

Site Controller

HTTP Primer

Date: December 2, 2020



Verifone®

Verifone Confidential

Verifone, Inc.

88 West Plumeria Drive

San Jose, CA 95134

Telephone: 408-232-7800

<http://www.verifone.com>

© 2020 Verifone, Inc. All rights reserved.

No part of this publication covered by the copyrights hereon may be reproduced or copied in any form or by any means - graphic, electronic, or mechanical, including photocopying, taping, or information storage and retrieval systems - without written permission of the publisher.

The content of this document is subject to change without notice. The information contained herein does not represent a commitment on the part of Verifone. All features and specifications are subject to change without notice.

Verifone, Ruby SuperSystem, and Secure PumpPAY are registered trademarks of Verifone, Inc. Ruby Card, iOrder, and Commander Site Controller are trademarks of Verifone. All other brand names and trademarks mentioned in this document are the properties of their respective holders.

Revision History

Date	Description
09/06/2017	Converted document from Sapphire to Site Controller to include Commander.
12/02/2020	Removed the entire section on Telnet.

Contents

1. Introduction	1
Glossary of Terms	1
2. Java and JavaScript Examples	3
Java Example	3
JavaScript Example	5
3. Obtaining and Releasing Credentials	9
Java Example	9
JavaScript Example	12
4. Requesting and Saving T-Logs	15
Requesting T-Logs	15
JavaScript Example	16
5. Getting and Posting Documents	19
GET Commands	19
POST Commands	19
JavaScript Example	20
6. Response Messages	25
Responses for GET vAppInfo	25
Responses for releaseCredential	27

1

Introduction

The Site Controller HTTP Primer provides an introduction to the Hyper-Text Transfer Protocol (HTTP) and how you can use it from Java, JavaScript, or VBScript to write programs that interact with Verifone Site Controller systems.

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, object-oriented protocol that can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

HTTP is a very simple protocol to use. Commander supports HTTPS.

Glossary of Terms

The following terms and definitions will assist the reader with understanding the content of the Site Controller URL Reference.

Glossary of Terms		
Term	URL Syntax	Definition
Site Controller	{site-controller_IP}	A generic term for Sapphire and Commander and applies to both.
Commander	{commander_IP}	It is the new generation of the Verifone Site Controller. The Commander specifically operates with the Ruby2 and the Topaz POS workstations.
Sapphire	n/a	Sapphire is the legacy generation of the Verifone Site Controller. Sapphire is an End-of-Life product as of June 25, 2020. References to Sapphire in the URL Reference are due to the use of "sapphire" as namespace definitions, or for identifying deprecated items.

2

Java and JavaScript Examples

The following examples show how to use common programming languages to retrieve XML information from Site Controller.

Java Example

The following example shows how to use a Java program to obtain information about the Site Controller application.

GetAppInfo

Request

```
/*
 * GetAppInfo.java
 *
 * Created on March 14, 2005, 9:07 AM
 */
import java.io.IOException;
import java.io.InputStream;
import
java.io.OutputStream;
import java.net.URL;
import java.net.URLConnection;
/**
 *
 * @author Name_X
 */
public class GetAppInfo
{
```

```

    private static String sapphireAddr = "192.168.31.11"
;
    /** Creates a new instance of GetAppInfo *
/
    public GetAppInfo() {
    }
    public void go()
throws Exception {
        /*
            ** get a data stream
        */
        URL appInfoURL = new URL("http://" + sapphireAddr + "/cgi-bin/CGILink?cmd=vAppInfo");
        URLConnection connection = appInfoURL.openConnection();
        InputStream viewStream = (InputStream)connection.getContent();
        OutputStream out = System.out;
        readOutput(viewStream, out);
    }
    void readOutput(InputStream in, OutputStream out) throws IOException {
        byte[] array = new byte[10240];
        for (;;) {
            int len = in.read(array);
            if (len == -1)
                break;
            out.write(array, 0, len);
        }
        out.flush();
    }
    /**
    * @param args the command line arguments
    */
    public static void main(String[] args) {
        if (args.length > 0) {
            sapphireAddr = args[0];
        }
        try {

