Zipping Payload in Java Mapping

Applies to:

SAP NetWeaver™ Exchange Infrastructure/Process Integration.

Summary

This article illustrates how to zip the payload in java mapping.

Author: Praveen Gujjeti, SAP NetWeaver Consultant.

Company: Bristlecone India Ltd.

Created on: 02 May 2008

Author Bio

Praveen Gujjeti is a certified SAP NetWeaver Consultant (PI Development) at Bristelcone India Ltd. His areas of expertise include JAVA/J2EE, Webmethods and SAP XI/PI.

Table of Contents

Introduction

This article demonstrates a sample case why we zip the payload in java mapping.

Scenario Case

We have a requirement where sender system is SCM (ABAP PROXY) and it is sending business data (Material Details) to XI/PI system. After this we have to post this data to HTTPServlet which is basically running on some other landscape

If we post the same data as we are receiving from SCM system it will take a lot of time for network transmission as HTTPServlet is running on some other landscape over the internet. Also we can't use the generic PAYLOAD ZIP BEAN either on Sender side or on the Receiver side as sender is ABAP PROXY and receiver is HTTP Servlet (both are on ABAP Stack, hence no modules).

Possible Solutions,

1. ZIP the data on the PROXY side.
2. ZIP the payload in Interface Mapping (Java Mapping) and route it to the receiver system (HTTPServlet).

Here I am considering the second case.

Scenario

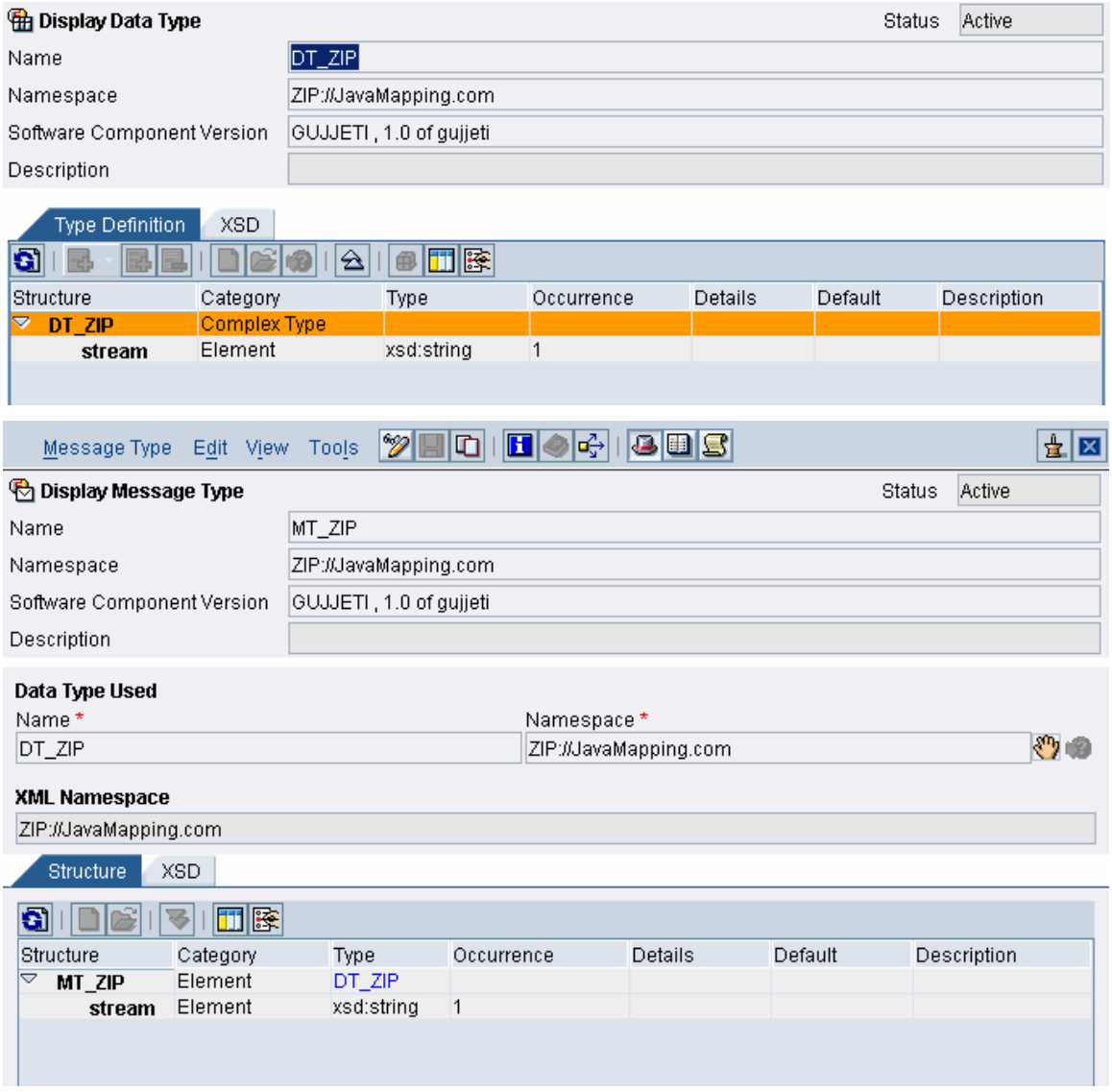
For simplicity & demonstration purposes I am using File System as the sender System (Don't use payload zip bean in Sender File Adapter as our aim is to zip it in the java mapping for the scenario case described above). Hence Scenario would be **FILE** to **HTTP Receiver.**

