the Component Window appears, simply drag and drop elements from the Input DOM to the Copybook and Map actions appear automatically. The following screen displays how the map screen will appear if you use the automatic map feature via the drag and drop.



NOTE: For more information, see *Creating a New XML Template* in the *Composer User's Guide.*

4 Performing CICS RPC Actions

About Actions

An *action* is similar to a programming statement in that it takes input in the form of parameters and performs specific tasks. Please see the chapters in the *Composer User's Guide* devoted to Actions.

Within the CICS RPC Component Editor, a set of instructions for processing XML documents or communicating with non-XML data sources is created as part of an *Action Model*. The Action Model performs all data mapping, data transformation, data transfer between mainframe and XML documents, and data transfer within components and services.

An Action Model is made up of a list of actions. All actions within an Action Model work together. As an example, one Action Model might read invoice data from a disk, retrieve data from a mainframe inventory database, map the result to a temporary XML document, make a conversion, and map the converted data to an output XML document.

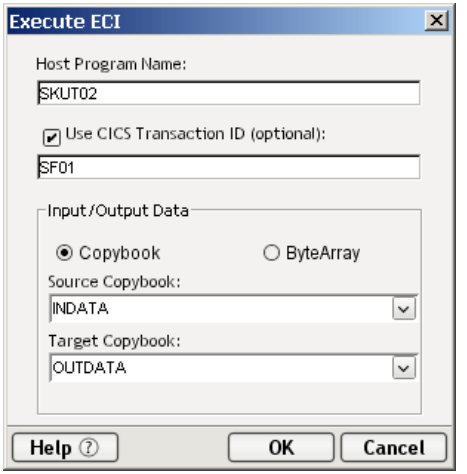
The Action Model mentioned above would be composed of several actions. These actions would:

- Open an invoice document and perform a CICS RPC command to retrieve invoice data from a mainframe database
- Map the result to a temporary XML document
- Convert a numeric code using a Code Table and map the result to an Output XML document.

About CICS RPC Specific Actions

There is an Action unique to CICS RPC called "Execute ECI." This action is used to make an external call by converting copybook data into a ByteArray and sending it to the mainframe, or by creating the ByteArray directly.

To access this action from the Menu Bar, click **Action Menu/New Action/Execute ECI**. The following dialog box appears:



The use of this dialog is as follows.

➢ **To define an Execute ECI Action:**

1 Enter the **Host Program Name**. In the example SKUT02 is the program name

2 Click inside the checkbox to activate field entry for a CICS Transaction ID. This field is optional, and only needs to be filled in if the system and/or your CICS environment requires a Transaction ID as well as a Host Program name.

NOTE: A typical Transaction ID has been entered in the above screen. If you have clicked the checkbox to enter a Transaction ID but fail to enter that information in the field provided, you will not be able to Execute ECI.

3 In the Input/Output Data section, select **Copybook** or **ByteArray**. Choose Copybook if you wish to convert Copybook data into a ByteArray and send it to the mainframe, receiving a ByteArray back from the mainframe. Choose ByteArray if you wish to create the ByteArray directly using the Convert actions and absorb and emit it directly.

4 Select **Source Copybook** (or **Source ByteArray**) from the pull down list box. In the example INDATA is the Source Copybook. If you select a ByteArray, you will have access to the ECMAScript Expression Builder.

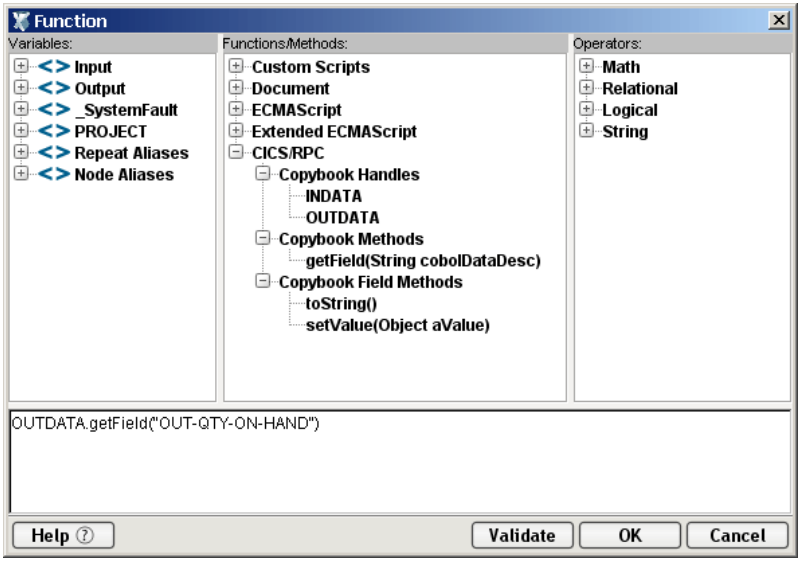
5 Select **Target Copybook** (or **Target ByteArray**) from the pull down list box. In the example OUTDATA is the Target Copybook. the pull down list box. In the example INDATA is the Source Copybook. If you select a ByteArray, you will have access to the ECMAScript Expression Builder.

6 Click **OK**. The component's Action Model now contains an Execute ECI action.

CICS RPC Specific Expression Builder Extensions

The purpose of the CICS RPC Expression Builder extensions is to provide an ECMAScript programmatic alternative to the functionality performed automatically when you drag and drop between a DOM and a Copybook and a Map action is created. If you click on the expression icon, the Source Expression dialog window appears.