```

```

        new GetAppInfo().go();
    }
    catch (Exception e) {
        System.out.println("Error getting App Info");
    }
}
}

```

Response

```

D:\Main>java GetAppInfo 10.64.132.40<?xml version="1.0" encoding="UTF-8"?><domain:appInfo
xmlns:domain="urn:vfi-sapphire:np.domain.2001-07-01"><site>1</
site><newproVersions><version name="VM" majorVersionNr="1" minorVersionNr="00"
releaseVersionNr="02" maintenanceVersionNr="00"
>1.00.02</version><version name="Sapphire" majorVersionNr="4" minor
VersionNr="00" releaseVersionNr="00" maintenanceVersionNr="00" timeStamp="2005-03-
06T21:42:14">4.00.00</version></newproVersions>
<gemproVersions><version dataSubsetName="gcomver" name="Gemcom" isLoadModule="0"
majorVersionNr="1" minorVersionNr="8" releaseVersionNr="8" maintenanceVersionNr="00"
timeStamp="2005-03-04T17:29:35-05:00"/><version dataSubsetName="applver" name="CITPAK
B030105.0823" isLoadModule="0" majorVersionNr="5" minorVersionNr="1
" releaseVersionNr="0" maintenanceVersionNr="00"/><version dataSubsetName="osver"
name="Gemix" isLoadModule="0" majorVersionNr="2" minorVersionNr="6" releaseVersionNr="18"
maintenanceVersionNr="00" timeStamp="2000-11-10T18:00:00-05:00"/><version
dataSubsetName="gcldm1"
name="gcomms.ldm" isLoadModule="1" size="29150" timeStamp="2005-03-01T14:53:43-05:00"/
><version dataSubsetName="gcldm2" name="gemcom.ldm" isLoadModule="1" size="269042"
timeStamp="2005-03-04T17:29:35-05:00"/></gemproVersions><topazVersions><version
register="102" name="Topaz" majorVersionNr="1" minorVersionNr="03" releaseVersionNr="00"
timeStamp="2005-03-12T16:28:04"/></topazVersions></domain:appInfo>

```

JavaScript Example

The following JavaScript example shows how to use JavaScript and the Microsoft™ MSXML parser to get the same XML document as in the previous example.

```

<?xml version="1.0" standalone="yes"?>
<!--
example script
-->
<package>
<job id="HelloWorld">
<?job debug="false" error="true" ?>
<runtime/>
<object id="shell" progid="WScript.Shell"/>
<script language="JScript"><![CDATA[ // <--make it opaque to XML

```

```
//-----  
function main()  
{  
  //variables  
  var cmd = "vAppInfo";  
  var CGILink = "/cgi-bin/CGILink?";  
  var base = "http://192.168.31.11";  
  var url = "";  
  
  //create a DOM to load the xml into  
  var domDoc = new ActiveXObject("MSXML2.DOMDocument.4.0");  
  
  //set the properties of the DOM  
  domDoc.async=false;  
  domDoc.validateOnParse = false;  
  domDoc.resolveExternals=false;  
  
  //build the url to load from  
  url = base + CGILink + "cmd=" + cmd;  
  //request the document using the load method of the DOM  
  domDoc.load(url);  
  //show the results  
  WScript.echo(domDoc.xml);  
}  
  
try {  
  main();  
  WScript.quit(0);  
}catch (e){  
  var str;  
  for ( var prop in e){  
    str += prop + ": " + e[prop] + "\n";  
    if(prop == 'dom'){  
      str += e[prop].xml;  
    }  
  }  
}
```

```
}  
WScript.echo("unexpected exception \n" + str);
```

```
}
```

```
]]></script>  
</job>  
</package>
```

Note that the returned document is the same as in the previous two examples.

3

Obtaining and Releasing Credentials

The initial request in a Site Controller document exchange session is always for a credential document. After you have completed all work in a session, you must release the credential.