Prerequisites

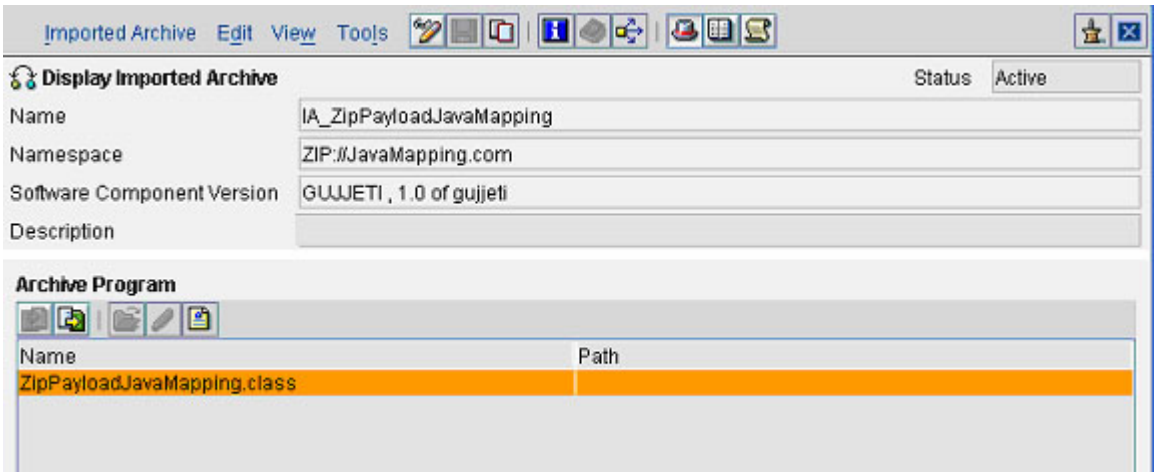
SAP XI/PI version with java mapping capabilities and a web server (TOMCAT) for testing this scenario.

