IBM

SMP/E for z/OS:

RECEIVE ORDER and the Automated Service Request Server Interface

Author: Kurt Quackenbush
IBM, z/OS SMP/E Design
kurtq@us.ibm.com
Date: June 21, 2019

This publication applies to IBM SMP/E for z/OS, V3R6 (program number 5655-G44) and to all subsequent releases and modifications until otherwise indicated in new editions.

Table of Contents

Chapter 1. Overview

The SMP/E RECEIVE ORDER command submits requests for PTFs and HOLDDATA to a remote server and automatically downloads the packages that result when those requests are fulfilled. SMP/E uses HTTPS to communicate with the server and any of HTTP, HTTPS, FTP or FTPS to download the package files.

To do:

Insert a picture showing the interaction between SMP/E and the logical servers, and generally provide more overview information.

To do:

Discuss briefly GIMZIP packages, and downloading them. Some notes:

1. The resulting content for an order submitted to the server is a GIMZIP package which can be downloaded to the client z/OS.
2. SMP/E expects to use a userid and password for authenticating with the download server.
3. SMP/E can use any of the FTP, FTPS, HTTP, or HTTPS protocols to download the GIMZIP package files.

There are three methods supported by the Automated Service Request server used by SMP/E:

Method	Description
submitUpdateOrder	The submitUpdateOrder method is used to submit a request to the server for a software update. The server will create a unique order, identified by an id, to describe this request. See chapter 2 “submitUpdateOrder” on page 6.
get	The get method is used to obtain current status for a particular software update order. See chapter 3 “get” on page 12.
close	The close method is used to tell the server the client has finished processing a particular software update order , and therefore it can be closed. See chapter 4 “close” on page 17.

Each method has a corresponding client request message and a server response message, which are described in detail later in this document. The server response messages described here are the minimal responses your server must provide to SMP/E. The IBM server is quite verbose and provides much more information than is shown in this document, but it is ignored by SMP/E. I have identified in this document the minimum required responses, but you are free to add additional information as long as it is proper XML syntax.

1.1 Client and server transaction sequence

Here's the typical sequence between the client (SMP/E) and the Automated Service Request server.

Step	SMP/E	Automated Service Request server
1	Build a software inventory by analyzing the FMID and PTF content of the global and target zones.	
2	Extract the named certificate from the security manager data base.	
3	POST a submitUpdateOrderRequest message to describe the requested content.	
4		Create a new order, start processing for the order, and respond with a submitUpdateOrderResponse message containing a unique transaction ID for the order, state="processing", and an appropriate process time value.
5	Wait until the processTime has passed, and then POST a getRequest message specifying the order's transaction ID.	
6		If the order processing is not yet complete, then respond with a getResponse message containing state="processing" and an appropriate process time value.
7	Wait until the processTime has passed, and then POST a getRequest message specifying the order's transaction ID.	
8		If the order processing is complete, then respond with a getResponse message containing state="complete.available" and dataPort information for the GIMZIP package containing the order content so the client can download the package files.
9	Download the GIMZIP package files which contain the order content using FTP, FTPS, HTTP, or HTTPS.	
10	POST a closeRequest message specifying the order's transaction ID to indicate the client's processing for this order is complete.	
11		"Close" the order and respond with a closeResponse message containing state="closed".

1.2 Client certificate

SMP/E provides user identity information to the Automated Service Request server in the form of an x.509 certificate. The certificate is a tamper proof container for whatever user identity information is needed by the server. The certificate is stored in a security manager data base, and SMP/E extracts the certificate out of this data base during RECEIVE ORDER command processing. Every request message SMP/E sends to the server contains this certificate, in ANS1.DER base64 encoding.

As a service provider, when a customer registers to use your Automated Service Request server, you are expected to generate a certificate for the customer, containing whatever information is appropriate. The customer then stores this certificate in their security manager data base on z/OS.

The certificates generated by IBM's Shopz application, for example, use the Subject field in the certificate to contain the IBM Customer Number value, and other information to identify the user. The IBM Customer Number identifies the customer to IBM's Automated Service Request server.

For example, these are the Issuer and Subject fields from an IBM generated certificate:

Issuer	C=US ST=Colorado L=Boulder O=IBM Corporation OU=IBM Automated Delivery Request
Subject	C=US O=zOSService OU=425268897 CN=IBM Customer Number: 4606985

It is important to understand this client certificate is **NOT** used for SSL client authentication between SMP/E and the remote server. The certificate is used solely as a container for customer identity information that is provided with each request message to the server.