Java Example

The following example shows how to use a Java program to request a credential, print its value, and then release it.

GetCredential

Request

```
/*
 * GetCredential.java
 *
 * Created on March 15, 2005, 8:15 AM
 */
import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
```

```

/** This class can be used to test all CGILink (view) commands
 * against a specified sapphire
 *
 * @author "mailto:name_x@xxxxxx.com"
 * @version 1.0 Copyright(c) 2001 Verifone Inc., All Rights Reserved
 */
public class GetCredential {
    private static String sapphireAddr = "192.168.31.11";

    /** Constructor for the class
     * @throws Exception on error in processing the request
     */
    public GetCredential() throws Exception {
    }

    /*******
     * This method performs the requested operation
     *****/

    void go() throws Exception {
        /**
         ** get a data stream
         */

        URL validateURL = new URL("http://" + sapphireAddr + "/cgi-bin/
        CGILink?cmd=validate&user=helpdesk&passwd=123");
        URLConnection validateConnection = validateURL.openConnection();
        /**
         ** make a parser and set it up
         */

        DocumentBuilder builder =
        DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document doc = builder.parse((InputStream)validateConnection.getContent());

        NodeList nodes = doc.getElementsByTagName("cookie");
        String cookie = null;
        if (nodes.getLength() > 0) {
            Node curNode = nodes.item(0);

```



```

        if (curNode != null) {
            Node val = curNode.getFirstChild();
            if (val != null) {
                cookie = val.getNodeValue();
            }
        }
    }
    /*
    ** examine results
other document requests in a normal session would happen here
    */
    System.out.println("cookie is "+cookie);
    /*
    ** release credential
    */
    URL releaseURL = new URL("http://"+sapphireAddr+"/cgi-bin/
CGILink?cmd=releaseCredential&cookie="+cookie);
    URLConnection releaseConnection = releaseURL.openConnection();
    // reqd to actually release the cookie
    releaseConnection.getContentType();
}

/** This method is the entry point for the program
 * Usage: java GetCredential ipaddr cmd [outFile name passwd param]
 * @param args the parameters required to perform the view function
 * @throws Exception on insufficient parameters and/or error in performing the request
 */
static public void main(String args[]) throws Exception {
    if (args.length > 0) {
        sapphireAddr = args[0];
    }
    try {
        new GetCredential().go();
    }
    catch (Exception e) {
        System.out.println("Error getting credential");
    }
}

```

```

    }
  }
}

```

Response

```

D:\Main>java GetCredential 10.64.132.40
cookie is 82b3e1

```

JavaScript Example

The following example shows how to use JavaScript to request a credential, print its value, and then release it.

Request

```

<?xml version="1.0" standalone="yes"?>
<!--
example script
-->
<package>
<job id="SampleCredential">
<?job debug="false" error="true" ?>
<runtime/>
<object id="shell" progid="WScript.Shell"/>
<script language="JScript"><![CDATA[ // <--make it opaque to XML

```

```

//-----

```

```

function main()
{
var cmd = "validate";
var user = "manager";
var passwd = "123";
var base = "http://192.168.31.11";
var CGILink = "/cgi-bin/CGILink?";
var url = "";
var cookie ;

```

```
//create a DOM to load the xml into
var domDoc = new ActiveXObject("MSXML2.DOMDocument.4.0");

//set the properties of the DOM
domDoc.async=false;
domDoc.validateOnParse = false;
domDoc.resolveExternals=false;

//build the url to load from
url = base + CGILink + "cmd=" + cmd + "&user=" + user + "&passwd=" + passwd;
//request the document using the load method of the DOM
domDoc.load(url);
//get the value of the cookie element
cookie = domDoc.selectSingleNode("//cookie").text;
//show the cookie
WScript.echo("Cookie is: " + cookie);

//Release the credential
//set cmd to relaseCredential
cmd = "releaseCredential";
//build the url
url = base + CGILink + "cmd=" + cmd + "&cookie=" + cookie;
domDoc.load(url);
//show the response
WScript.echo("Response is: \n" + domDoc.xml);
}

try {
main();
WScript.quit(0);
}catch (e){
var str;
for ( var prop in e){
str += prop + ": " + e[prop] + "\n";
if(prop == 'dom'){
str += e[prop].xml;
}
```

```
}  
}  
WScript.echo("unexpected exception \n" + str);  
  
}  
]]></script>  
</job>  
</package>
```

