Installing the SAP MII 12.1 JTA JDBC Custom Action Blocks

Applies to:

SAP MII 12.1 and newer, SAP NetWeaver CE 7.1 EhP1. For more information, visit the Manufacturing homepage.

Summary

The custom action blocks referenced in this document are for providing JTA access to any JDBC NetWeaver Datasource that supports commit and rollback functionality. These are useful for maintaining data integrity when performing a large number of commands where support for commit and rollback is required. This latest update of actions includes the use of prepared statements, batching of statements, and the querying of uncommitted data. Download attached files here.

Author: Salvatore Castro

Company: SAP Labs, LLC

Created on: 09 August, 2011

Author Bio

Salvatore Castro of SAP Labs has a Bachelors Degree in Computer Engineering and a Masters Degree in Computer Science both through the Rochester Institute of Technology. He is a member of the MII Product Management group under John Schaefer and came aboard SAP through the Lighthammer acquisition.

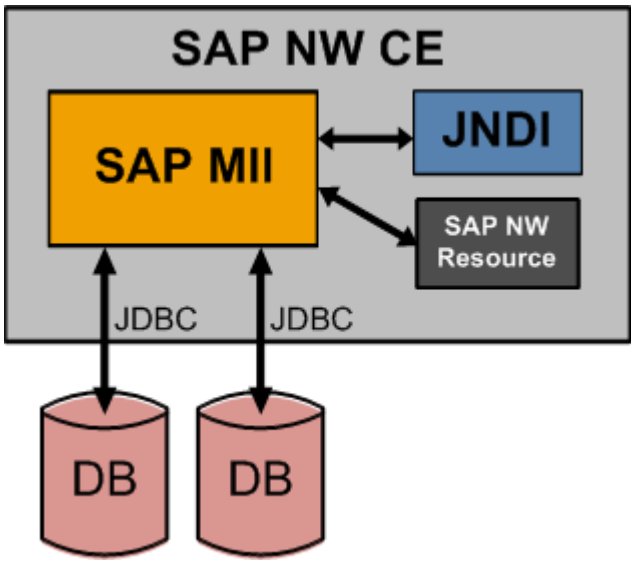
Table of Contents

Business Scenario for Using JTA with JDBC

When integrating various systems in a manufacturing environment the need to write various messages into a single or multiple databases at a high volume and with control over commit and rollback. These actions provide the capability to load data into any number of configured NetWeaver Datasource and based on success or failure criteria can commit and rollback the command operations. These types of operations are typical when performing a large number of commands where it is important to ensure data integrity. If all of the commands are successful then commit the changes, otherwise rollback. The only scenario that is not covered due to a limitation of JDBC is if you have multiple Datasources and the commit operation succeeds on some and not the other Datasources. In this case it's not possible to rollback the already committed data, however it is highly unlikely that a commit will fail.

Technical Operation

The operation of the JDBC Transaction action blocks leverages the use of the NetWeaver Datasource configuration and the ability to lookup the connection via the JNDI. The action blocks automatically add the "jdbc/" name for the JNDI lookup so that the developer only needs to specify the name of the configured data source to use. The actions also support a verbose tracing mode to help provide the developer with additional insight and troubleshooting information at design time. The components referred to in this document are shown in the diagram below:



It is important to note that if multiple commands are performed and a commit is not issued and the end action or the end of the transaction is reached that all of the uncommitted data will be rolled-back. This will also occur if a transaction is terminated before a commit action is reached.

Configuring a NetWeaver CE Datasource

The usage of these actions depends on the creation of a NetWeaver Datasource connection to a database that is either local or available remotely on the network. In order to create a valid Datasource perform the following steps:

1. Login to the SAP NetWeaver administrator
    a. http://<server<:port>>/nwa
2. Select Configuration Management -> Infrastructure
3. Select Application Resources

