



IBM Software Group

WebSphere Partner Gateway

User Exits

Steven Koehler, Consulting IT
Architect, Focused Technologies
Practice for WBI, ISSW

© 2005 IBM Corporation



Agenda

- **User Exit Framework Overview**
- **Choosing The Right User Exit**
- **Designing a User Exit**
- **Code Development**
- **Creating Documentation**
- **Creating User Exit Package**
- **System Deployment**
- **User Exit Testing**
- **Sample User Exit**
- **On-Line References**
- **Questions**

User Exit Framework Overview

- **What is a User Exit?**

WPG users can create a multitude of Java modules using API calls available with the WPG SDK. The points in WPG where these Java modules are invoked are called User Exits.

User Exit Framework Overview

- **What operations can User Exits perform?**
 - Receiving Documents
 - HTTP, FTP, File System, Database, eMail, ...
 - Processing Documents
 - Encryption, Signing, Packaging, ...
 - Sending Documents
 - HTTP, FTP, File System, Database, eMail, ...

Let us investigate each of these three areas in depth...

User Exit Framework Overview

- **Receiving Documents**

Documents can be pulled or pushed into WPG with custom developed code. This type of User Exit is called a Receiver. Receiver User Exits are available in the Community Console as a Target.

User Exit Framework Overview

- **Processing Documents**

These types of User Exits are commonly used for custom signing and encryption methods as well as modifying data.

User Exit Framework Overview

- **Sending Documents**

Documents can be sent out from WPG over a custom design transport. This type of User Exit is available in the Community Console as a Gateway.

Choosing The Right User Exit

- **With many types of User Exits, study the business needs of how the system must interact and use this as the guide when selecting the combination of User Exits to be designed and deployed.**

Design

- **First Steps**

- Review product architecture
- Review pertinent standards, such as FTP, ACIF, AS

- **Second Steps**

- Develop a written design document
- Consider implementing Java design standards

Development

- **Any Java development tool can be used, but use of WSAD allows integrated debugging**
- **Source control**
- **JavaDoc**
- **JUnit**
- **Separation of code for sharing in different WPG modules**
- **Use a common User Exit project template**

Documentation Is Also For Internal Use

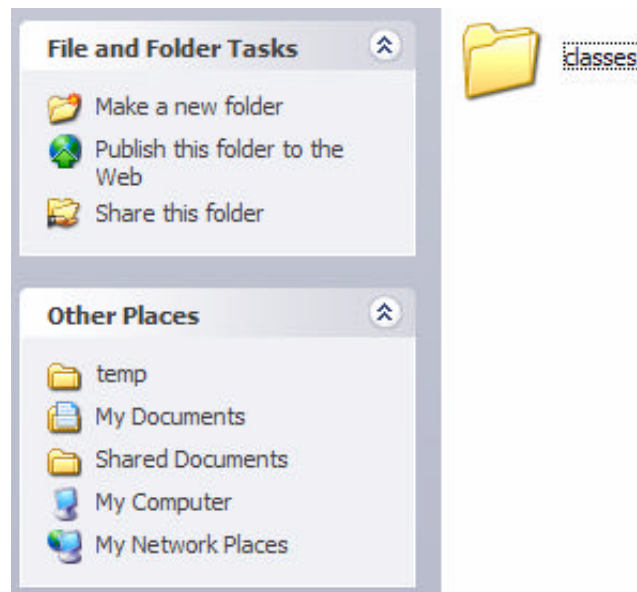
- **Adequate user documentation should be written and delivered as part of the packaging. This documentation should include at least the following topics:**
 - Installation
 - Configuration
 - Testing
 - Troubleshooting
- **JavaDocs**
- **Deliver all documentation in a portable format, such as PDF or ASCII text file.**

Packaging

- **A complete User Exit will consist of the following components:**
 - Compiled class files
 - Deployment descriptor file (XML)
 - Documentation

Deployment

- **Depending of type of User Exit, the class files will be copied to a certain directory.**



Deployment

- **The descriptor file is loaded into the Community Console.**

Import Transport Welcome, Hub Administrator

[Create Target](#) [Delete Transport Type](#) [List](#) [Help](#)

Load a Transport XML File

File:

Commit to Database: Yes No

Overwrite data: Yes No

Deployment

Target Details Welcome, Hub Administrator

[List](#) [Help](#)

Target Name *

Status Enabled Disabled

Description

Transport *

Target Configuration

Gateway Type: *

Subscribe File:

Polling Seconds:

Configuration Point Handlers: ▼

Deployment

- **After files are deployed to the classes directory, the component will need to be restarted to recognize any class file changes.**

Testing

- **Unit Tests**
- **Developer Tests**
- **QA**
- **Functional Testing**

Sample User Exit

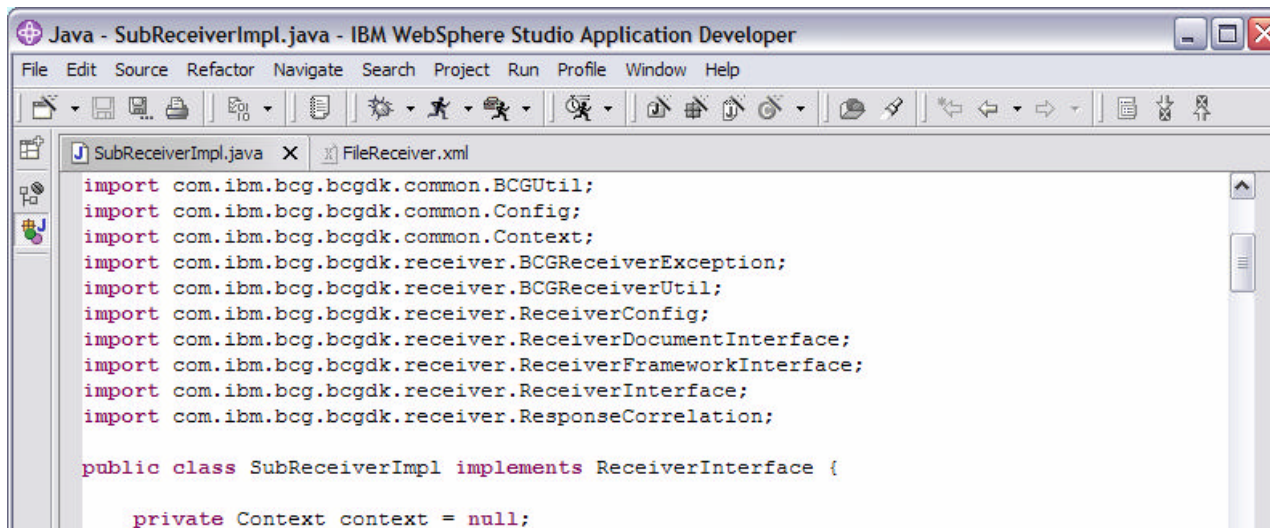
- **This sample is a Receiver User Exit that receives a file from a fixed location on a fixed interval. The file will not be deleted from the file system. This type of User Exit is called a subscription receiver.**

Sample User Exit

- **Open a development tool and create a standard Java application project.**
- **Import necessary libraries, including the following:**
 - log4j
 - bcg-sdk
 - junit

Sample User Exit

- **Create the initial class that implements the ReceiverInterface**



```
Java - SubReceiverImpl.java - IBM WebSphere Studio Application Developer
File Edit Source Refactor Navigate Search Project Run Profile Window Help
SubReceiverImpl.java x FileReceiver.xml
import com.ibm.bcg.bcgdk.common.BCGUtil;
import com.ibm.bcg.bcgdk.common.Config;
import com.ibm.bcg.bcgdk.common.Context;
import com.ibm.bcg.bcgdk.receiver.BCGReceiverException;
import com.ibm.bcg.bcgdk.receiver.BCGReceiverUtil;
import com.ibm.bcg.bcgdk.receiver.ReceiverConfig;
import com.ibm.bcg.bcgdk.receiver.ReceiverDocumentInterface;
import com.ibm.bcg.bcgdk.receiver.ReceiverFrameworkInterface;
import com.ibm.bcg.bcgdk.receiver.ReceiverInterface;
import com.ibm.bcg.bcgdk.receiver.ResponseCorrelation;

public class SubReceiverImpl implements ReceiverInterface {

    private Context context = null;
```

Sample User Exit

- **The ReceiverInterface has the following methods that must be implemented:**

- ▲ `init(Context, ReceiverConfig)`
- ▲ `processResponse(ResponseCorrelation, ReceiverDocumentInterface)`
- ▲ `refreshConfig(ReceiverConfig)`
- ▲ `startReceiving()`
- ▲ `stopReceiving()`

We will explore each of these methods in detail...