Chapter 2. submitUpdateOrder

The submitUpdateOrder method is used to submit a software update order to the Automated Service Request server. A software update order is a request to obtain SMP/E consumable PTFs and/or HOLDDATA. The server will create a unique order, identified by an id, to describe this order.

2.1 URL format

The following shows the format of the URL for this request:

```
https://<host><:port></path>/ServiceProvider
```

where:

- "`https://<host><:port>`" specifies the server address and port.
- "`</path>`" identifies the path to the provider on the server.
- "`/ServiceProvider`" indicates the provider for the submitUpdateOrder method.

The https://<host><:port>/<path> portion of the URL is specified by the user on the "url" attribute in the <ORDERSERVER> tag used by the SMP/E RECEIVE ORDER command. SMP/E adds the further qualification, "ServiceProvider" in this case, as appropriate for the request.

2.2 Request content

The request will include a submitUpdateOrderRequest message to describe the content for the requested order. Like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Header>
<header xmlns="http://www.ibm.com/xmlns/b2b/ecc/v1.0">
  <targetURI>uri</targetURI>
  <credentials>
    <certificate>certificate-data</certificate>
  </credentials>
</header>
</soap:Header>

<soap:Body>
  <submitUpdateOrderRequest
    xmlns="http://www.ibm.com/xmlns/b2b/ecc/v1.0/UpdateOrder”>
    <subject xmlns="http://www.ibm.com/xmlns/b2b/ecc/v1.0">
      <product>ibm/zOS/PlatformSoftware</product>
      <reference>SoftwareInventory</reference>
    </subject>
    <updateId>update-type</updateId>
    <includeRequisites>true</includeRequisites>
    <includeSupersedes>false</includeSupersedes>
    <deliveryPreference>
      <medium>electronic</medium>
      <dataPortPreference xmlns="http://www.ibm.com/xmlns/b2b/ecc/v1.0">
        <transport>FTP</transport>
        <transport>FTPS</transport>
      </dataPortPreference>
    </deliveryPreference>
```

```
  optional-software-inventory

  </submitUpdateOrderRequest>
</soap:Body>

</soap:Envelope>
```

where:

XML	Description
<targetURI>	The universal resource identifier (URI) of the service provider on the server. This value is the same as the URL for the HTTP request, https://host:port/path/ServiceProvider.
<certificate>	An x.509 certificate in ANS1.DER base64 encoding. The certificate describes the credentials for the user that is making a request to the server. The certificate is stored in a security manager data base, and SMP/E extracts the named certificate from the security manager data base and includes it in the request to the server.
<subject>	Identifies the scope of the updates being requested. For SMP/E requests, the subject always includes <product>ibm/zOS/PlatformSoftware</product> no matter what products are installed, and may also include a reference to a software inventory attachment.
<reference>	Identifies the attachment entity within the XML request that describes the software inventory. The software inventory attachment within the request has a URI of "SoftwareInventory". This reference to a software inventory attachment is included in the request only for orders requesting one or more PTFs. Orders that request only HOLDDATA do not include a software inventory attachment, and therefore, do not include this <reference> tag. See 2.2.1 "Software inventory" on page 8 for details.
<updateId>	Identifies the specific updates being requested. The values specified here are determined by the value of the CONTENT operand on the SMP/E RECEIVE ORDER command. · all – All available PTFs applicable to the software inventory. · critical – All available PTFs applicable to the software inventory that resolve a critical problem. · recommended – All available PTFs applicable to the software inventory that are recommend or resolve a critical problem. · zOS/PTF/ppppppp – where ppppppp is a single PTF identifier. · zOS/APAR/aaaaaaa – where aaaaaaa is a single APAR identifier. · zOS/HOLDDATA – The current HOLDDATA. If PTFS or APARS were specified on the CONTENT operand of the

XML	Description
	RECEIVE ORDER command, then multiple <updateId> elements may be specified to identify all requested PTFs or APARs.

2.2.1 Software inventory

The software inventory attachment in the request is optional, and its exact form is dependent on the inventory format specified on the SMP/E RECEIVE ORDER command. If HOLDDATA was specified on the CONTENT operand of the SMP/E RECEIVE ORDER command, then no software inventory is produced or included in the order request. However, if any other value was specified on the CONTENT operand of the SMP/E RECEIVE ORDER command, then a software inventory is produced and included in the submitUpdateOrderRequest message.

2.2.1.1 Inventory = all

If inventory="all" was specified in the SMP/E ORDERSERVER data set for the RECEIVE ORDER command, then the software inventory will be in a generic form that identifies all installed FMIDs and PTFs regardless of their naming convention. The following XML is used to describe the software inventory:

