Importing an SNA Custom Handshake Class

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–3498–05
June 2008

080916@20795

Contents

◆ ◆ ◆ CHAPTER 1

Importing an SNA Custom Handshake Class

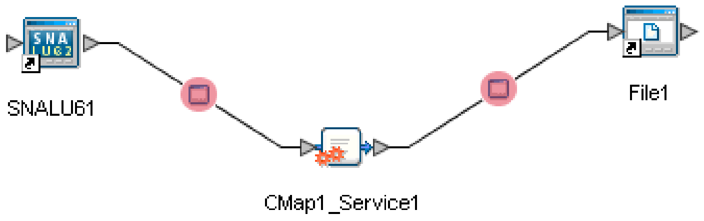
This page provides links to conceptual information on how to import an SNA Custom Handshake Classs.

- "To Import an SNA Custom Handshake Class" on page 5
- "Deploying an SNA Custom Handshake Class" on page 12

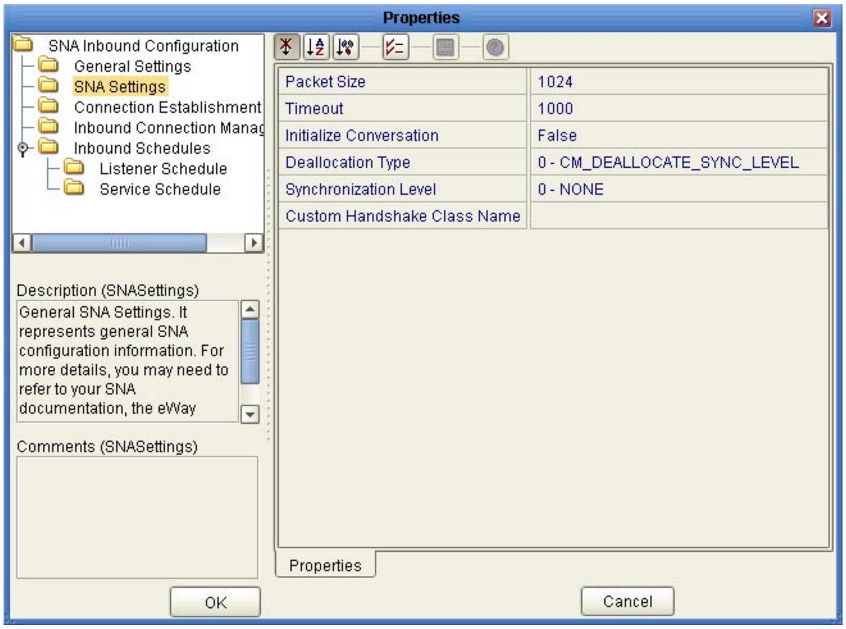
To Import an SNA Custom Handshake Class

To import an SNA custom handshake class, follow the instructions below:

1. On the Enterprise Designer's Inbound SNA Connectivity Map, double-click the SNA Adapter icon.



The SNA Adapter Properties window appears, displaying the default properties for the Inbound Adapter.



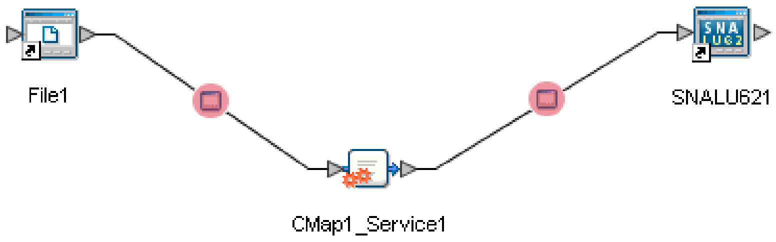
Description (SNASettings)

General SNA Settings. It represents general SNA configuration information. For more details, you may need to refer to your SNA documentation, the eWay