Sample User Exit

```
public void init(Context context, ReceiverConfig receiverConfig)
    throws BCGReceiverException {
    this.context = context;
    this.receiverConfig = receiverConfig;

    List rcvTrgtConfigs = receiverConfig.getTargetConfigs();
    Iterator it = rcvTrgtConfigs.iterator();

    for (Config tgtConfig = null; it.hasNext(); tgtConfig = null) {
        tgtConfig = (Config) it.next();
        ConfigLocation = (String) tgtConfig.getAttribute(CONFIG_LOCATION);
        PollingInterval =
            Integer.parseInt(
                (String) tgtConfig.getAttribute(POLLING_INTERVAL));

        transportType = (String) tgtConfig.getAttribute("TRANSPORTTYPE");
        target = (String) tgtConfig.getAttribute("RCVCONFIGNAME");
    }
}
```

Sample User Exit

```
public void processResponse (  
    ResponseCorrelation responsecorrelation,  
    ReceiverDocumentInterface receiverdocumentinterface)  
    throws BCGReceiverException {  
  
}
```

Sample User Exit

```
public void refreshConfig(ReceiverConfig newRecvConfig)
    throws BCGReceiverException {
    List rcvTrgtConfigs = receiverConfig.getTargetConfigs();
    Iterator it = rcvTrgtConfigs.iterator();

    for (Config tgtConfig = null; it.hasNext(); tgtConfig = null) {
        tgtConfig = (Config) it.next();
        ConfigLocation = (String) tgtConfig.getAttribute(CONFIG_LOCATION);
        PollingInterval =
            Integer.parseInt(
                (String) tgtConfig.getAttribute(POLLING_INTERVAL));
    }
}
```