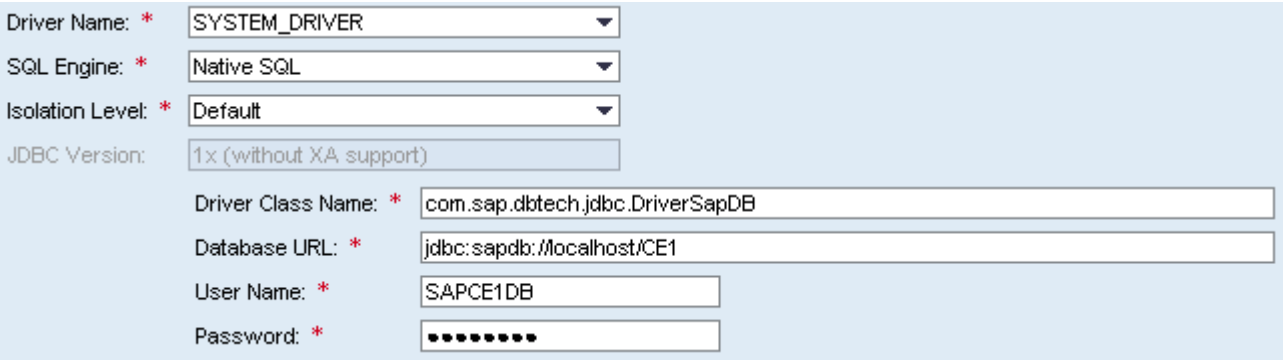
Deploying a JDBC Driver

In order to create a connection to a database instance the JDBC driver must first be deployed, if you have one deployed already skip to the next section, Creating a Datasource.

4. From the Drop-down list select Show JDBC Drivers
5. Verify that your driver is not already in the list
6. Press the “Create New Resource” button and select “Deploy New JDBC Driver”
7. Name the driver and Press the “Add New Driver File” button and browse to the JDBC driver.
8. Press the Save button
9. Verify that the JDBC driver has a Green square for the state

Creating a Datasource

10. From the Drop-down list select JDBC Custom DataSources
11. Verify that a connection to the database instance does not already exist
12. Press the “Create New Resource” button and select “New JDBC Custom DataSource”
13. Select the Driver you need for the connection and define the rest of your connection properties, these are specific to your database. For my JDBC driver to connect to my MaxDB instance I used the following (Note this was my NW database instance):



14. Select the Connection Pooling tab and configure the pool based on your scenario requirements.
15. Press the Save button
16. Verify that the newly created resource has a Green square for the state

Uploading and Deploying the Custom Actions

The deployment of custom actions is documented in the MII v12.1 Help documentation which can be found online (http://help.sap.com -> SAP Business Suite -> Manufacturing Integration & Intelligence).