4

Requesting and Saving T-Logs

Requesting T-Logs

A T-Log (transactionSet/transSet) is an electronic journal of all transactions and journal events from all POS stations at a site during a given period. Transaction sets are available only for period 1 and period 2, where a period 2 transaction set contains all the period 1 transactions since the last period 2 close. Prior to requesting the transaction set, you can request a list of available transaction sets by issuing the **vtlogpdl** command:

```
https://{site-controller_IP}/cgi-bin/CGILink?  
cmd=vtlogpdl&cookie={cookie}.
```

Note: *You do not need to request a list of transaction sets if you already know the report identifier. You can use the **filename** argument to request a specific transaction set.*

The document returned contains a list of all transaction sets available and a set of parameters that you need to use to request a given transaction set. All closed transaction sets contain a name in a yyyy-mm-dd.nnn format, where *nnn* is the sequence number of the report.

The following example uses the information from the credential and **vtlogpdl** XML documents to request a closed period 1 transaction set:

```
https://{sapphire_IP}/cgi-bin/CGILink?  
cmd=vperiodrept&period=1&filename=2003-12-05.001&cookie=a1d7e4
```

```
https://{commander_IP}/cgi-bin/CGILink?  
cmd=vtransset&period=1&filename=2003-12-05.001&cookie=a1d7e4
```

The **cmd** and **cookie** parameters come from the credential, while the **period** and **filename** come from the **vreportpdl**. The above request returns the **transactionSet/transSet** as an XML document. Since the transaction set can be rather large, you can also retrieve it in a *gzip* format by changing the cmd from **vperiodrept** to **vperiodreptz**, or **vtransset** to **vtranssetz**.

The raw transaction set can be difficult to work with. You can convert the transaction set to an easier to use T-Log format using the tlogDoc.xslt XML style sheet. The path to the tlogDoc.xslt style sheet is:

`https://{sapphire_IP}/templates/vfit/transactionSet/tlogDoc.xslt`

`https://{commander_IP}/templates/vfit/transSet/tlogDoc.xslt`

Note: See the Site Controller Style Sheets Reference for information.

The transformation process must be initiated on the client; that is, when a **transactionSet/transSet** is requested, the Site Controller does not transform it to the T-Log format. Instead, the client side must transform the **transactionSet/transSet** document.

JavaScript Example

The following example shows how to download a transaction set from Site Controller and save it to a disk. Note that get and release credential are included.

```
<?xml version="1.0" standalone="yes"?>
<!--
example script
-->
<package>
<job id="getTlog">
<?job debug="false" error="true" ?>
<runtime/>
<object id="shell" progid="WScript.Shell"/>
<script language="JScript"><![CDATA[ // <--make it opaque to XML

//-----

function main()
{
var cmd = "validate";
var user = "manager";
var passwd = "123";
var filename = "current";
```

```
var period = "2";
var base = "http://192.168.31.11";
var CGILink = "/cgi-bin/CGILink?";
var url = "";
var cookie = "";

//create a DOM to load the xml into
var domDoc = new ActiveXObject("MSXML2.DOMDocument.4.0");

//set the properties of the DOM
domDoc.async=false;
domDoc.validateOnParse = false;
domDoc.resolveExternals=false;
//GET A CREDENTIAL
//build the url to load from
url = base + CGILink + "cmd=" + cmd + "&user=" + user + "&passwd=" + passwd;
//request the document using the load method of the DOM
domDoc.load(url);
//get the value of the cookie element
cookie = domDoc.selectSingleNode("//cookie").text;

//GET THE TLOG
//Set cmd to vperiodrept (cmd to get tlogs)
cmd = "vperiodreptz";
//build url
url = base + CGILink + "cmd=" + cmd + "&filename=" + filename + "&period=" + period +
"&cookie=" + cookie;
//get the tlog
domDoc.load(url);
//write the tlog to disk
domDoc.save(filename + ".xml")

//RELEASE THE CREDENTIAL
//set cmd to relaseCredential
cmd = "releaseCredential";
```

```
//build the url
url = base + CGILink + "cmd=" + cmd + "&cookie=" + cookie;
domDoc.load(url);
//show the response
WScript.echo("Response is: \n" + domDoc.xml);

}

try {
main();
WScript.echo("Done");
WScript.quit(0);
}catch (e){
var str;
for ( var prop in e){
str += prop + ": " + e[prop] + "\n";
if(prop == 'dom'){
str += e[prop].xml;
}
}
WScript.echo("unexpected exception \n" + str);
}

]]></script>
</job>
</package>
```