```
<attachment xmlns="http://www.ibm.com/xmlns/b2b/ecc/v1.0">
  <thisURI>SoftwareInventory</thisURI>
  <description>SMPE Software Inventory</description>
  <descriptor>data/inventory.software</descriptor>
  <type>application/vnd.ibm.zos.gimxsid.all</type>
  <gimxsidSwInventory>
    <gimxsidVersion>vv.rr.mm.pp</gimxsidVersion>
    <date>yyyy-mm-ddThh:mm:ss</date>
    <fmidList>
      <id>fmid-name</id>
      <id>fmid-name</id>
    </fmidList>
    <ptfList>
      <id>ptf-name</id>
      <id>ptf-name</id>
    </ptfList>
  </gimxsidSwInventory>
</attachment>
```

where:

Property	Description
<gimxsidVersion>	The version, release, modification, and PTF service level of the SMP/E GIMXSID service routine that is used to produce the software inventory information.
<date>	The date and time when the SMP/E GIMXSID service routine produced the software inventory information.
<fmidList>	The list of FMIDs for all of the FUNCTION SYSMODs received in the global zone, and installed in the target zones defined for the software instance. Each FMID is described by a separate <id> tag.
<ptfList>	The list of PTF SYSMODs received in the global zone, or applied in

Property	Description
	all target zones defined for the software instance in which they are applicable. Each PTF is described by a separate <id> tag.

The software inventory information in the submitUpdateOrderRequest is the same as that produced by the SMP/E GIMXSID service routine. For more details see the GIMXSID software inventory data service routine topic of the "SMP/E for z/OS Reference".

2.2.1.2 Inventory = ibm

If inventory="ibm" was specified in the SMP/E ORDERSERVER data set or defaulted for the RECEIVE ORDER command, then the software inventory will be in the form expected by the IBM Automated Service Request server that identifies all installed FMIDs and only PTFs whose IDs match the naming convention used by IBM. The following XML is used to describe the software inventory:

```
<attachment xmlns="http://www.ibm.com/xmlns/b2b/ecc/v1.0">
  <thisURI>SoftwareInventory</thisURI>
  <description>SMPE Software Bitmap Inventory</description>
  <descriptor>data/inventory.software</descriptor>
  <type>application/vnd.ibm.zos.smpebitmap</type>
  <data encoding="base64">inventory-data</data>
</attachment>
```

where:

Property	Description
<data>	The base64 encoded software inventory, containing a list of FMIDs for all of the FUNCTION SYSMODs received in the global zone, and installed in the target zones defined for the software instance, and a list of PTF SYSMODs received in the global zone, or applied in all target zones defined for the software instance in which they are applicable.

The software inventory information in the submitUpdateOrderRequest is the same as that produced by the SMP/E GIMXSID service routine. For more details see the GIMXSID software inventory data service routine topic of the "SMP/E for z/OS Reference".

2.3 Expected response

After processing the submitUpdateOrderRequest message, the Automated Service Request server is expected to respond with an HTTP 200 response code and a submitUpdateOrderResponse message, like this:

```
<Envelope>
<Body>

<submitUpdateOrderResponse>
  <thisURI>uri</thisURI>
  <state>state</state>
  <updateIdExpansion>level</updateIdExpansion>
```

