



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

Copyright 2008 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more U.S. patents or pending patent applications in the U.S. and in other countries.

U.S. Government Rights – Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, the Solaris logo, the Java Coffee Cup logo, docs.sun.com, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and SunTM Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Products covered by and information contained in this publication are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical or biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Contents

1	Importing an SNA Custom Handshake Class	5
	To Import an SNA Custom Handshake Class	5
	Steps when Building your own Class:	8
	Deploying an SNA Custom Handshake Class	12
	Sample Code for Inbound Mode:	12
	Sample Code for Outbound Mode:	14

Importing an SNA Custom Handshake Class

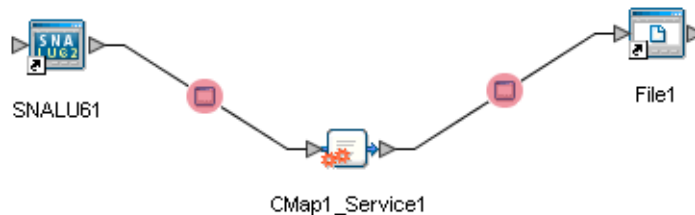
This page provides links to conceptual information on how to import an SNA Custom Handshake Class.

- [“To Import an SNA Custom Handshake Class” on page 5](#)
- [“Deploying an SNA Custom Handshake Class” on page 12](#)

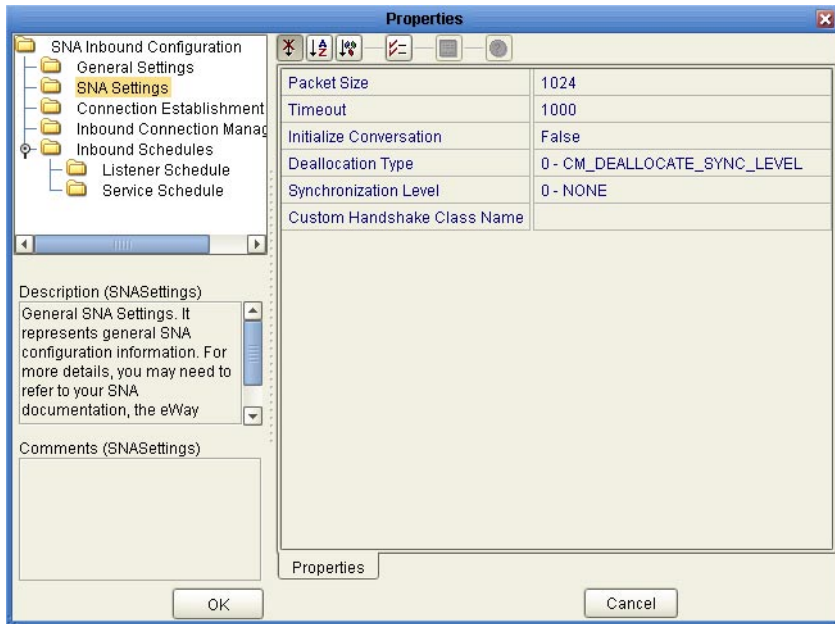
To Import an SNA Custom Handshake Class

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

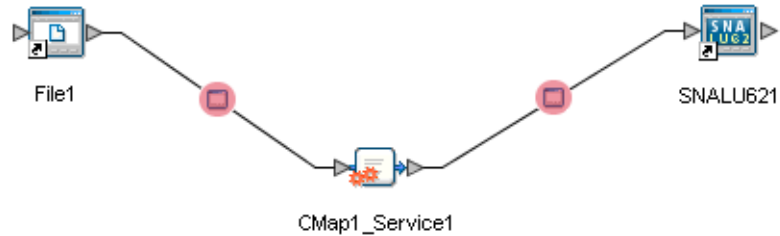
1. On the Enterprise Designer’s Inbound SNA Connectivity Map, double-click the SNA Adapter icon.



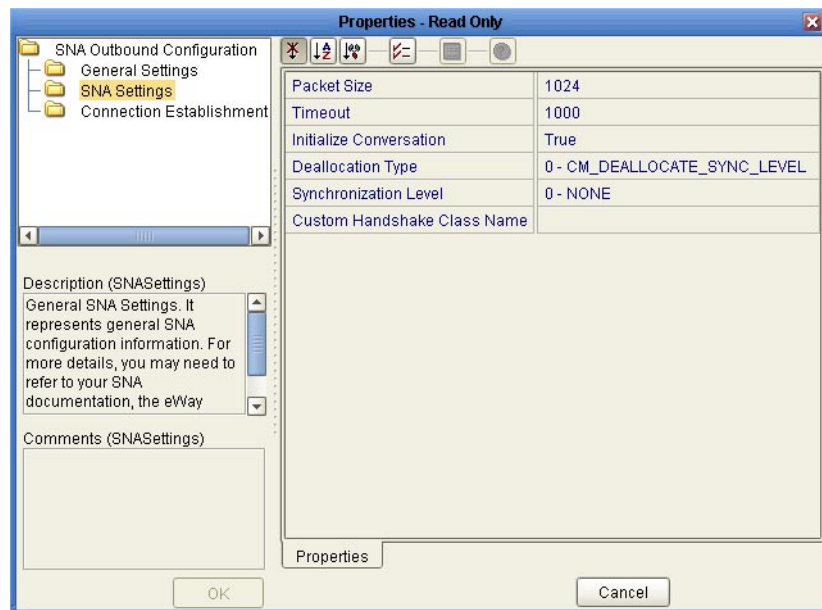
The SNA Adapter Properties window appears, displaying the default properties for the Inbound Adapter.