5

Getting and Posting Documents

You retrieve and send documents to Site Controller by using HTTP GET and POST requests to a Common Gateway Interface (CGI) application.

GET Commands

GET commands are used to view data from a Site Controller and to obtain a credential document.

Most browser commands are simple GET requests. All GET requests can be issued from the browser address directly. The browser turns a URL into a sequence of lines that make up an HTTP GET command. The URL should be valid and separate from the sequence of lines.

The syntax for a GET request is shown in the following example:

```
https://{site-controller_IP}/cgi-bin/CGILink?  
cmd=vpayrollpdlist&cookie={cookie}
```

Note: Site Controller GET command names begin with a 'v' (for 'view').

POST Commands

POST commands are used to update datasets and to retrieve and update PLU-related data. When you use a POST command, you must supply a payload; that is, the XML document that contains the data you want to update on the Site Controller.

Note: Site Controller POST command names begin with a 'u' (for 'update').

JavaScript Example

```
<?xml version="1.0" standalone="yes"?>
<!--
example script
-->
<package>
<job id="getAndPostData">
<?job debug="false" error="true" ?>
<runtime/>
<object id="shell" progid="WScript.Shell"/>
<script language="JScript"><![CDATA[ // <--make it opaque to XML

//-----
function main()
{

var cmd = "validate";
var user = "manager";
var passwd = "123";
var base = "http://192.168.31.11";
var CGILink = "/cgi-bin/CGILink?";
var CGIUplink = "/cgi-bin/CGIUplink";
var url = "";
var cookie = "";
var payload = "";

//create a DOM to load the xml into
var domDoc = new ActiveXObject("MSXML2.DOMDocument.4.0");

//set the properties of the DOM
domDoc.async=false;
domDoc.validateOnParse = false;
domDoc.resolveExternals=false;
```

```
//GET THE CREDENTIAL
//build the url to load from
url = base + CGILink + "cmd=" + cmd + "&user=" + user + "&passwd=" + passwd;
//request the document using the load method of the DOM
domDoc.load(url);
//get the value of the cookie element
cookie = domDoc.selectSingleNode("//cookie").text;

//GET THE BANNER CONFIG DOC
//Set the cmd to vbannercfg
cmd="vbannercfg";
//Build the url
url = base + CGILink + "cmd=" + cmd + "&cookie=" + cookie;
domDoc.load(url);
//show the xml
WScript.echo(domDoc.xml);

//Send the document back to Sapphire unmodified
//POST THE BANNER CONFIG DOC
//Create a HTTP object to post data [payload] to Sapphire
var oHttp = new ActiveXObject("MSXML2.XMLHTTP.4.0");

//Set the cmd to ubannercfg
cmd="ubannercfg";

//build the url
url = base + CGIUplink

//open connection
oHttp.open("POST", url, false);

//build payload. Payload contains the cmd and cookie as well as the data. The data is seperated
from the name/value pairs through CRLFs
payload = "cmd=" + cmd + "&cookie=" + cookie + "\r\n\r\n" + domDoc.xml + "\r\n\r\n";

//send the data
```

```

oHttp.send(payload);

//Display the contents of responseXML if the status.text is OK
if (oHttp.statusText != "OK") {
WScript.echo("Server Error while posting data: \n" + oHttp.statusText);
}else{
oHttp.responseXML
WScript.echo("Post of bannercfg response is: \n" + oHttp.responseXML.xml);
}

//RELEASE THE CREDENTIAL
//set cmd to relaseCredential
cmd = "releaseCredential";
//build the url
url = base + CGILink + "cmd=" + cmd + "&cookie=" + cookie;
domDoc.load(url);
//show the response
WScript.echo("Release Credential response is: \n" + domDoc.xml);

}

try {
main();
WScript.quit(0);
}catch (e){
var str;
for ( var prop in e){
str += prop + ": " + e[prop] + "\n";
if(prop == 'dom'){
str += e[prop].xml;
}
}
WScript.echo("unexpected exception \n" + str);
}

]]></script>