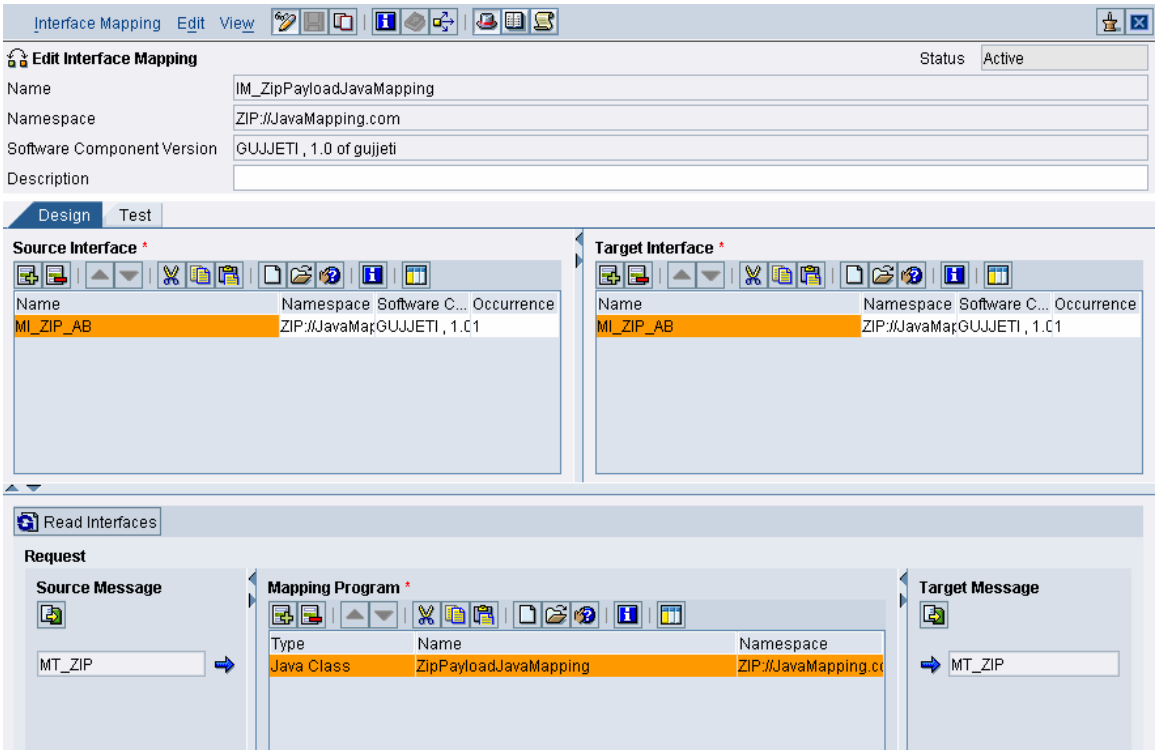
Design Objects in IR

1. Create one Data Type.
2. Create one Message Type and assign the above Data Type.
3. Create one Message Interface of Type Asynchronous, Abstract.(will be used for both File System on inbound side and HTTP Servlet application on outbound side w.r.t. XI/PI)
4. Import the externally compiled java code in the imported achieves.(check the code below)
5. Create interface Mapping. (Use the above java class in mapping).




Message Interface Edit View Tools
Edit Message Interface
Status Active
Name MI_ZIP_AB
Namespace ZIP://JavaMapping.com
Software Component Version GUJJETI , 1.0 of gujjeti
Description
Definition Context Objects WSDL
Attributes
Category Inbound Outbound Abstract
Mode Synchronous Asynchronous
Message Types
Message Type Name * Namespace *
MT_ZIP ZIP://JavaMapping.com


Imported Archive Edit View Tools
Display Imported Archive
Status Active
Name IA_ZipPayloadJavaMapping
Namespace ZIP://JavaMapping.com
Software Component Version GUJJETI , 1.0 of gujjeti
Description
Archive Program
Name Path
ZipPayloadJavaMapping.class


Interface Mapping Edit View
Edit Interface Mapping
Status Active
Name IM_ZipPayloadJavaMapping
Namespace ZIP://JavaMapping.com
Software Component Version GUJJETI , 1.0 of gujjeti
Description
Design Test
Source Interface *
Name Namespace Software C... Occurrence
MI_ZIP_AB ZIP://JavaMap GUJJETI , 1.0 1
Target Interface *
Name Namespace Software C... Occurrence
MI_ZIP_AB ZIP://JavaMap GUJJETI , 1.0 1
Read Interfaces
Request
Source Message
MT_ZIP
Mapping Program *
Type Name Namespace
Java Class ZipPayloadJavaMapping ZIP://JavaMapping.c
Target Message
MT_ZIP

ZipPayloadJavaMapping Code

```
//---------------------------------------------------------------------------CODE---
-----------------------------------------------------
import com.sap.aii.mapping.api.StreamTransformation;
import com.sap.aii.mapping.api.*;
import java.io.*;
import java.util.*;
import java.util.zip.*;
import java.util.Map;
public class ZipPayloadJavaMapping implements StreamTransformation
{
   static final String ENTRYNAME = "ServeltShouldProcessThis.xml";
   static final int BUFFER = 1024*1000;
   String ourSourceFileName;
          private Map param;
          public ZipPayloadJavaMapping(){  }
   public void setParameter (Map map)
   {
          param = map;
          if (param == null)
          {
                 param = new HashMap();
          }
   }
   public static void main(String args[])
   {
          try
          {
                 FileInputStream fin = new FileInputStream(args[0]);
                 FileOutputStream fout = new FileOutputStream(args[1]);
                 ZipPayloadJavaMapping mapping = new ZipPayloadJavaMapping();
                 mapping.execute(fin, fout);
          }
          catch (Exception e)
                 {
                        e.printStackTrace();
                 }
   }
    public void execute(InputStream inputstream, OutputStream outputstream)
    {
                 DynamicConfiguration conf = (DynamicConfiguration)
param.get("DynamicConfiguration");
                 DynamicConfigurationKey KEY_FILENAME =
DynamicConfigurationKey.create("http://sap.com/xi/XI/System/File","FileName");

                 ourSourceFileName = conf.get(KEY_FILENAME);
                 int len = 0;
                 byte buf[] = new byte[BUFFER];

                 try
                 {
                        BufferedInputStream bis = new
BufferedInputStream(inputstream);
                        ZipInputStream zis = new ZipInputStream(bis );
```

```
                        ZipEntry objZipEntry = null;
                        bis.mark(0);

                        //ZipInputStream zis = new ZipInputStream(inputstream);

                        ZipOutputStream zos = new ZipOutputStream(outputstream);
                        if((objZipEntry = zis.getNextEntry()) != null)
                        {
                               bis.reset();
                               while ((len = bis.read(buf)) > 0)

                                      outputstream.write(buf, 0, len);

                        }
                        else
                        {
                               //DOM, SAX or NORMAL PARSING POSSILBE before ZIPPING -
    NOT IMPLEMENTED
                               bis.reset();
                               if (ourSourceFileName != null)
                               {
                                      objZipEntry = new ZipEntry(ourSourceFileName);
                               }
                               else
                               {
                                       objZipEntry = new ZipEntry(ENTRYNAME);
                               }
                               zos.putNextEntry(objZipEntry);
                               while ((len = bis.read(buf)) > 0)
                               {
                                      zos.write(buf, 0, len);
                               }
                               zis.close();
                               zos.close();
                        }
                }
                catch(Exception e){//Write the error to a log file on Server
    Directory – Not Implemented}
        }
    }
    //-------------------------------------------------------------------------CODE---
    ------------------------------------------------------
```

