Custom FTP with WebSphere Message Broker

Nagesh Subrahmanyam May 25, 2011

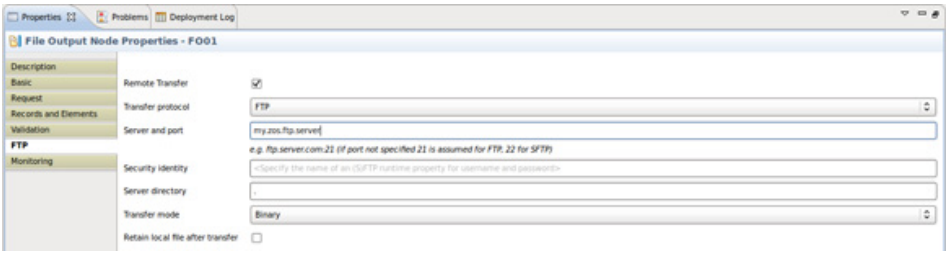
The WebSphere Message Broker FileInput and FileOutput nodes do not support custom FTP. This article shows you how to use Apache Commons Net in a JavaCompute node to enable custom FTP and extend Message Broker to use the full range of FTP capabilities.

Introduction

IBM® WebSphere® Message Broker (hereafter called Message Broker) is an IBM enterprise service bus product for solutions involving message flows. Briefly, these message flows take an input message, apply some transformation to it, and deliver it to an end-point using the built-in Message Broker nodes.

The FileInput and FileOutput nodes are designed to read and write files, respectively, between a file system and an FTP server. However, neither of these nodes support custom FTP commands.

Figure 1



A typical FTP session

The following transcript shows a typical FTP session started from a Linux® machine and going to an MVS® system on a z/OS® server:

Example 1. FTP session from Linux to z/OS

```
nsubrahm@nsubrahm:~$ ftp big.blue.server
Connected to 0.0.0.0.
220-TCPFTP1 IBM FTP CS V1R12 at big.blue.server, 10:03:31 on 2011-03-16.
220 Connection will close if idle for more than 5 minutes.
Name (0.0.0.0:nsubrahm): USERID0
331 Send password please.
Password:
230 USERID0 is logged on.  Working directory is "USERID0.".
Remote system type is MVS.
ftp> site RECFM=FB LRECL=100 BLKSIZE=27900
200 SITE command was accepted
ftp> put Messages.csv 'A.TEST.GDG(+1)'
local: Messages.csv remote: 'A.TEST.GDG(+1)'
200 Port request OK.
125 Storing data set A.TEST.GDG.G0003V00
250 Transfer completed successfully.
336887 bytes sent in 1.17 secs (281.2 kB/s)
ftp> bye
221 Quit command received. Goodbye.
```

This FTP session uses the `SITE` command to pass information for the target dataset. The Message Broker FileOutput node does not have this capability, so this article will show you how to implement it.

The Java solution

The Java™ solution is based on the Apache Commons Net package. As described in the javadoc for `FTPClient`, start by connecting to an FTP server, logging on with your user ID and password, and then use a `get` or `put` command. Here is a sample implementation.

Example 2. Sample implementation of FTP using Apache Commons Net

```
package fileTransferProtocol;

import org.apache.commons.net.ftp.*;
import java.io.*;

public class FileTransferProtocol {
     public static void main (String [ ] args)  {
     String serverName ="my.zos.mainframe"  ;
     String userName ="userid" ;
     String password ="********" ;
     FTPClient ftp = new FTPClient() ;
     //Connect to the server
     try { ftp.connect (serverName) ;
          String replyText =ftp.getReplyString()  ;
          System.out.println (replyText) ;
     }
     catch (Exception  e)  {
               e.printStackTrace () ;
     }
     //Login to the server
     try {   ftp.login (userName, password) ;
               String replyText = ftp.getReplyString() ;
               System.out.println (replyText);
     } catch (Exception e) {
               e.printStackTrace();
     }
     //Tell server that the file will have JCL records
     try {   ftp.site ("RECFM=FB LRECL=140") ;
               String replyText = ftp.getReplyString() ;
```

```
                System.out.println (replyText) ;
        }
        catch (Exception e) {
                e.printStackTrace() ;
        }
        //Submit the job from the text file.
        try {   FileInputStream inputStream = new FileInputStream ("/my/source/file") ;
                ftp.storeFile (serverName,inputStream) ;
                String replyText = ftp.getReplyString() ;
                System.out.println (replyText) ;
        }
        catch (Exception e) {
                e.printStackTrace() ;
        }
        //Quit the server
        try { ftp.quit() ;
        }
        catch (Exception e) {
                 e.printStackTrace() ;
        }
    }
}
```

Using the JavaCompute node

The previous section showed a sample implementation for sending a file from local to a remote file system. This section shows you how to wrap this implementation in a Message Broker JavaCompute node.