The specifics for installing these actions are simply to navigate to the following in the MII Menu:

1. System Management -> Custom Actions
2. Press the Upload button
3. In the assembly file upload window perform the following:
    a. Browse to the **SAPMIIJTA.jar** file
    b. Press the Save button
4. Press the Deploy Button

Your actions will now appear in the MII Workbench the next time you open up the transaction editor under the “SAP NW JTA” category.

Using the JTA Actions

Once you have a NetWeaver datasource configured and the JTA Actions deployed you can now use them for managing data loading into your database(s). These actions work in a very similar manner as the current JRA and JCo session action block work with a Start, Command, Commit, Rollback, and End action. It is important to note that if you have not committed your database commands and the End action block is reached all of your data modifications will be rolled back. In order to demonstrate the usage of capabilities of these actions there were two NW Datasources created, Northwind & Northwind2. They both point to separate SAP MaxDB database instances where a database table was created.

To create the table in MaxDB used in this example use the following SQL:

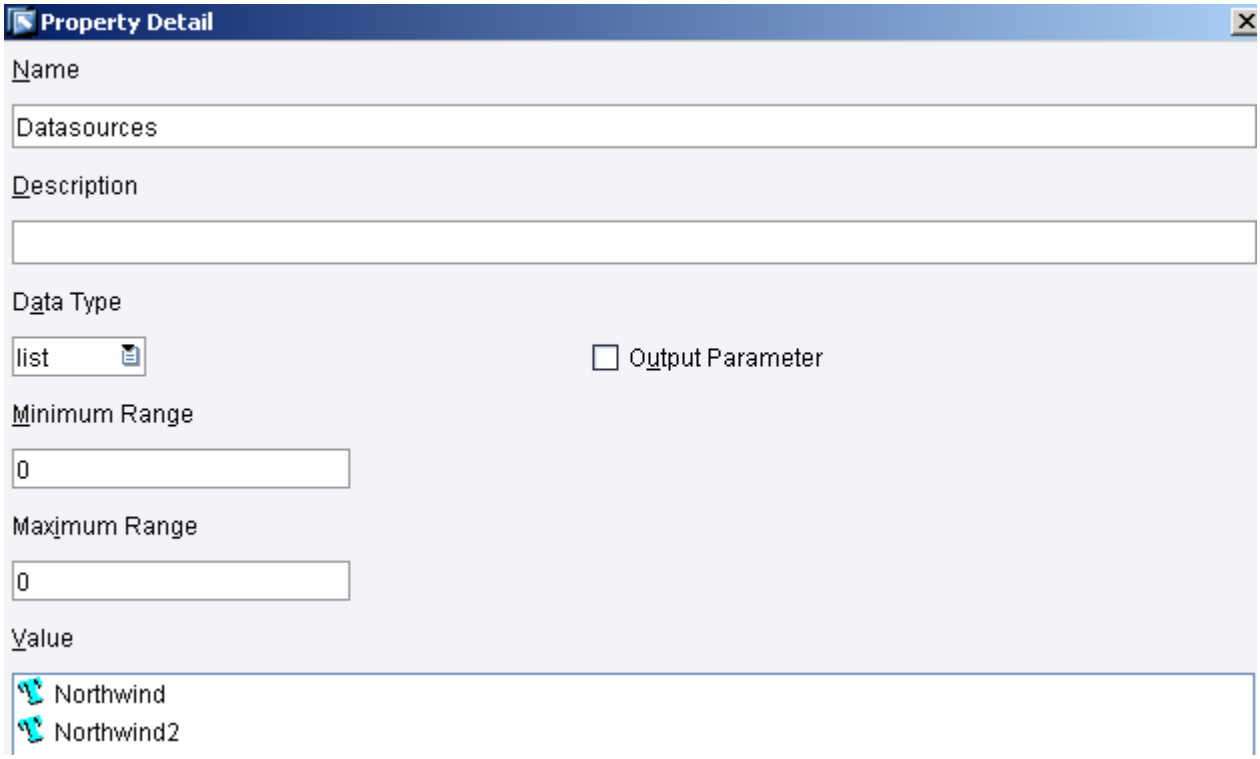
CREATE TABLE MSGS (id INT DEFAULT SERIAL, DateTime TIMESTAMP, Message CHAR(250), TagName CHAR(20), Value DOUBLE PRECISION)