```
  <processTime>period</processTime>
</submitUpdateOrderResponse>

</Body>
</Envelope>
```

where:

Property	Description
<thisURI>	The Universal Resource Identifier (URI) for the order on the server. The expected value is of the form: https://host:port/path/UpdateOrder?orderid=order-id where “order-id” is the identifier for the subject order on the server. The order-id value may be up to 10 alpha-numeric characters.
<state>	The state, or status, for the order on the server. For submitUpdateOrderResponse the state may be one of: • processing – the order is being actively processed by the server. • error – an error was detected by the server. See 5 “Faults and Errors” on page 19 for details.
<updateIdExpansion>	Used only when the updateId value in the request is “recommended”. In this case <updateIdExpansion> will indicate a service level for the recommended PTFs that will be supplied in the order content. The value is expected in either of the following forms: 1. zOS/RSU/nnnn, where nnnn is a four-digit year and month Recommended Service Update (RSU) identifier. For example, zOS/RSU/1707 corresponds to the RSU1707 level. Or, 2. zOS/ssssssss, where ssssssss is any 1-8 character value representing the level. For example, zOS/JULY2017. **Note:** The fix for SMP/E APAR IO25034 (PTF UI01835) modifies SMP/E processing to accept the level specification in the form zOS/ssssssss. Prior to this APAR, SMP/E only accepted “zOS/RSU/” and would throw an error if some other value was found.
<processTime>	Expected time for the server to process and complete the subject order. This value is a duration (also called Period) measured from when the service provider creates the response document. SMP/E will wait for this length of time (for this period), before it queries the server for status on the subject order. The format of the value is PTsecondsS where seconds is from 1 to 10 decimal digits. For example, <processTime>PT120S</processTime> means 120 seconds.

2.4 Server processing

After receiving a submitUpdateOrderRequest message, the Automated Service Request server is expected to respond synchronously with a response message, and also start an asynchronous process on the server to satisfy the UpdateOrder request.

If an error is detected with the submitUpdateOrderRequest message the server must respond with either:

- A submitUpdateOrderResponse message containing <state>error</state> and the appropriate fault information.
- A Fault message containing the appropriate fault information.

See Chapter 5 “Faults and Errors” on page 19 for details.

The asynchronous processing to satisfy the UpdateOrder request uses the information provided by the request message and the server's PTF catalog. If the <updateId> in the submitUpdateOrderRequest message specified zOS/HOLDDATA, then the client requested only HOLDDATA and no PTFs. Therefore the server should create a GIMZIP package containing only HOLDDATA. If the <updateId> in the submitUpdateOrderRequest message specified one of the PTF content choices, then the server should use that value, with the provided software inventory information, to produce a GIMZIP package containing the appropriate PTFs, and HOLDDATA.

The software inventory identifies the FMIDs and PTFs that are installed in the SMP/E target zones, and received in the global zone. The FMIDs described in the software inventory provide scope for a PTF selection process, and the PTFs described in the software inventory identify which PTFs the client already has available and therefore should not be included in the resultant GIMZIP package.

The logical HOLDDATA and PTF content for a GIMZIP package may each be composed of multiple physical files. That is, the HOLDDATA content in a GIMZIP package may be composed of one or more sequential data sets or UNIX files containing ++HOLD statements. The PTF content in a GIMZIP package may be composed of one or more sequential data sets or UNIX files containing the Modification Control Statements (MCS) for the PTFs that satisfy the requested selection criteria, and the appropriate ++ASSIGN statements for each of the PTFs. For more information about GIMZIP packages, see the GIMZIP packaging service routine topic in the “SMP/E for z/OS Reference”.

After the HOLDATA and/or the PTF sequential data sets or UNIX files have been created, and the GIMZIP service routine has been run to generate the GIMZIP package, the SHA-1 hash value described by the <?PKGHASH> declaration in the GIMPAF.XML file must be obtained so it can be provided to the client in an eventual getResponse message. The package directory and files should be placed on an appropriate download server where the client can access the files and download them to its local z/OS system. This download server may be the same as the host for the UpdateOrder service provider, or it may be a completely different host.

Chapter 3. get

The get method is used to obtain the current status for an update order from the Automated Service Request server.

3.1 URL format

The following shows the format of the URL for this request:

```
https://<host><:port></path>/UpdateOrder?orderid=<order-id>
```

where:

- "`https://<host><:port>`" specifies the server address and port.
- "`</path>`" identifies the path to the provider on the server.
- "`/UpdateOrder`" indicates the provider for the UpdateOrder method.
- "`<order-id>`" is the unique identifier for the subject order.

The https://<host><:port>/<path> portion of the URL is specified by the user on the "url" attribute in the <ORDERSERVER> tag used by the SMP/E RECEIVE ORDER command. SMP/E adds the further qualification, "UpdateOrder?orderid=<order-id>" in this case, as appropriate for the request.

3.2 Request content

The request will include a getRequest message, like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Header>
<header xmlns="http://www.ibm.com/xmlns/b2b/ecc/v1.0">
  <targetURI>uri</targetURI>
  <credentials>
    <certificate>certificate-data</certificate>
  </credentials>
</header>
</soap:Header>

<soap:Body>
  <getRequest xmlns="http://www.ibm.com/xmlns/b2b/ecc/v1.0/UpdateOrder">
  </getRequest>
</soap:Body>

</soap:Envelope>
```

where:

Property	Description
<targetURI>	The universal resource identifier (URI) of the service provider on the server. This value is the same as the URL for the HTTP request, https://host:port/path/UpdateOrder?orderid=order-id where "order-id" is the unique identifier for the subject order.
<certificate>	An x.509 certificate in ANS1.DER base64 encoding. The certificate describes the credentials for the user that is making a request to the

Property	Description
	server. The certificate is stored in a security manager data base, and SMP/E extracts the named certificate from the security manager data base and includes it in the request to the server.