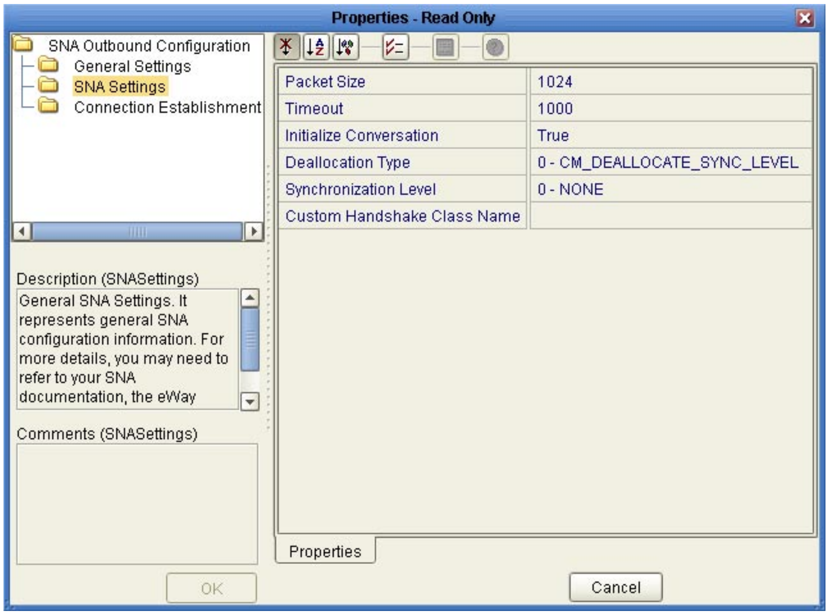
Comments (SNASettings)

Packet Size	1024
Timeout	1000
Initialize Conversation	False
Deallocation Type	0 - CM_DEALLOCATE_SYNC_LEVEL
Synchronization Level	0 - NONE
Custom Handshake Class Name	

2. Edit the Custom Handshake Class Name property in the Inbound Properties window. For the sample code provided with the Adapter, enter com.stc.connector.snalu62.api.SNACustomerHandshakeImplSampleAccept.
3. On the Enterprise Designer's Outbound SNA Connectivity Map, double-click the SNA Adapter icon.



The SNA Adapter Properties window appears, displaying the default properties for the Oubound Adapter.



Packet Size	1024
Timeout	1000
Initialize Conversation	True
Deallocation Type	0 - CM_DEALLOCATE_SYNC_LEVEL
Synchronization Level	0 - NONE
Custom Handshake Class Name	

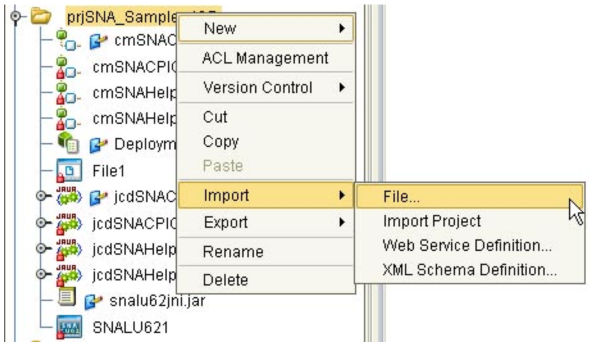
Description (SNASettings)
General SNA Settings. It represents general SNA configuration information. For more details, you may need to refer to your SNA documentation, the eWay

Comments (SNASettings)

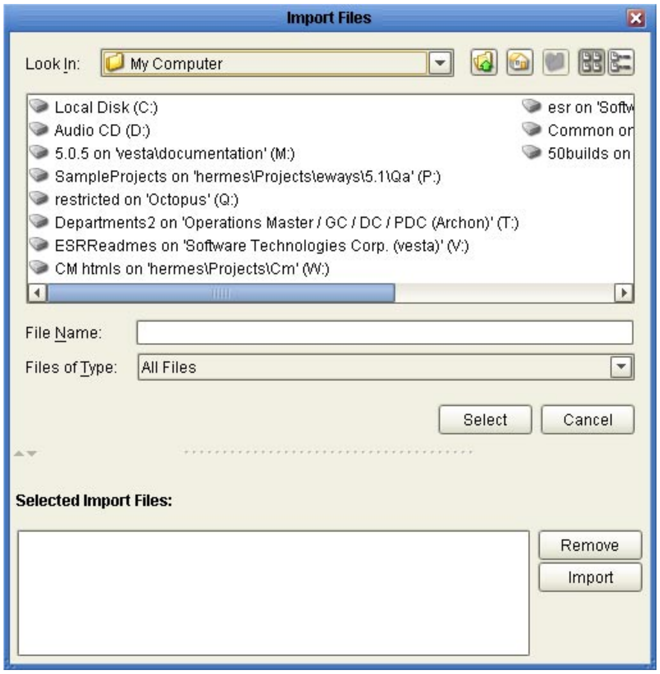
4. Edit the Custom Handshake Class Name property in the Outbound Properties window. For the sample code provided with the Adapter, enter com.stc.connector.snalu62.api.SNACustomerHandshakeImplSampleInitialize.
5. Redeploy your project.