If you have MS SQL you can use this SQL to create the table:

CREATE TABLE MSGS(ID int IDENTITY(1,1) PRIMARY KEY, DATETIME DATETIME, MESSAGE VARCHAR(250), TAGNAME VARCHAR(20), VALUE FLOAT)

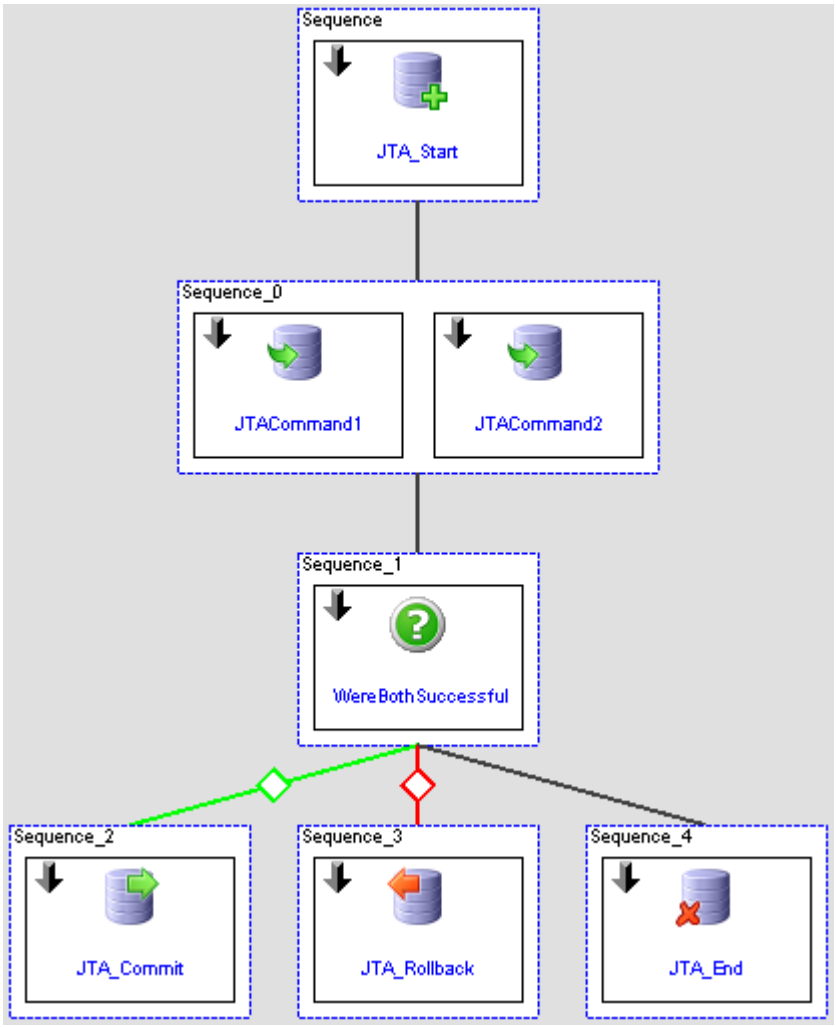
Create a Basic JTA Transaction

Create the following Transaction input property:

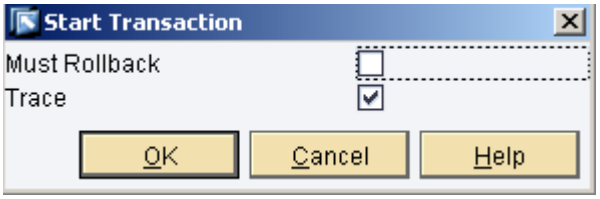


For this example I have two NetWeaver Datasources created and I have added them to the transaction list property.