These ECMAScript Extensions are available for you to use in customizing your own Auto Map. Each copybook utilized by the CICS RPC component is represented as an ECMAScript object. These copybook objects have one available method: `getField` (). The `getField` method returns a Copybook Field object which in turn has two methods as discussed below.



- `copybookField getField(String cobolData Desc)`

  Novell exteNd extension method.

  Returns a CopybookField object.

For example:

To resolve duplicate named `cobolDataDesc` in a Copybook, reference the parent `cobol Data Desc` as follows:

`getField("PARTID IN INDATA")` given the following copybooks

```
01COMMAREA
05 INDATA
10PARTID
05 OUTDATA
10PARTID
```

The returned Copybook Field object has two methods:

- `String toString()`
  Novell exteNd extension method.
  Returns the string value for CopybookField object.
- `void setValue(Object aValue)`
  Novell exteNd extension method.
  Sets the value for a CopybookField object.

Using Other Actions in the RPC Component Editor

In addition to actions already discussed, you have all the standard Basic and Advanced Composer actions at your disposal as well. The complete listing of Basic Composer Actions can be found in Chapter 7 of the *Composer User's Guide*. Chapter 8 contains a listing of the more Advanced Actions available to you.

Handling Errors and Messages

This section describes common error messages you may see while using the animation tools.

ECI Connection Error Messages

Exception Messages:

- dt002001: Error executing component: ECI Exception java.io.IOException:CCL6651E: Unable to connect to the Gateway.

```
[address = tiger,port = 2006]
[java.net.UnknownHostException: tiger]
```

- Nested Message: rtCICSRPC000301:ECIException java.io.IOException: CCL6651E: Unable to connect to the Gateway.

```
[address = tiger, por t= 2006]
[java.net.UnknownHostException: tiger]
```

- Nested Message: CCL6651E: Unable to connect to the Gateway.

```
[address = tiger, port = 2006]
[java.net.UnknownHostException: tiger]
```

NOTE: This error message indicates to the user that the Hostname used for the ECI Connection is incorrect and/or does not exist in the environment during execution. **To correct the problem**, verify the parameters in the Connection Dialog specifically check the host name IP address.

Data Type Translation Error Messages

Exception Messages:

- dt002701: Error trying to map ‘TESTALL.getField(“ESIGNED”)’
- Nested Message: rt001002:** Error evaluating

  ‘TESTALL.getField(“ESIGNED”).setValue(Input.Xpath(“COMMAREA/EXTERNAL

  DECIMAL/ESIGNED”))’
- Nested Message: Runtime error Error in Java method set Value

  Caused by exception:

  Error converting source data type to the selected copybook field.

  NOTE: This error message is the most common, when a user enters a value that does not comply with the data type of the target field in a COBOL Copybook. An example would be if a user entered the value LOR8437 into a numeric data type field such as COMP, COMP-3, or 9999999.

A List of Data Types Supported

NOTE: Listed are the COBOL data types supported by CICS RPC for PICTURE clauses.

- Alphabetic items
- Numeric Items
- Numeric-edited items
- Alphanumeric items
- Alphanumeric-edited items
- External floating point items except for exponents

B List of Characters for Data Types Supported

Listed are the characters for the data types supported by CICS RPC for PICTURE clauses.

PICTURE symbols include:

A B E P S V X Z 9 0 / , . + - CR DB * $

These characters are used for editing data types. Their meaning can be found in any COBOL reference source.

C Environmental Differences between Animation Testing and Deployment

Testing

There are environmental differences between testing in CICS RPC and Deployment Testing. The main thing to be aware of is that connection points to the CICS server you use for design-time testing might not be the same as those used in the production environment. In addition, the basic environmental testing differences that apply to *all* components should be noted. See the Envinronmental Differences appendix n the *exteNd Composer User Guide* for more information.

D CICS RPC Glossary

CICS

Stands for Customer Information Control System. An IBM-licensed teleprocessing sub-system that accepts on-line input from terminals, executes programs to fulfill user requests and manages buffers, storage and file input/output.

CICS RPC

CICS RPC stand for Customer Information Control System Remote Procedure Call.

COBOL

Stands for Common Business Oriented Language used primarily on legacy applications.

COMMAREA (DFHCOMMAREA)

Stands for communication area. It is a command level facility used to transfer information between two programs within a transaction or between two transactions from the same terminal.

Copybook

A record structure consisting of individually defined COBOL data descriptors as it is defined in a program or copybook (copybooks can be contained within other copybooks). A copybook may also be reused by other programs and copybooks.

EBCDIC

Collating sequence used by IBM mainframes. In the EBCDIC collating sequence, lowercase letters appear before uppercase letters and numbers appear last.

Endian

Term used to describe the order in which a sequence of bytes are stored in the computer memory. Big-endian is an order in which the most significant value in the sequence is stored first at the lowest storage address. Little-endian is an order where the least significant value in the sequence is stored first.

Gateway Parameters

There are two parameters to fine tune connections: *Initworker* and *Initconnections*. *Initworker* represents the initialized number of worker threads. *Initconnections* represents the initialized number of connections.

Native Environment Pane

A pane in the CICS RPC Component Editor that simulates an actual CICS environment when you issue a query.

E List of Code Pages Supported by Java

Novell exteNd Composer's ability to perform character encoding conversions is tied directly to the Java VM in use. The supported encodings vary between different implementation of the Java 2 platform. For information on the latest encodings, go to **http://java.sun.com.**

Sun's Java 2 Runtime Environment for Windows comes in two different versions: US-only and international. The international version (which includes the **lib\i18n.jar** file) supports all encodings.

Index

O

P

Q

R

T

X