The Message Broker information center has plenty of JavaCompute node information. However, one point needs to be emphasized: the sample implementation shown in the previous section requires the Commons Net JAR files to be available on the `CLASSPATH`. Similarly, when using the JavaCompute node, you need to add this dependency on an external JAR file. For more information, see Adding Java code dependencies in the Message Broker information center.

Type of Java class for JavaCompute node

The Message Broker Toolkit provides templates of classes based on the expected functionality of a JavaCompute node. For more information on the templates listed below, see Creating Java code for a JavaCompute node in the Message Broker information center.

- Filter – to filter incoming messages
- Modify – to modify incoming messages
- Create – to create a new message

The choice of a template depends on the particular message flow -- for example, whether the incoming message will be propagated to subsequent nodes after the file transfer, or whether no further action will be taken after the file transfer and therefore no message needs to be propagated.

Java class description

This section describes the Java class that you use to do custom FTP.

Basic inputs for FTP session

The essential items for transferring files through FTP are:

- Server host name or IP address
- User ID and password
- Source and target files

This information should be available for the class that uses FTPClient from Apache Commons Net to actually transfer files. The class shown in this article accepts the above values either as a `java.util.Properties` object or as the name of file that holds these values as name-value pairs. With a `Properties` object, you have the advantage of setting the values dynamically. With a file, no hard-coding of values is required. At the same time, by setting appropriate access control, the file may be read only by the ID under which the broker runs.

Properties object

Since different FTP sessions may require different custom commands, the `Properties` object need not be same all the time. This article describes an implementation where the FTP session is required to deliver a file to a z/OS server by specifying the dataset control block (DCB) and some space values. The name-value pairs for the Properties object are shown below:

Example 3. Sample name-value pairs for Properties object

```
INTF1.server=big.blue.server
INTF1.userid=USERID01
INTF1.password=HeyThere
INTF1.targetDSN='A.DUMMY.GDG(+1)'
INTF1.targetDSN.RECFM=FB
INTF1.targetDSN.LRECL=100
INTF1.targetDSN.BLKSIZE=27900
INTF1.targetDSN.pri=10
INTF1.targetDSN.sec=5
INTF1.targetDSN.unit=CYL
```

Invocation

Here is a sample of a JavaCompute node that invokes the class for custom FTP:

Example 4. Sample JavaCompute node code to invoke class for custom node

```
package customFTP;

import com.ibm.broker.javacompute.MbJavaComputeNode;
import com.ibm.broker.plugin.*;

public class FTPMessageFlow_JC01 extends MbJavaComputeNode {
   public void evaluate(MbMessageAssembly inAssembly) throws MbException {
      MbOutputTerminal out = getOutputTerminal("out");
      MbMessage inMessage = inAssembly.getMessage();

      MbMessage outMessage = new MbMessage(inMessage);
      MbMessageAssembly outAssembly = new MbMessageAssembly(inAssembly,outMessage);

      try {
         /**
          * All logic for this JavaCompute node.
          *
```

```
            * Build Properties object here OR
            * Provide file name here
            *
            * Invoke custom FTP here
            *
            */
         } catch (Exception e) {
            /**
            * Handle exceptions
            */
      } finally {
         outMessage.clearMessage();
      }
}
```

As shown in the above code, the idea is to implement the required business logic and then either prepare the `Properties` object or else pass the file name so that the custom FTP class is invoked.

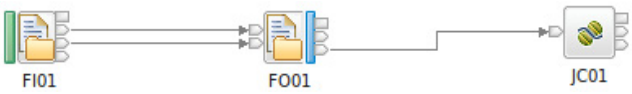
Sample message flows

Here are two sample message flows where the files are sent to a target machine via FTP.

Connected to a FileOutput node

In this message flow, connect the **In** terminal of the **JavaCompute** node from the **End of data** terminal of the **FileOutput** node. In this set-up, after the file has been completely written to, it will be ready for delivery via FTP. For the sake of simplicity, create the file to be transferred by copying from a FileInput node. In practice of course, creating the file may be more complex.

Figure 2. Message flow



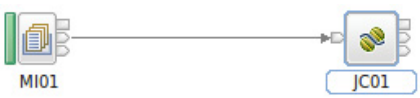
Running the message flow

In the JavaCompute node, mention the file name that holds the name-value pairs. Alternatively, the JavaCompute node may create the Properties object by hand and pass it. Then, create a file name as expected in `FI01`. The message flow can be now packaged into a BAR file and then deployed and run.

Connected to an MQInput node

In this message flow, connect the **In** terminal of the **JavaCompute** node from the **Out** terminal of the **MQInput** node. In this set-up, expect a message with a name of the to-be-delivered file, and use this name to actually deliver the file. In practice of course, creating the message may be more complex.