To compile the above program you need set “aii_map_api.jar” file in your CLASSPATH. Once it is compiled ZIP the class file using zip utility such as WinZip or use JAR Command of JAVA.

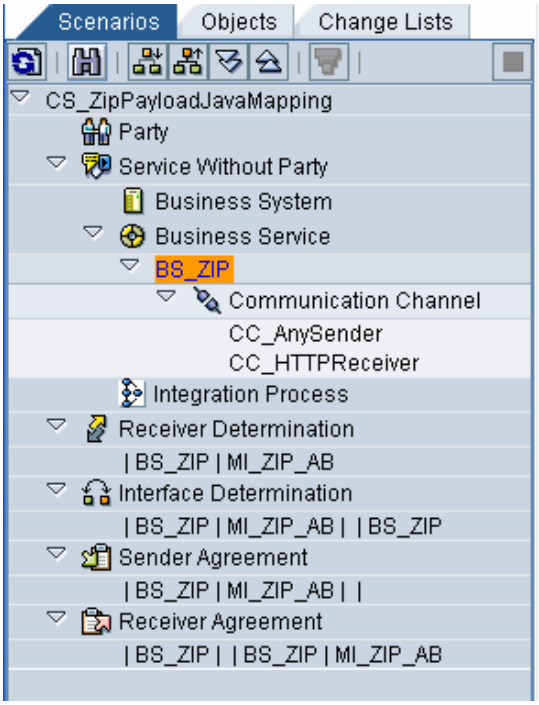
(Working Directory :> jar –cvMf <ZIPFileName>.zip ZipPayloadJavaMapping.class).

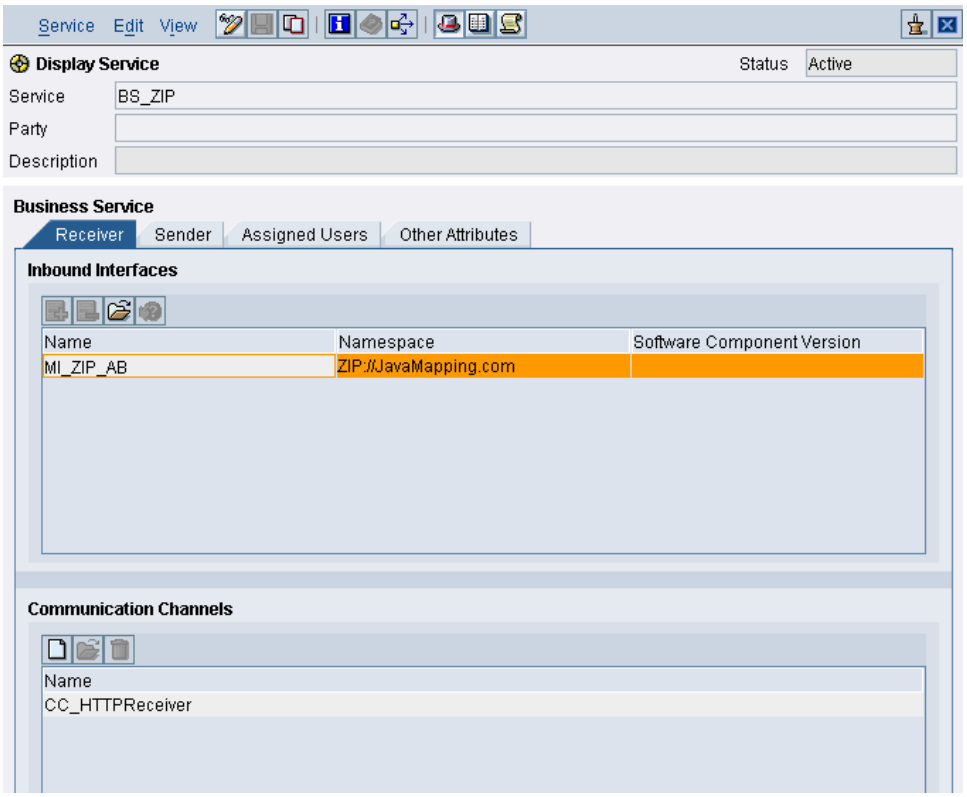
Import this zip file Integration Repository as Import Archive Object which will be used in the Interface Mapping Object. Activate all the objects. Now Design part is completed.