Steps when Building your own Class:

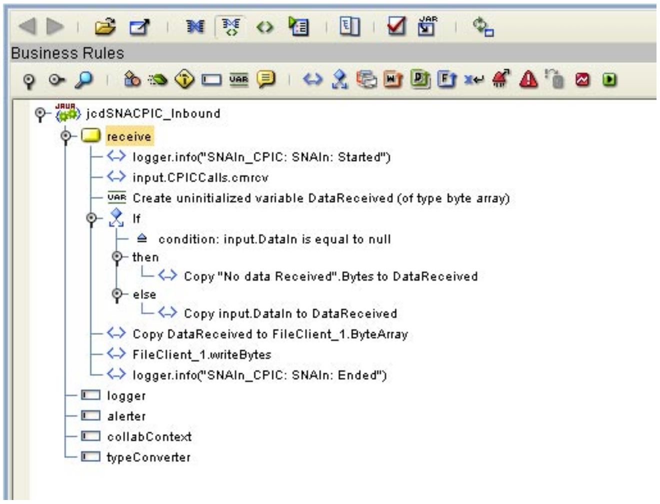
1. Prepare a JAR file that includes your built class.
2. From the Project Explorer, right-click the sample Project and select Import > File from the shortcut menu.



The Import Files window appears.


Import Files
Look In:
My Computer
Local Disk (C:)
Audio CD (D:)
5.0.5 on 'vesta\documentation' (M:)
SampleProjects on 'hermes\Projects\eways\5.1\Qa' (P:)
restricted on 'Octopus' (Q:)
Departments2 on 'Operations Master / GC / DC / PDC (Archon)' (T:)
ESRReadmes on 'Software Technologies Corp. (vesta)' (V:)
CM htmls on 'hermes\Projects\Cm' (W:)
esr on 'Softw
Common or
50builds on
File Name:
Files of Type:
All Files
Select
Cancel
Selected Import Files:
Remove
Import

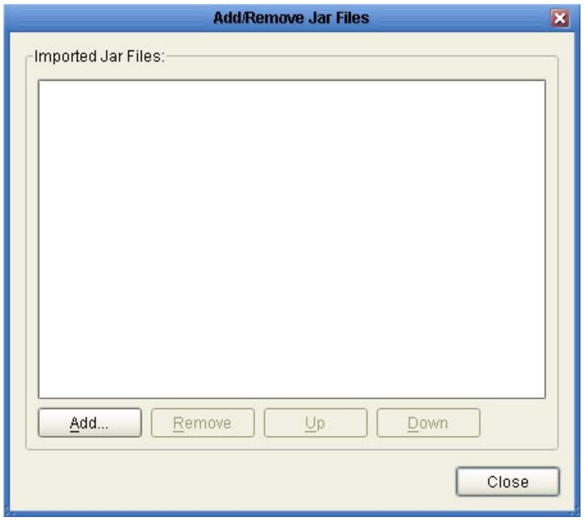
3. Locate your JAR file and click Select. Your selected JAR file appears in the Selected Import Files pane at the bottom of the Import Files window.
4. Click Import. Your selected JAR file appears in your sample Project's folder in the left pane of the Enterprise Designer.
5. Click the Import JAR file button on the Business Rules toolbar in the right pane of Enterprise Designer.