3.3 Expected response

After processing the getRequest message, the Automated Service Request server is expected to respond with an HTTP 200 response code and a getResponse message. If the order content is not yet ready for download, then the expected getResponse message is like this:

```
<Envelope>
<Body>

<getResponse>
<state>order-state</state>
<processTime>period</processTime>
</getResponse>

</Body>
</Envelope>
```

If the order content is ready for download, then the expected getResponse message is like this:

```
<Envelope>
<Body>

<getResponse>
  <state>complete.available</state>
  <processTime>PT0S</processTime>
  <attachment>
    <dataPort>
      <dataURI>data-uri</dataURI>
      <authentication>
        <username>user-id</username>
        <password>password</password>
      </authentication>
      <hash>gimzip-hash</hash>
    </dataPort>
  </attachment>
</getResponse>

</Body>
</Envelope>
```

where:

Property	Description
<state>	The state, or status, for the order on the server. For getResponse the state may be one of: • processing – the order is actively being processed by the server. • complete.available – indicates the server finished processing the

Property	Description
	order and the order content files are ready for download. • complete.noUpdates – the server finished processing the order, but no PTFs satisfied the selection criteria. • error – an error was detected by the server. See 5 “Faults and Errors” on page 19 for details.
<processTime>	Expected time remaining for the server to process and complete the subject order. This value is a duration (also called Period) measured from when the service provider creates the response document. SMP/E will wait for this length of time before it queries the server for status on the subject order. The format of the value is PTsecondsS where seconds is from 1 to 10 decimal digits. For example, <processTime>PT120S</processTime> means 120 seconds.
<dataURI>	The universal resource identifier (URI) which describes the location of the order content. Specifically, it is the location of the GIMPAF.XML file for the GIMZIP package that contains the requested PTFs and/or HOLDDATA. The expected format of the value may be any of the following forms: • `ftp://host:port/path/GIMPAF.XML` • `ftps://host:port/path/GIMPAF.XML` • `http://host:port/path/GIMPAF.XML` • `https://host:port/path/GIMPAF.XML` where “host” is the host name or IP address for the server where the files reside. This host may be the same as the host where the Automated Service Request server is running, or a completely different server. “port” is optional, and “path” is the directory on the identified server where the GIMPAF.XML file resides. The protocol specified in this value is ignored and does not imply how the order content files will actually be downloaded. **Note:** The fix for SMP/E APAR IO25034 (PTF UI01835) modifies SMP/E processing to accept any of the “ftp” “ftps” “http” and “https” protocols in the specified URI. Prior to this APAR, SMP/E only accepted “ftp” and would throw an error if some other value was found.
<username>	The user-id that is allowed access to the order content files on the server.
<password>	The password for the user-id that is allowed access to the order content files on the server.
<hash>	The SHA-1 package hash value for the GIMZIP package containing the order content, encoded into base64. After SMP/E downloads the GIMPAF.XML file, it calculates a SHA-1 hash value for the file and compares it to this value provided by the server to ensure the file has

Property	Description
	not been modified after it was generated or tampered with during transit. See 3.3.1 “<hash>” on page 15 for details.

3.3.1 <hash>

The hash value in the server's response is the SHA-1 package hash value for the GIMZIP package containing the order content, encoded into base64. After SMP/E downloads the GIMPAF.XML file, it calculates a SHA-1 hash value for the file and compares it to this value provided by the server to ensure the file has not been modified after it was generated or tampered with during transit.

A 40-character SHA-1 hash value is generated by the GIMZIP service routine and recorded in the <?PKGHASH> declaration within the GIMPAF.XML file. The server must encode the EBCDIC 40-character SHA-1 hash value into base64 before providing the value in the server response.