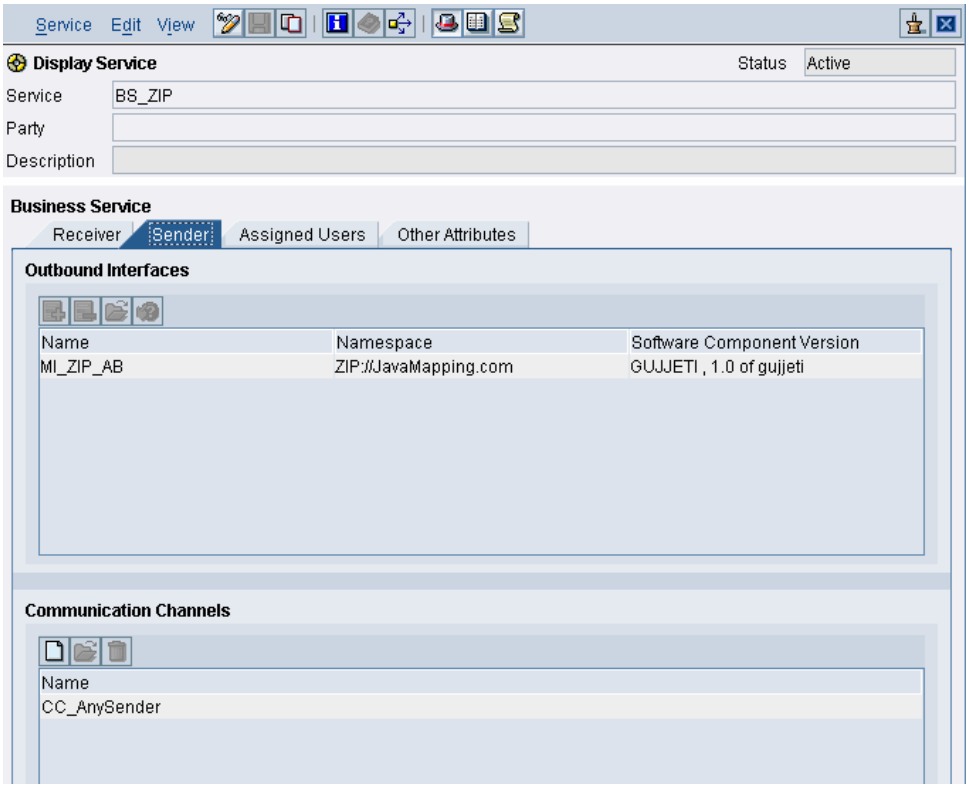
If you want to execute this program as a standalone application for testing purposes, don’t forget to comment the Dynamic Configuration Variables, otherwise it will throw exception.