Next create the following transaction using the newly deployed SAP NW JTA custom actions:



For the JTA_Start action define the following Configuration:



Then link in the Transaction property Transaction.Datasources to the JTA_Start.DataSourceNames

For the JTACommand1 action define the following links:

- "JTA_Start" **to** JTACommand1.JTATransactionStartSource
- "INSERT INTO MSGS(DATETIME, TAGNAME, MESSAGE, VALUE) VALUES('" & datefromxmlformat(datenow, "yyyy-MM-dd HH:mm:ss") & "', 'L1Speed', 'Here', 12.1)" **to** JTACommand1.SQLCommand
- Transaction.Datasources{0} **to** JTACommand1.CommandDatasource

For the JTACommand2 action define the following links

- "JTA_Start" **to** JTACommand2.JTATransactionStartSource
- "INSERT INTO MSGS(DATETIME, TAGNAME, MESSAGE, VALUE) VALUES('" & datefromxmlformat(datenow, "yyyy-MM-dd HH:mm:ss") & "', 'L2Speed', 'There', 12.2)" **to** JTACommand2.SQLCommand
- Transaction.Datasources{1} **to** JTACommand2.CommandDatasource

For the conditional action, WereBothSuccessful, configure it for two inputs and AND them. Next define the following links:

- JTACommand1.Success **to** WereBothSuccessful.Input1
- JTACommand2.Success **to** WereBothSuccessful.Input2

For the JTA_Commit action define the following link:

- "JTA_Start" **to** JTA_Commit.JTATransactionStartSource

For the JTA_Rollback action define the following link:

- "JTA_Start" **to** JTA_Rollback.JTATransactionStartSource

Finally for the JTA_End action define the following link:

- "JTA_Start" **to** JTA_End.JTATransactionStartSource

Now it is time to test the operation of the transaction in your environment in order to better understand the behavior of the actions. Run the transaction so that both commands succeed, so that either one fails, or also so both succeed but without the commit action block and observe how the underlying database table reacts.

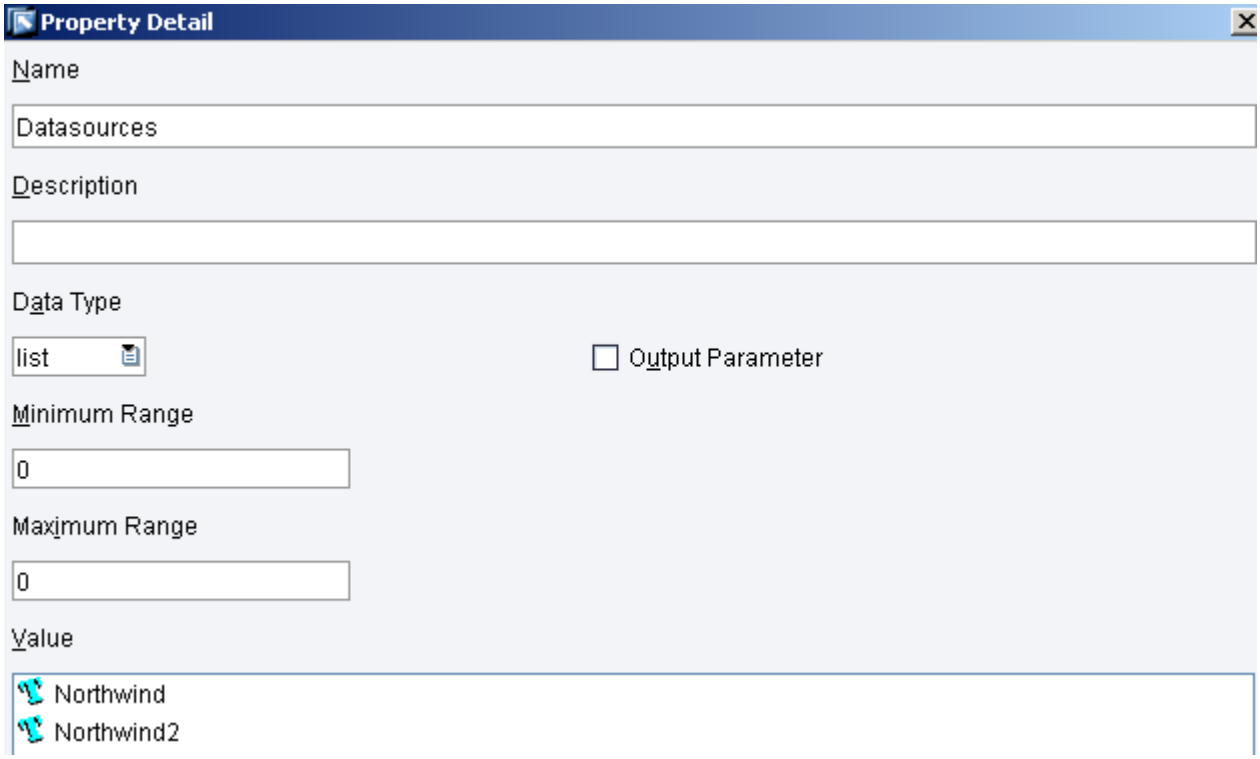
Using the JTA Prepared Statement, Batch Execute, and Query Actions