Business Rules

```
jcdSNACPIC_Inbound
  receive
    logger.info("SNAIn_CPIC: SNAIn: Started")
    input.CPICCalls.cmrcv
    Create uninitialized variable DataReceived (of type byte array)
    If
      condition: input.DataIn is equal to null
      then
        Copy "No data Received".Bytes to DataReceived
      else
        Copy input.DataIn to DataReceived
    Copy DataReceived to FileClient_1.ByteArray
    FileClient_1.writeBytes
    logger.info("SNAIn_CPIC: SNAIn: Ended")
  logger
  alerter
  collabContext
  typeConverter
```

The Add/Remove JAR Files window appears.



6. Locate your JAR file and click Add. Your selected JAR file appears in the Imported JAR Files pane.
7. Click Close. Your selected JAR file appears under your sample Project's Collaboration in the left pane of the Enterprise Designer.

Note – If you make any changes to the class, repeat steps 2 through 7.

The Java Collaboration can handle the SNA connection completely using the sample Class (com.stc.connector.snalu62.api.SNACustomerHandshakeImplSampleDummy). This class has been implemented in the SNA Adapter. The sample code for this custom class is as follows:

```
package com.stc.connector.snalu62.api;

import com.stc.connector.logging.LogFactory;
import com.stc.connector.logging.Logger;
import com.stc.connector.snalu62.exception.SNAApplicationException;

/**
 * This is a sample class to implement the interface SNACustomerHandshake.
 * It implements a dummy handshake. That is, the method startConversation()
 does not perform a function.
 * No SNA conversation is established inside this implementation class.
You should establish the SNA conversation manually
(e.g. in the java Collaboration).
*/

public class SNACustomerHandshakeImplSampleDummy implements SNACustomerHandshake {
    public static final String version = "cvs $Revision: 1.1.2.2 $
$Date: 2005/11/10 21:40:15 $";
    private Logger logger = LogFactory.getLogger
("STC.eWay.SNALU62." + getClass().getName());

    /**
     * Constructor
     *
     */
    public SNACustomerHandshakeImplSampleDummy() {
        super();
    }

    /**
     * @see com.stc.connector.snalu62.api.SNACustomerHandshake#startConversation
(com.stc.connector.snalu62.api.SNACPICCalls)
     */
    public void startConversation(SNACPICCalls cpic) throws SNAApplicationException {
        logger.info("SNACustomerHandshakeImplSampleDummy.startConversation():
Done nothing here.");

    }

}
```

Deploying an SNA Custom Handshake Class

To further utilize the capabilities of the SNA Adapter, this sectoin provides guidelines for implementing a custom handshake class in a deployed Project. After the default Collaboration is generated, you can then modify the Collaboration to suit your application's needs. While you will need to write your own code for both Inbound and Outbound SNA modes, the following code is also provided as the source for the class that is implemented in the SNA adapter.