Configuration Objects in ID

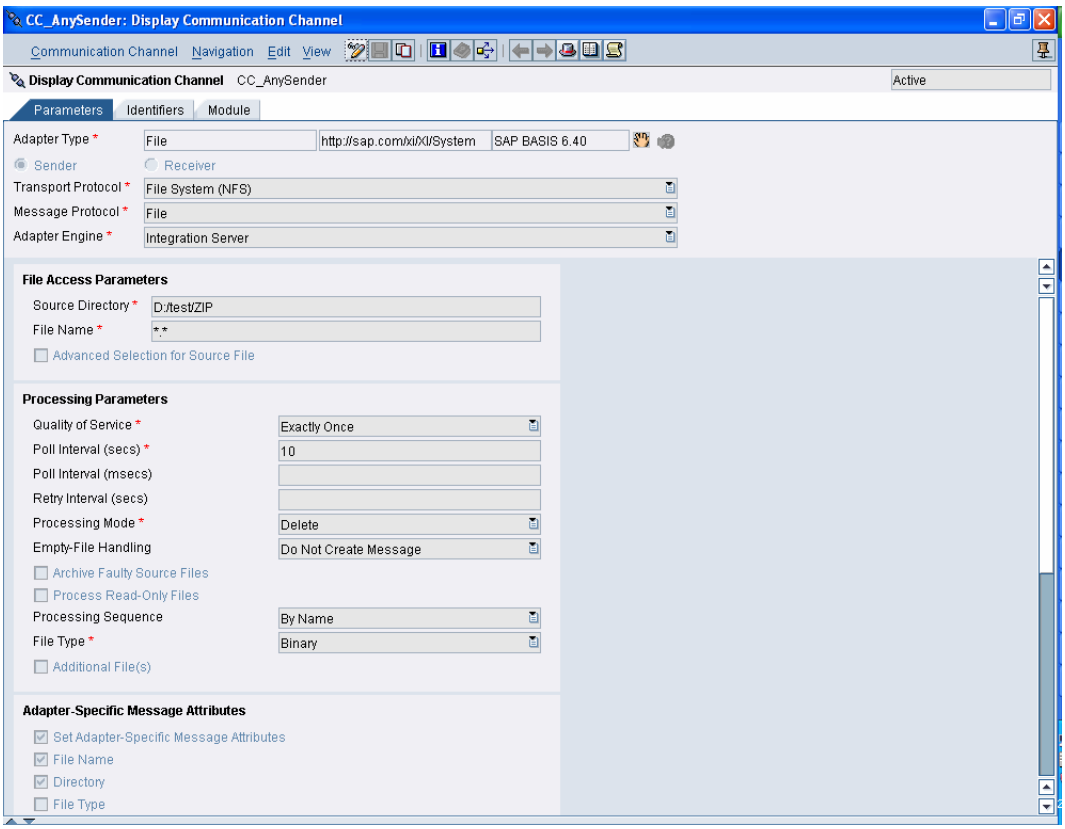
1. Create a new configuration Scenario.
2. Create one Business Service (BS_ZIP) for both File Sender and HTTP Receiver. Add the abstract message interface (MI_ZIP_AB) in both send and receiver tabs of the business service and save. We are using same interface for sender and receiver.
3. Create Communication Channels for File Sender and HTTP Receiver.
4. As usual create sender agreement, Receiver Determination, Interface Determination and Receiver Agreement and associate the Communication Channels for Sender Agreement and Receiver Agreement.




Service Edit View
Display Service
Status Active
Service BS_ZIP
Party
Description
Business Service
Receiver Sender Assigned Users Other Attributes
Inbound Interfaces
Name Namespace Software Component Version
MI_ZIP_AB ZIP://JavaMapping.com
Communication Channels
Name
CC_HTTPReceiver