To encode the GIMZIP 40-character SHA-1 hash value, the server must first convert the hash value's hexadecimal characters to their binary equivalents, and then encode the binary value into base64. Here is a simple Java program to perform this conversion and encoding operation:

```
public class GimzipHash {

  public static byte[] hexToBytes(String s) {
    int len = s.length();
    byte[] data = new byte[len / 2];
    for (int i = 0; i < len; i += 2) {
      data[i / 2] = (byte) ((Character.digit(s.charAt(i), 16) << 4)
                           + Character.digit(s.charAt(i+1), 16));
    }
    return data;
  }

  public static void main(String[] args) {
    String hash = args[0];
    System.out.print("Input GIMZIP hash = ");
    System.out.println(hash);
    System.out.print("Base64 encoded hash = ");
    System.out.println(
      javax.xml.bind.DatatypeConverter.printBase64Binary(hexToBytes(hash)));
  }
}
```

Example output from the above sample Java program looks like this:

```
Input GIMZIP hash = FDC4F6BA60960782148DF3916285D94041873BF1
Base64 encoded hash = /cT2umCWB4IUjfORYoXZQEGHO/E=
```

3.4 Server processing

After receiving a getRequest message, the Automated Service Request server is expected to respond

synchronously with an HTTP response code of 200 and a getResponse message that describes the status of the subject order. If the asynchronous processing for the order on the server has not yet completed, then the getResponse must indicate <state>processing</state> and an appropriate <processTime> value. The client will send another getRequest after waiting the number of seconds provided by <processTime>.

If the processing for the order has completed, and a GIMZIP package has been produced containing the requested content, then the getResponse must indicate <state>complete.available</state> and provide the expected information the client will use to download the GIMZIP package files.

If the processing for the order has completed, but no PTFs satisfied the requested selection criteria, then the getResponse must indicate <state>complete.noUpdates</state>.

If an error was detected during processing for the order, then the server must respond with either:

- An HTTP response code of 200 and a getResponse message containing <state>error</state> and appropriate fault information.
- An HTTP response code of 500 and a Fault message containing appropriate fault information.

See Chapter 5 “Faults and Errors” on page 19 for details.

Chapter 4. close

The close method is used to tell the Automated Service Request server that the client has completed its processing for the subject update order.

4.1 URL format

The following shows the format of the URL for this request:

```
https://<host><:port></path>/UpdateOrder?orderid=<transactionId>
```

where:

- "`https://<host><:port>`" specifies the server address and port.
- "`</path>`" identifies the path to the provider.
- "`/UpdateOrder`" indicates the provider for the UpdateOrder method.
- "`<transactionId>`" is the unique ID for the subject order.

The https://<host><:port>/<path> portion of the URL is specified by the user on the "url" attribute in the <ORDERSERVER> tag used by the RECEIVE ORDER command. SMP/E adds the further qualification, "UpdateOrder?orderid=<order-id>" in this case, as appropriate for the request.

4.2 Request content

The request will include a closeRequest message, like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Header>
<header xmlns="http://www.ibm.com/xmlns/b2b/ecc/v1.0">
  <targetURI>uri</targetURI>
  <credentials>
    <certificate>certificate-data</certificate>
  </credentials>
</header>
</soap:Header>

<soap:Body>
  <closeRequest xmlns="http://www.ibm.com/xmlns/b2b/ecc/v1.0/UpdateOrder">
  </closeRequest>
</soap:Body>

</soap:Envelope>
```

where:

Property	Description
<targetURI>	The universal resource identifier (URI) of the service provider on the server. This value is the same as the URL for the HTTP request, https://host:port/path/UpdateOrder?orderid=order-id where "order-id" is the unique identifier for the subject order.
<certificate>	An x.509 certificate in ANS1.DER base64 encoding. The certificate describes the credentials for the user that is making a request to the

Property	Description
	server. The certificate is stored in a security manager data base, and SMP/E extracts the named certificate from the security manager data base and includes it in the request to the server.

4.3 Expected response

After processing the closeRequest message, the Automated Service Request server is expected to respond with an HTTP 200 response code and a closeResponse message, like this:

```
<Envelope>
<Body>

<closeResponse>
<state>order-state</state>
</closeResponse>

</Body>
</Envelope>
```

where:

Property	Description
<state>	The state, or status, for the order on the server. For closeResponse the state may be one of: • closed – processing for the order has been completed, and the order content files will be deleted from the download server. • error – an error was detected by the server. See 5 “Faults and Errors” on page 19 for details.

4.4 Server processing

After receiving a closeRequest message, the Automated Service Request server is expected to respond synchronously with an HTTP response code of 200 and a closeResponse message indicating the subject order has been “closed”. The server can do whatever it deems appropriate to “close” an order. For example, the order content GIMZIP package can be removed from the download server.

If the close processing for the order has completed successfully, then the closeResponse message must indicate <state>closed</state>.

If an error was detected during close processing for the order, then the server must respond with either:

- An HTTP response code of 200 and a closeResponse message containing <state>error</state> and appropriate fault information.
- An HTTP response code of 500 and a Fault message containing appropriate fault information.

See Chapter 5 “Faults and Errors” on page 19 for details.

Chapter 5. Faults and Errors

When the Automated Service Request server detects an error while processing a request, the server response is expected to contain information to describe the error. There are two types of error responses, Faults and Errors, roughly defined as follows:

Faults are often the result of errors encountered in synchronous activities. The server responds with an HTTP response code of 500 and a specific response message form that describes the fault.

Errors are often the result of errors encountered in asynchronous activities. The server responds with an HTTP response code of 200, but with a state of "error" in the response message and information to describe the error.

In reality, SMP/E recognizes and acts upon Faults and Errors equally, regardless of the HTTP response code. That is, when the server responds with an HTTP response code of 500, SMP/E will accept either a Fault or Error response. Likewise, when the server responds with an HTTP response code of 200, SMP/E will accept either a Fault or Error response.

5.1 Fault responses

When a fault is encountered by the Automated Service Request server, the server responds with an HTTP response code of 500 and the following message in place of a submitUpdateOrderResponse, getResponse, or closeResponse message.

```
<Envelope>
<Body>

<Fault>
  <faultcode>fault-code</faultcode>
  <faultstring>fault-string</faultstring>
  <detail>
    <fault-code>
      <description>fault-description</description>

      optional-additional-fault-information

    </fault-code>
  </detail>
</Fault>

</Body>
</Envelope>
```

where:

Property	Description
<faultcode>	A code, in the form of a textual name, to indicate the specific error detected by the server. See 5.3 "Fault codes" on page 21 for a description of the fault codes recognized by SMP/E.
<faultstring>	A longer form of the fault code. This value is ignored by SMP/E.
<description>	A detailed, and human consumable, description of the error detected by the server.

This is an example fault response, returned from the IBM Automated Service Request server:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Header></soapenv:Header>
<soapenv:Body>

<soapenv:Fault>
<faultcode xmlns:p760="http://www.ibm.com/xmlns/b2b/ecc/v1.0">
p760:Server.System
</faultcode>
<faultstring>com.ibm.ecc.protocol.ServerSystem</faultstring>
<detail encodingStyle="">
<p760:Server.System xmlns:p760="http://www.ibm.com/xmlns/b2b/ecc/v1.0">
<p760:description>Sorry, there is some problem with our database server.
</p760:description>
</p760:Server.System>
</detail>
</soapenv:Fault>

</soapenv:Body>
</soapenv:Envelope>
```

5.2 Error responses

When an error is encountered by the Automated Service Request server, the server may respond with an HTTP response code of 200 and a submitUpdateOrderResponse, getResponse, or closeResponse message with a state of “error” and information that describes the error,

```
<Envelope>
<Body>

<response-type>
  <state>error</state>
  <error>
    <faultcode>fault-code</faultcode>
    <faultstring>fault-string</faultstring>
    <description>fault-description</description>
  </error>
</response-type>

</Body>
</Envelope>
```

where:

Property	Description
<response-type>	The response type may be <submitUpdateOrderResponse>, <getResponse>, or <closeResponse>, depending on the request type.
<state>	The state, or status, for the order on the server. • error – an error was detected by the server.

Property	Description
<faultcode>	A code, in the form of a textual name, to indicate the specific error detected by the server. See 5.3 “Fault codes” on page 21 for a description of the fault codes recognized by SMP/E.
<faultstring>	A longer form of the fault code.
<description>	A detailed, and human consumable, description of the error detected by the server.

5.3 Fault codes

The fault codes recognized by SMP/E are as follows.