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.snal62.api.SNACustomerHandshakeImplSampleAccept`.
3. On the Enterprise Designer's Outbound SNA Connectivity Map, double-click the SNA Adapter icon.



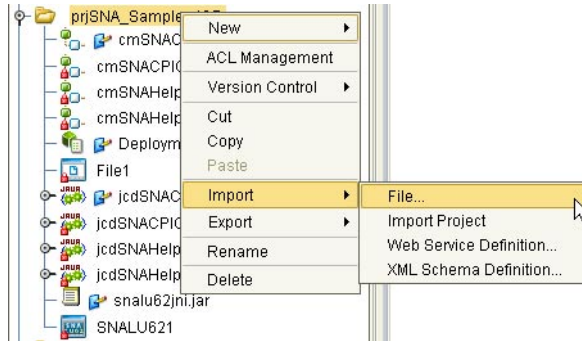
The SNA Adapter Properties window appears, displaying the default properties for the Outbound Adapter.



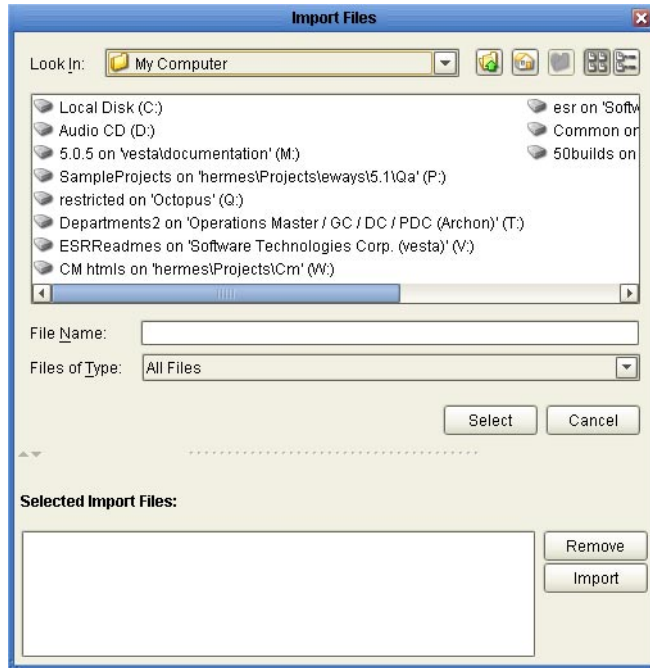
4. Edit the Custom Handshake Class Name property in the Outbound Properties window. For the sample code provided with the Adapter, enter `com.stc.connector.snalu62.api.SNACustomerHandshakeImplSampleInitialize`.
5. Redeploy your project.

Steps when Building your own Class:

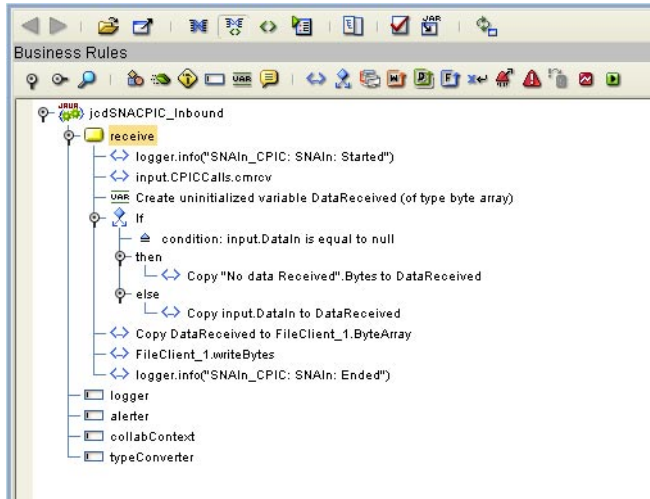
1. Prepare a JAR file that includes your built class.
2. From the Project Explorer, right-click the sample Project and select Import > File from the shortcut menu.



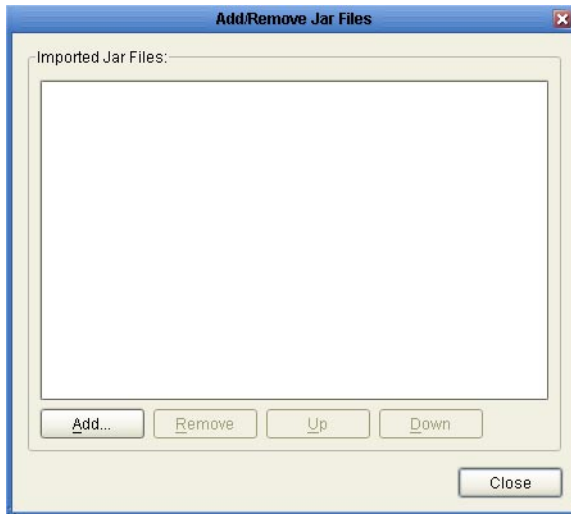
The Import Files window appears.



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.



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 section 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.snal62.api;

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

import com.stc.connector.snal62.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.snal62.api.SNACustomerHandshake#startConversation
     (com.stc.connector.snal62.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);
}

}

}
```