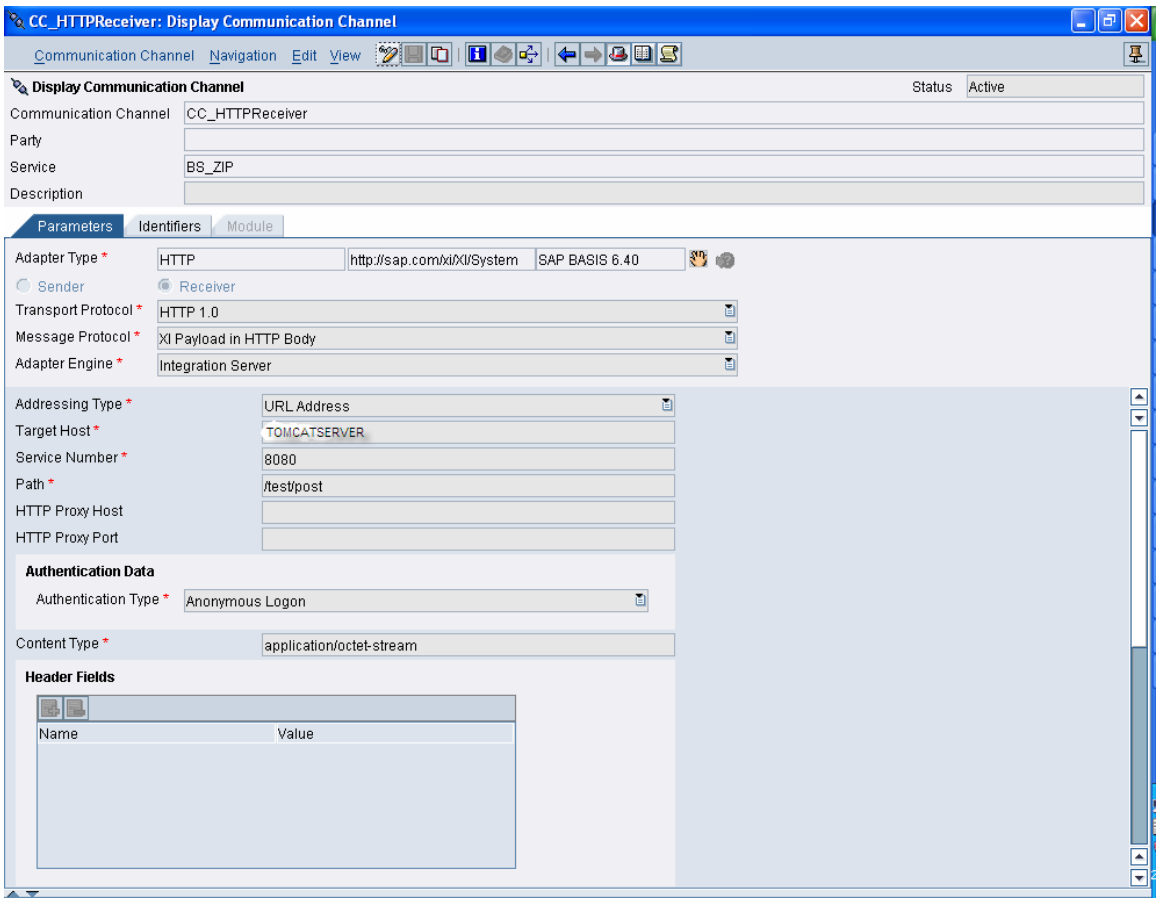

Service Edit View
Display Service
Status Active
Service BS_ZIP
Party
Description
Business Service
Receiver Sender Assigned Users Other Attributes
Outbound Interfaces
Name Namespace Software Component Version
MI_ZIP_AB ZIP://JavaMapping.com GUJJETI , 1.0 of gujjeti
Communication Channels
Name
CC_AnySender

File Adapter Configuration



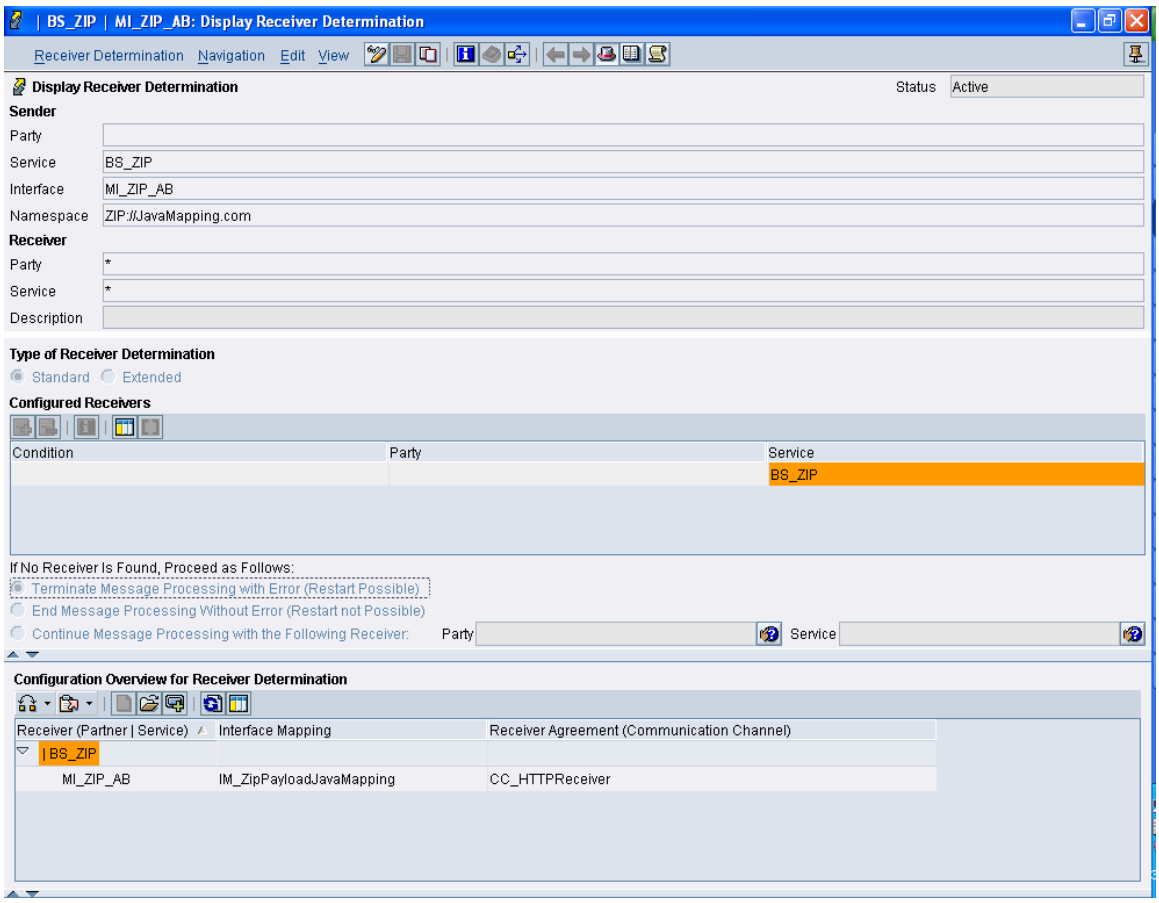
Check the Adapter Specific Message Attributes and also File Name as ZipPayloadJavaMapping program uses this FileName as ZipEntry using Dynamic Configuration. (If the input stream on behalf of File Adapter is not a zip stream, Check ELSE condition in the ZipPayloadJavaMapping program code where we are using the hard coded value of variable ENTRYNAME)