Code	Description
Client.AccountInformation	The account information, or user, identified by information in the client certificate, is not recognized by the server.
Client.Authentication.NotAuthorized	The client is not authorized to use the services of the server.
Client.Authentication.CredentialsExpired	Client certificate is no longer valid.
Client.Authentication.CredentialsMismatch	Access to the order is denied. The certificate does not match the certificate provided when the order transaction began.
Client.Authentication.NotRecognized	Invalid client certificate.
Client.InsufficientInformation	The server cannot process the request.
Client.InvalidInformation	The server cannot process the request.
Client.InvalidTransactionId	The server does not recognize the transaction identifier specified for the request.
Client.MalformedRequest	The server does not recognize the request.
Client.ObjectNotFound	One or more requested PTFs or APARs were not found. Either specific PTFs, or PTFs for specific APARs, were requested, and one or more of those PTFs or APARs are unknown to the server. See 5.3.1 “Client.ObjectNotFound fault” on page 22 for further details.
Server.DataTooLarge	The request was not satisfied because the order content to satisfy the request would have exceeded the server's defined maximum size for GIMZIP package files. See 5.3.2 “Server.DataTooLarge fault” on page 23 for further details.
Server.Processing	The server had a general processing error.

Code	Description
Server.Processing.DataCollection	An error was detected as the server was collecting information from one of the services it uses.
Server.ServiceUnavailable	One or more of the services used by the server are unavailable.
Server.System	The server, or any of the services it uses, are unavailable.
Server.UnsupportedRequest	The server does not support the request.

Other fault codes not described above may be provided in the Fault or Error response from the server. SMP/E assumes any unknown fault code in the response means the server was unsuccessful processing the request. The fault code, fault string, and description will be echoed in the SMP/E RECEIVE ORDER command output.

5.3.1 Client.ObjectNotFound fault

The Client.ObjectNotFound fault response is used when one or more requested PTFs or APARs are unknown to the server. If one or more specific PTFs were requested, then one or more of those PTFs are unknown to the server. If PTFs for one or more specific APARs were requested, then one or more of those APARs are unknown to the server. For both PTF and APAR requests, the server responds with the following:

```
<Envelope>
<Body>

<Fault>
  <faultcode>Client.ObjectNotFound</faultcode>
  <faultstring>fault-string</faultstring>
  <detail>
    <Client.ObjectNotFound>
      <description>fault-description</description>
      <uri>ptf-name</uri>
      <uri>ptf-name</uri>
    </Client.ObjectNotFound>
  </detail>
</Fault>

</Body>
</Envelope>
```

where:

Property	Description
<uri>	Identifies a single requested object (PTF or APAR) that is unknown by the server. Each unknown PTF or APAR will be indicated in the response by a separate <uri> specification.

This is a simplified example of the Client.ObjectNotFound fault response, returned from the IBM Automated Service Request server:

```
<Envelope>
```

```
<Body>

<Fault>
  <faultcode>Client.ObjectNotFound</faultcode>
  <faultstring>com.ibm.ecc.protocol.ClientObjectNotFound</faultstring>
  <detail>
    <Client.ObjectNotFound>
      <description>Error encountered during processing when requested fix could
not be found.</description>
      <detail>
        <originator>16.2.10 hsb</originator>
        <transactionId>H74121486</transactionId>
      </detail>
      <uri>UZ00001</uri>
      <uri>UZ00002</uri>
    </Client.ObjectNotFound>
  </detail>
</Fault>

</Body>
</Envelope>
```

5.3.2 Server.DataTooLarge fault

The Server.DataTooLarge fault response is used only if the server enforces a maximum package size and detects the resultant package for a request is in fact larger than the implemented maximum size. Servers are not required to enforce a maximum package size limitation. However, if the server does enforce a maximum package size, and if the server detects the resultant package will be larger than can be supported by the server, the server responds with the following:

```
<Envelope>
<Body>

<Fault>
  <faultcode>Server.DataTooLarge</faultcode>
  <faultstring>ServerDataTooLarge</faultstring>
  <detail>
    <Server.DataTooLarge>
      <description>fault-description</description>
      <resultSize>actual-package-size</resultSize>
      <maxSize>maximum-package-size</maxSize>
    </Server.DataTooLarge>
  </detail>
</Fault>

</Body>
</Envelope>
```

where:

Property	Description
<resultSize>	The actual size of the GIMZIP package files
<maxSize>	The maximum GIMZIP package size as defined by the server.

When SMP/E receives a response with the Server.DataTooLarge fault it will stop processing and issue a message like this:

GIM69212S

RECEIVE PROCESSING HAS FAILED FOR ORDER ORD00001. A PACKAGE FOR ORDER ORD00001 WITH ORDERID H00000172 CANNOT BE CREATED BECAUSE THE PACKAGE SIZE (10G BYTES) WOULD EXCEED THE THRESHOLD FOR INTERNET DELIVERY (8G BYTES).

In this message example, the server response contained <resultSize> of “10G” and <maxSize> of “8G”.

End-of-Document