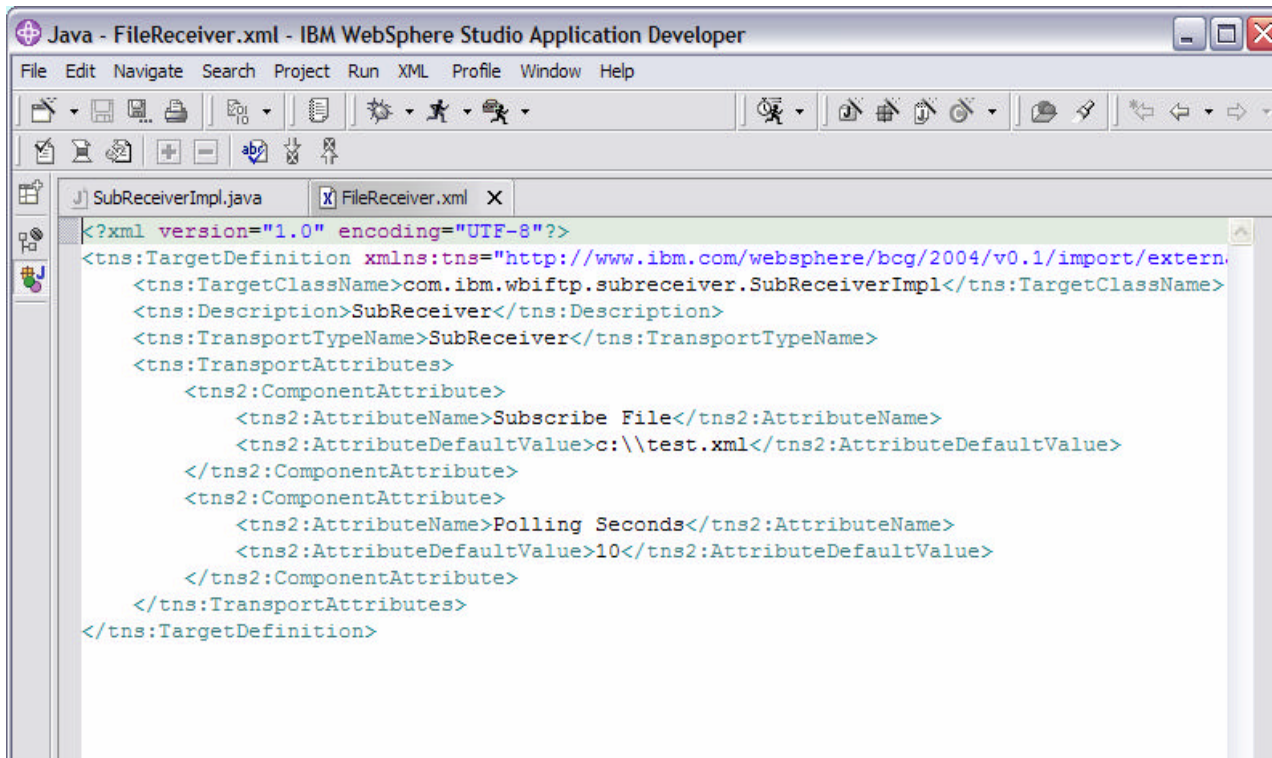

Sample User Exit

```
public void startReceiving() throws BCGReceiverException {  
    isEnabled = true;  
}
```

Sample User Exit

```
public void stopReceiving() throws BCGReceiverException {  
    isEnabled = false;  
}
```

Sample User Exit



```
<?xml version="1.0" encoding="UTF-8"?>
<tns:TargetDefinition xmlns:tns="http://www.ibm.com/websphere/bcg/2004/v0.1/import/extern.
  <tns:TargetClassName>com.ibm.wbiftp.subreceiver.SubReceiverImpl</tns:TargetClassName>
  <tns:Description>SubReceiver</tns:Description>
  <tns:TransportTypeName>SubReceiver</tns:TransportTypeName>
  <tns:TransportAttributes>
    <tns2:ComponentAttribute>
      <tns2:AttributeName>Subscribe File</tns2:AttributeName>
      <tns2:AttributeValue>c:\\test.xml</tns2:AttributeValue>
    </tns2:ComponentAttribute>
    <tns2:ComponentAttribute>
      <tns2:AttributeName>Polling Seconds</tns2:AttributeName>
      <tns2:AttributeValue>10</tns2:AttributeValue>
    </tns2:ComponentAttribute>
  </tns:TransportAttributes>
</tns:TargetDefinition>
```

Sample User Exit

```
ReceiverFrameworkInterface rcvrImpl = null;
ReceiverDocumentInterface rcvrDoc =
    BCGReceiverUtil.createReceiverDocument();
rcvrDoc.setDocument(fileLocation);
ReceiverDocumentInterface rcDoc[] = null;
rcvrImpl = BCGReceiverUtil.getReceiverFramework();
rcDoc = rcvrImpl.preProcess(transportType, target, rcvrDoc);

rcvrDoc.setAttribute(
    "ReceiverDestinationType",
    "Production");
rcvrDoc.setAttribute("requestURI", "");
rcvrDoc.setAttribute(
    "MsgLengthIncHeaders",
    Long.toString(fileLocation.length()));
rcvrDoc.setAttribute(
    "content-length",
    Long.toString(fileLocation.length()));
rcvrDoc.setAttribute(
    "x-aux-in-file-name",
    fileLocation.getName());
rcvrDoc.setAttribute(
    "OriginalFileName",
    fileLocation.getAbsolutePath());

for (int j = 0; j < rcDoc.length; j++)
    rcvrImpl.process(transportType, rcDoc[j]);
```

References

- **B2B Solutions using WebSphere BI Connect Version 4.22**
 - <http://www.redbooks.ibm.com/redbooks/pdfs/sg246355.pdf>
- **WPG 6.0 Programmer's Guide**
 - <http://www-306.ibm.com/software/integration/wspartnergateway/library/infocenter>
- **WSAD**
 - <http://www-306.ibm.com/software/awdtools/developer/application/index.html>
- **JUnit**
 - <http://www.junit.org/index.htm>
- **Log4J**
 - <http://logging.apache.org/log4j>



Questions?