HTTP Adapter Configuration



Specify Content type as application/octet-stream in HTTP Reciever Communication Channel.

Receiver Determination



That's all configuration part is done.

HTTPServlet Code

```
//------------------------------------------------------------------------------CODE---
-------------------------------------------------------
import java.io.*;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.*;
import java.util.zip.*;
import javax.servlet.http.*;
import javax.servlet.*;
public class PostServlet extends HttpServlet
{
   String str = null;
   int i;
   int count = 0;
   static final int BUFFER = 1024*10000;
   byte buf[] = new byte[BUFFER];
         public PostServlet()
         {    }
     public static final void copyInputStream(InputStream in, OutputStream out)
     throws IOException
     {
                byte[] buffer = new byte[1024];
                int len;
                while((len = in.read(buffer)) >= 0)
                  out.write(buffer, 0, len);
                //in.close();
                out.close();
     }
public void doPost(HttpServletRequest httpservletrequest, HttpServletResponse
httpservletresponse)  //throws IOException
         {
         try
         {
                httpservletresponse.setContentType("text/html");
                PrintWriter printwriter = httpservletresponse.getWriter();
                ServletInputStream sis = httpservletrequest.getInputStream();
                ZipInputStream zis = new ZipInputStream((InputStream)sis);
                ZipEntry entry = null;

                count = 0;
while((entry = zis.getNextEntry()) != null) //Business logic should be here.
                  {
                            count++;
                            if(entry.isDirectory())
                                  {
                                    System.err.println("Extracting directory: " +
entry.getName());

                                    (new File("C:\\" + entry.getName())).mkdirs();
                                    continue;
                                  }
                            else
                              {
                                  if ( entry.getName().indexOf("/") != -1)
```

```
                                        {
                                               (new File("C:\\" +
entry.getName().substring(0, entry.getName().lastIndexOf("/")))).mkdirs();
                                               System.err.println("Extracting directory:
" + entry.getName().substring(0, entry.getName().lastIndexOf("/")));
                                               System.err.println("Extracting file: " +
entry.getName());
                                        }
                                        copyInputStream(zis, new BufferedOutputStream(new
FileOutputStream("C:\\" + entry.getName())));
                                        System.err.println("FileExtraction is done: " +
"C:\\" + entry.getName());
                                  }
                  }
                if(count == 0)
                       System.out.println("No Zip Entries To Process");
                zis.close();
          }
          catch (Exception ioe)
                {
                         System.err.println("Unhandled exception:");
                         ioe.printStackTrace();
                         return;
                }
    }
}
//-------------------------------------------------------------------------------CODE---
-----------------------------------------------------
```

HTTPServlet Description

In this servlet I am catching the incoming stream and just unzipping to a local directory of Servlet server machine (TOMCAT). In real time you can add your business logic instead of this. E.g., you might post this data to a database after some transformation with in the servlet code.

The above coding can be still enhanced or modified according to your requirements. You can deploy the above servlet in the Tomcat Server. You can download tomcat at http://tomcat.apache.org/download-55.cgi

You can find deployment procedure at http://tomcat.apache.org/tomcat-5.5-doc/appdev/index.html

I am also providing the following web.xml (deployment descriptor) for the servlet.

Web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">
    <display-name>Unzip Servlet</display-name>
    <description>Unzip Servlet</description>
    <servlet>
        <servlet-name>PostServlet</servlet-name>
        <servlet-class>PostServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>PostServlet</servlet-name>
        <url-pattern>/post/*</url-pattern>
    </servlet-mapping>
</web-app>
```

Execute the Scenario

For testing purposes install a TOMCAT on different machine other than XI server within your landscape. Start tomcat and deploy the above servlet.

Now execute the scenario with any file (binary or text) by copying it in the configured File Directory. Once file is processed by the File Adapter it will be posted to the HTTPServlet and Servlet processes the stream and you can view the status in TOMCAT console.

That's all.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.