The actions allow for higher efficiency when interacting with various database instances by allowing the MII transaction to access to additional features of the JDBC interface. Making use of a Prepared Statement has numerous performance advantages over basic SQL Commands primarily due to the ability to cache the command. Not wanting to reiterate a commonly known concept here in this document, the details of how much performance benefits can be realized are dependent on the database, but more information can be found here and on other sites across the internet:

http://www.theserverside.com/news/1365244/Why-Prepared-Statements-are-important-and-how-to-use-them-properly

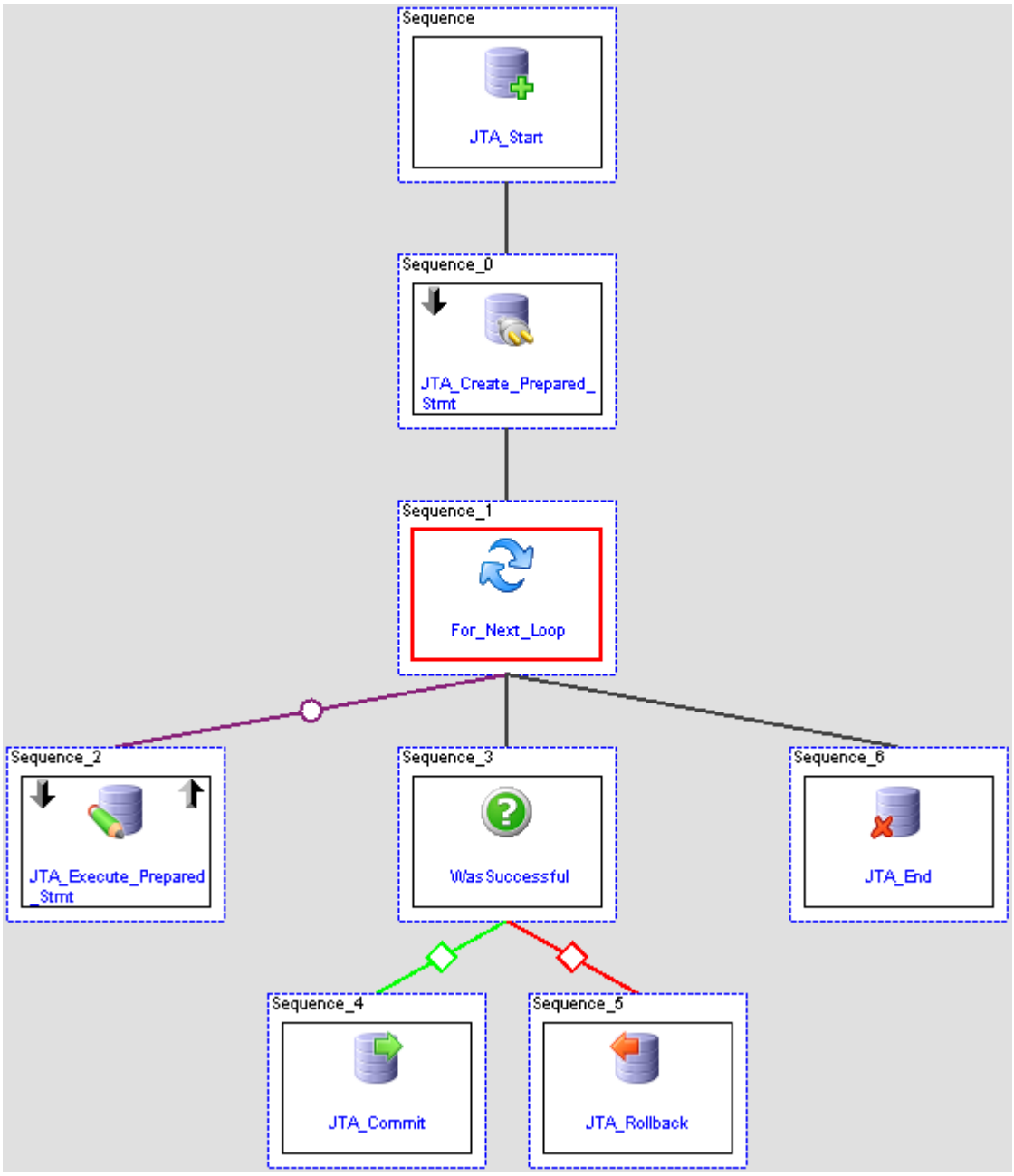
JTA Prepared Statement Action

Create the following Transaction input property:

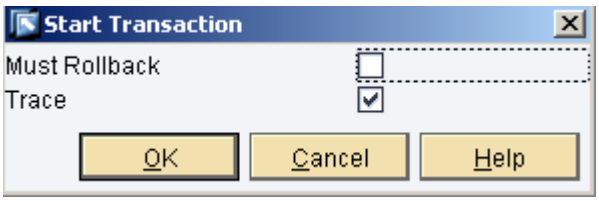


For this example I have two NetWeaver Datasources created and I have added them to the transaction list property.

Next create the following transaction using the newly deployed SAP NW JTA custom actions:



For the JTA_Start action define the following Configuration:



Then link in the Transaction property Transaction.Datasources to the JTA_Start.DataSourceNames

For the JTA_Create_Prepared_Stmt action define the following links:

- "JTA_Start" **to** JTA_Create_Prepared_Stmt.JTATransactionStartSource
- "INSERT INTO MSGS(DATETIME, TAGNAME, MESSAGE, VALUE) VALUES(?,?,?,?)" to JTA_Create_Prepared_Stmt.SQLCommand
- Transaction.Datasources{0} to JTA_Create_Prepared_Stmt.CommandDatasource

For the For_Next_Loop define any number of iterations to run the repeat

For the JTA_Execute_Prepared_Stmt action define the following links

- “JTA_Create_Prepared_Stmt " **to** JTA_Execute_Prepared_Stmt.JTATransactionPreparedStatement
- Link to JTA_Execute_Prepared_Stmt.SQLValueList the value: append(emptylist, datefromxmlformat(datenow, "yyyy-MM-dd HH:mm:ss"), "L2Speed", "There", "12.2")
- In the Outgoing Links, link the expression:
    - !JTA_Execute_Prepared_Stmt.Success
    - To For_Next_Loop.break

For the conditional action, WasSuccessful, check if the JTA_Execute_Prepared_Stmt.Success is true:

For the JTA_Commit action define the following link:

- "JTA_Start" **to** JTA_Commit.JTATransactionStartSource

For the JTA_Rollback action define the following link:

- "JTA_Start" **to** JTA_Rollback.JTATransactionStartSource

Finally for the JTA_End action define the following link:

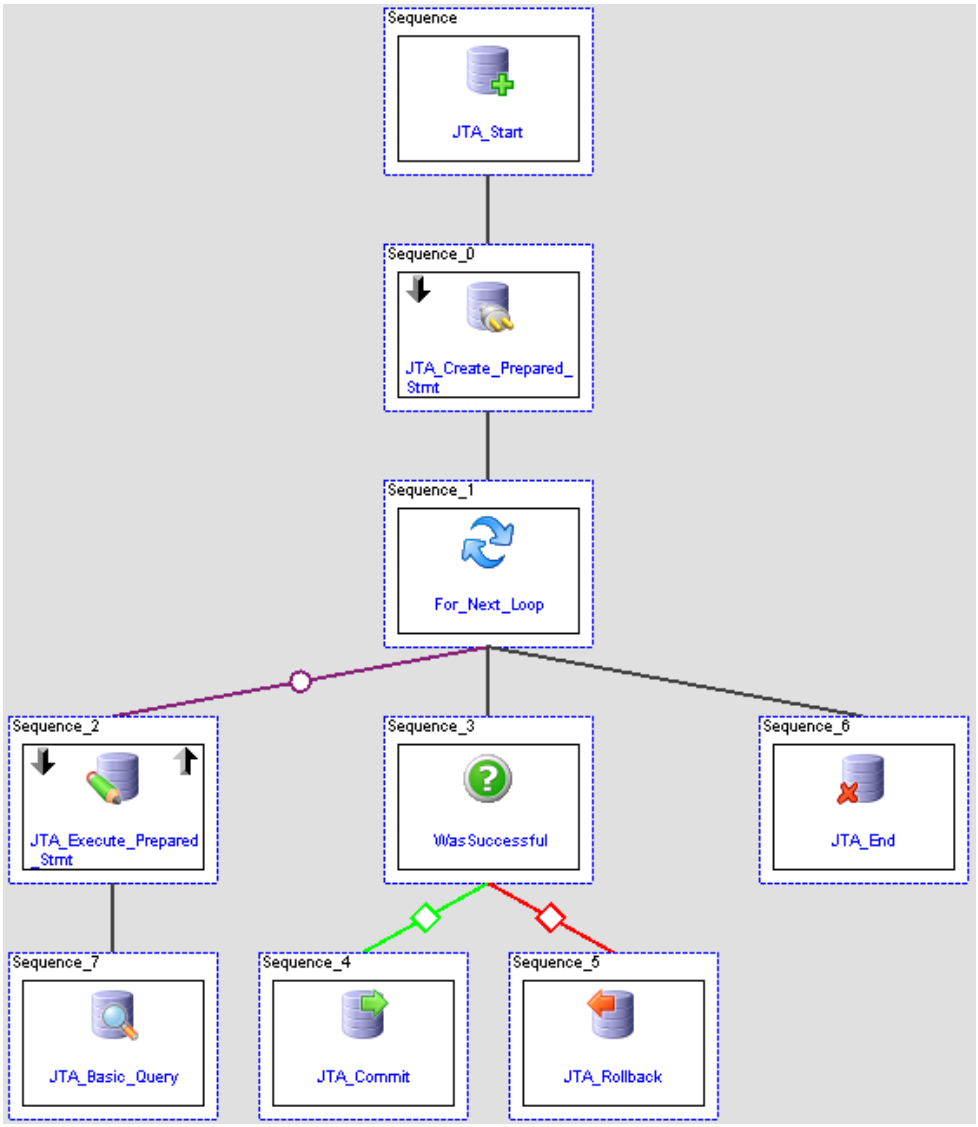
- "JTA_Start" **to** JTA_End.JTATransactionStartSource

Now it is time to test the operation of the transaction in your environment in order to better understand the behavior of the actions. Run the transaction so that both commands succeed, so that either one fails, or also so both succeed but without the commit action block and observe how the underlying database table reacts.

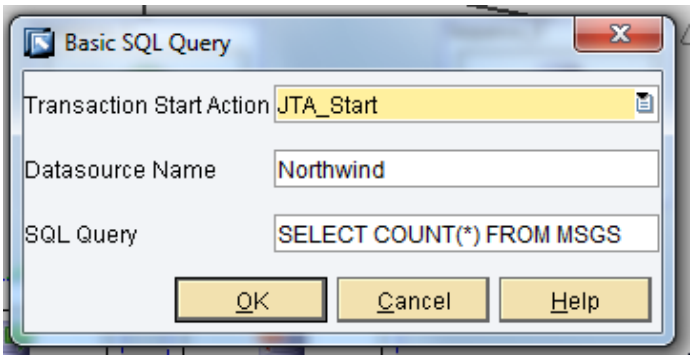
Using the JTA Basic Query

The purpose of this query is to read values out of the database that may not be committed yet so that the flexibility to the JDBC interface is exposed to the user. This may be useful when multiple systems are writing data to a table simultaneously and calculations need to be made on the data before it's committed to a table.

From the transaction above modify it to include a JTA_Basic_Query action after the JTA_Execute_Prepared_Stmt action as shown below:



Configure the JTA_Basic_Query action as follows:

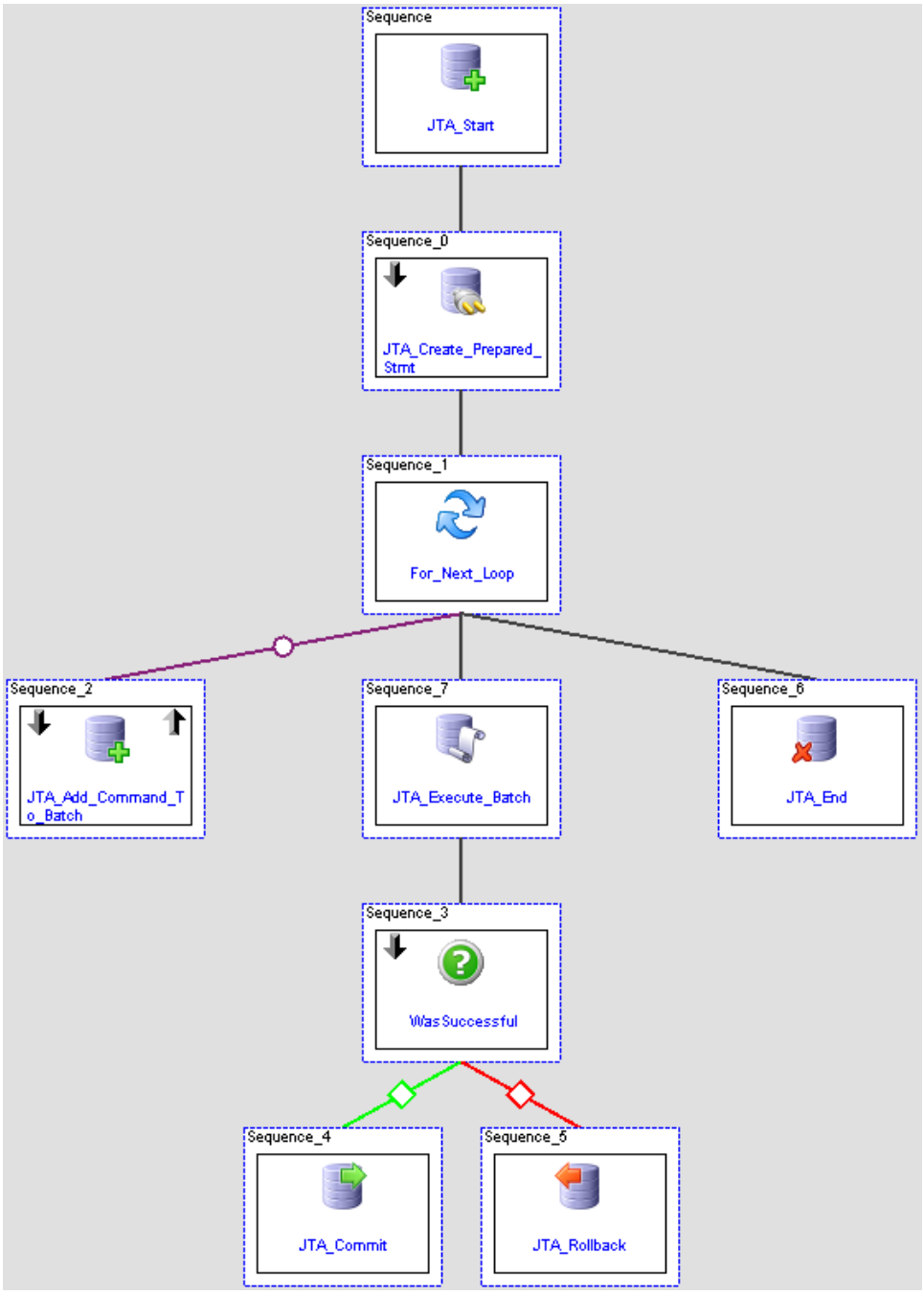


Add in a tracer action after the JTA_Basic_Query that logs out the contents of the JTA_Basic_Query .OutputXML property. You will see the count rise even though the values are not yet committed to the database.

Using the Batch Prepared Statement Action

This action is useful to control the number of commands that are executed simultaneously rather than individually executing them.

To use this action modify the above transaction to look like this:



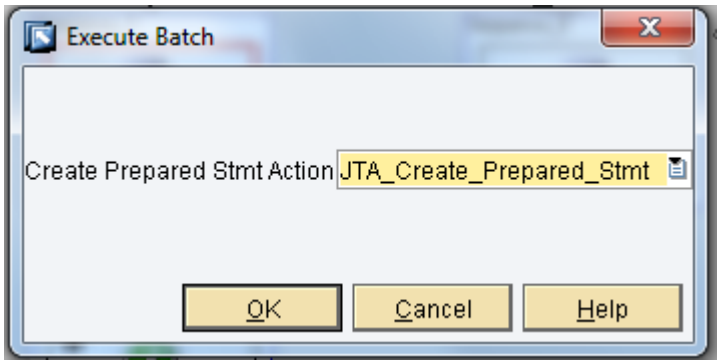
Configure the JTA_Add_Command_To_Batch action to point to the JTA_Create_Prepared_Statement action and link in this value:

- append(emptylist, datefromxmlformat(datenow, "yyyy-MM-dd HH:mm:ss"), "L2Speed", "There", "12.2")
- To the input JTA_Add_Command_To_Batch.SQLValueList

For the JTA_Add_Command_To_Batch Outgoing links, link in this value:

- !JTA_Add_Command_To_Batch.Success
- To For_Next_Loop.Break

Next configure the JTA_Execute_Batch action to look like this:



For the WasSuccessful action modify the Incoming link so that the JTA_Execute_Batch.Success is linked to the WasSuccessful.Input1 property.

Related Content

The SAP MII Wiki

The SAP MII Forum

SAP NetWeaver CE Datasource Help Page

Sun Java JDBC Tutorial

For more information, visit the Manufacturing homepage.

Copyright