Sample Code for Inbound Mode:

```
package com.stc.connector.snalu62.api;

import com.stc.connector.logging.LogFactory;
import com.stc.connector.logging.Logger;

import com.stc.connector.snalu62.exception.SNAApplicationException;

/*
 * This is a sample class to implement the interface SNACustomerHandshake.
 * It implements a simple Accept_Conversation scenario for windows platform.
*/
public class SNACustomerHandshakeImplSampleAccept implements SNACustomerHandshake {
    public static final String version = "cvs $Revision: 1.1.2.1.2.2 $
$Date: 2005/11/10 21:40:15 $";
    private Logger logger = LogFactory.getLogger("STC.eWay.SNALU62.
" + getClass().getName());
    private String logMsg;

    /**
     * Constructor
     *
     */
    public SNACustomerHandshakeImplSampleAccept() {
        super();
    }
    /**
     * @see com.stc.connector.snalu62.api.SNACustomerHandshake#startConversation
(com.stc.connector.snalu62.api.SNACPICCalls)
     */
    public void startConversation(SNACPICCalls cpic) throws SNAApplicationException {
        try {
            //do whatever checking logics before/after the following CPIC call
 on your desires
            cpic.cmsltp();

            //do whatever checking logics before/after the following CPIC call
 on your desires
           cpic.cmaccp();
           if (!cpic.getConversationAttributes().returnCodeIs(0) &&   // 0: CM_OK
               !cpic.getConversationAttributes().returnCodeIs(35))
{  //35: CM_OPERATION_INCOMPLETE
               logMsg = "SNACustomerHandshakeImplSampleAccept.startConversation():
The return code is <"
                   + cpic.getConversationAttributes().getReturnCode()
                   + ">.";
               logger.error(logMsg);
               throw new SNAApplicationException(logMsg);
           }

           if (cpic.getConversationAttributes().returnCodeIs(35))
{  //35: CM_OPERATION_INCOMPLETE
               logger.info("SNACustomerHandshakeImplSampleAccept.startConversation():
About to call cmwait ...");
               //do whatever checking logics before/after the following CPIC call
 on your desires
               cpic.cmwait();
           }

           if (!cpic.getConversationAttributes().returnCodeIs(0) ||
               !cpic.getConversationAttributes().convReturnCodeIs(0)) { // 0: CM_OK
               logMsg = "SNACustomerHandshakeImplSampleAccept.startConversation():
The return_Code is <"
                   + cpic.getConversationAttributes().getReturnCode()
                   + "> and the conversation_Return_Code is <"
                   + cpic.getConversationAttributes().getConvReturnCode()
                   + ">. SNA conversation is not established.";
               logger.error(logMsg);
               throw new SNAApplicationException(logMsg);
           }

           //do whatever other logics on your desires here
           //...
       } catch (Exception e) {
           logMsg = "SNACustomerHandshakeImplSampleAccept.startConversation():
Failed. Got exception ["
               + e.toString()
               + "].";
           logger.error(logMsg, e);
           throw new SNAApplicationException(logMsg, e);
       }

    }

}
```

Sample Code for Outbound Mode:

```
package com.stc.connector.snalu62.api;

import com.stc.connector.logging.LogFactory;
import com.stc.connector.logging.Logger;

import com.stc.connector.snalu62.exception.SNAApplicationException;

/**
 * This is a sample class to implement the interface SNACustomerHandshake.
 * It implements a simple Initialize_Conversation scenario for windows platform.
*/
public class SNACustomerHandshakeImplSampleInitialize implements SNACustomerHandshake {
    public static final String version = "cvs $Revision:
 1.1.2.1.2.2 $   $Date: 2005/11/10 21:40:15 $";
    private Logger logger = LogFactory.getLogger("STC.eWay.SNALU62." + getClass().
getName());
    private String logMsg;

    /**
     * Constructor
     *
     */
    public SNACustomerHandshakeImplSampleInitialize() {
        super();
    }

    /**
     * @see com.stc.connector.snalu62.api.SNACustomerHandshake#startConversation
(com.stc.connector.snalu62.api.SNACPICCalls)
     */
    public void startConversation(SNACPICCalls cpic) throws SNAApplicationException {
        try {
            //do whatever checking logics before/after the following CPIC call on your
 desires
            cpic.cminit();

            //do whatever checking logics before/after the following CPIC call
 on your desires
            cpic.cmssl();

            //do whatever checking logics before/after the following CPIC call on your
desires
            cpic.cmallc();
            if (!cpic.getConversationAttributes().returnCodeIs(0)) { // 0: CM_OK
                logMsg = "SNACustomerHandshakeImplSampleInitialize.
startConversation(): The return_Code is <"
```

```
                    + cpic.getConversationAttributes().getReturnCode()
                    + ">. SNA conversation is not established.";
                logger.error(logMsg);
                throw new SNAApplicationException(logMsg);
            }

            //do whatever other logics on your desires here
            //...
        } catch (Exception e) {
            logMsg = "SNACustomerHandshakeImplSampleInitialize.startConversation():
Failed. Got exception ["
                + e.toString()
                + "].";
            logger.error(logMsg, e);
            throw new SNAApplicationException(logMsg, e);
        }

    }

}
```