Figure 3. Message flow



Running the message flow

In the JavaCompute node, mention the file name that holds the name-value pairs. Alternatively, the JavaCompute node may create the Properties object by hand and pass it. Then, put a message with the file name to be delivered. The message flow can be now packaged into a BAR file and then deployed and run.

The zip file that you can download at the bottom of the article has four Message Broker projects – two for each of the message flows mentioned above. In each pair, one has the message flow and the other has the Java code. You can import these zip files into your workspace, then build and deploy them on a broker. Changes specific to your environment maybe needed for the file, such as MQ nodes.

Conclusion

This article has shown you how to use Apache Commons Net in a WebSphere Message Broker JavaCompute node to enable custom commands for FTP, and you can extend this solution to support secure FTP using SSH FTP and FTP-SSL, thus enabling Message Broker to use the full range of FTP capabilities.

Downloadable resources

Description	Name	Size
Code sample	dw.zip	39 KB

Related topics

- **WebSphere Message Broker resources**
  - WebSphere Message Broker developer resources page
    Technical resources to help you use WebSphere Message Broker for connectivity, universal data transformation, and enterprise-level integration of disparate services, applications, and platforms to power your SOA.
  - WebSphere Message Broker product page
    Product descriptions, product news, training information, support information, and more.
  - WebSphere Message Broker V7 information center
    A single Web portal to all WebSphere Message Broker V7 documentation, with conceptual, task, and reference information on installing, configuring, and using your WebSphere Message Broker environment.
  - What's new in WebSphere Message Broker V7
    WebSphere Message Broker V7 provides universal connectivity with its ability to route and transform messages from anywhere to anywhere. Through its simple programming model and a powerful operational management interface, it makes complex application integration solutions much easier to develop, deploy, and maintain. This article describes the major enhancements in V7.
  - Download free trial version of WebSphere Message Broker V7
    WebSphere Message Broker V7 is an ESB built for universal connectivity and transformation in heterogeneous IT environments. It distributes information and data generated by business events in real time to people, applications, and devices throughout your extended enterprise and beyond.
  - WebSphere Message Broker documentation library
    WebSphere Message Broker specifications and manuals.
  - WebSphere Message Broker forum
    Get answers to your technical questions and share your expertise with other Message Broker users.
  - WebSphere Message Broker support page
    A searchable database of support problems and their solutions, plus downloads, fixes, and problem tracking.
  - Apache Commons Net library
    This library helps you implement the client side of many basic Internet protocols. The library provides fundamental protocol access, not higher-level abstractions, so that programmers can construct their own custom implementations.
- **WebSphere resources**
  - developerWorks WebSphere developer resources
    Technical information and resources for developers who use WebSphere products. developerWorks WebSphere provides product downloads, how-to information, support resources, and a free technical library of more than 2000 technical articles, tutorials, best practices, IBM Redbooks, and online product manuals. Whether you're a beginner, an expert, or somewhere in between, you'll find what you need to build enterprise-scale SOA solutions using the open-standards-based WebSphere software platform.
  - developerWorks WebSphere application connectivity developer resources

How-to articles, downloads, tutorials, education, product info, and other resources to help you build WebSphere application connectivity and business integration solutions.
- developerWorks WebSphere SOA and Web services developer resources
  How-to articles, downloads, tutorials, education, product info, and other resources to help you design and build WebSphere SOA and Web services solutions.
- Most popular WebSphere trial downloads
  No-charge trial downloads for key WebSphere products.
- WebSphere forums
  Product-specific forums where you can get answers to your technical questions and share your expertise with other WebSphere users.
- WebSphere on-demand demos
  Download and watch these self-running demos, and learn how WebSphere products and technologies can help your company respond to the rapidly changing and increasingly complex business environment.
- developerWorks WebSphere weekly newsletter
  The developerWorks newsletter gives you the latest articles and information only on those topics that interest you. In addition to WebSphere, you can select from Java, Linux, Open source, Rational, SOA, Web services, and other topics. Subscribe now and design your custom mailing.
- WebSphere-related books from IBM Press
  Convenient online ordering through Barnes & Noble.
- WebSphere-related events
  Conferences, trade shows, Webcasts, and other events around the world of interest to WebSphere developers.

- **developerWorks resources**
  - Trial downloads for IBM software products
    No-charge trial downloads for selected IBM® DB2®, Lotus®, Rational®, Tivoli®, and WebSphere® products.
  - developerWorks blogs
    Join a conversation with developerWorks users and authors, and IBM editors and developers.
  - developerWorks tech briefings
    Free technical sessions by IBM experts to accelerate your learning curve and help you succeed in your most difficult software projects. Sessions range from one-hour virtual briefings to half-day and full-day live sessions in cities worldwide.
  - developerWorks podcasts
    Listen to interesting and offbeat interviews and discussions with software innovators.
  - IBM Education Assistant
    A collection of multimedia educational modules that will help you better understand IBM software products and use them more effectively to meet your business requirements.