```

</job>

</package>

6

Response Messages

Responses for GET vAppInfo

GOOD GET RESPONSE vAppInfo

The following good response returns the vAppInfo document in response to the URL command **<https://10.64.128.181/cgi-bin/CGILink?cmd=vAppInfo>**.

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<domain:applInfo xmlns:domain="urn:vfi-sapphire:np.domain.2001-07-01">
  <site>AB123</site>
  <newproVersions>
    <version name="VM" majorVersionNr="1" minorVersionNr="00" releaseVersionNr="02"
      maintenanceVersionNr="00">1.00.02</version>
    <version name="Sapphire" majorVersionNr="3" minorVersionNr="02" releaseVersionNr="11"
      maintenanceVersionNr="00" timeStamp="2005-01-28T08:26:07">3.02.11</version>
  </newproVersions>
  <gemproVersions>
    <version dataSubsetName="gcomver" name="Gemcom" isLoadModule="0" majorVersionNr="1"
      minorVersionNr="8" releaseVersionNr="3" maintenanceVersionNr="01" timeStamp="2004-11-
      03T21:14:48-05:00"/>
    <version dataSubsetName="applver" name="BUYPAK B112904.1432" isLoadModule="0"
      majorVersionNr="4" minorVersionNr="1" releaseVersionNr="9" maintenanceVersionNr="00"/>
    <version dataSubsetName="osver" name="Gemix" isLoadModule="0" majorVersionNr="2"
      minorVersionNr="6" releaseVersionNr="18" maintenanceVersionNr="00" timeStamp="2000-11-
      10T18:00:00-05:00"/>
    <version dataSubsetName="gcldm1" name="gcomms.ldm" isLoadModule="1" size="28766"
      timeStamp="2004-11-03T21:11:32-05:00"/>
    <version dataSubsetName="gcldm2" name="gemcom.ldm" isLoadModule="1" size="258074"
      timeStamp="2004-11-03T21:14:48-05:00"/>
  </gemproVersions>
  <topazVersions/>
</domain:applInfo>
```

Bad GET Response vappinfo

The following fault document is returned in response to the URL command **https://10.64.129.107/cgi-bin/CGILink?cmd=vappinfo** because **cmd** is case sensitive.

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<VFI:Response xmlns:VFI="urn:vfi-sapphire:np.domain.2001-07-01">
  <VFI:Fault>
    <faultCode>CGIPortal.InternalError</faultCode>
    <faultString>CGIPortal Error</faultString>
    <detail>
      <e:vfiFault xmlns:e="urn:vfi-sapphire:np.domain.2001-07-01">
        <e:message>no value called cookie found</e:message>
      </e:vfiFault>
    </detail>
  </VFI:Fault>
</VFI:Response>
```


Responses for releaseCredential

Good GET Response for releaseCredential

The following good response release a credential in response to the URL command **https://10.64.128.181/cgi-bin/CGILink?cmd=releaseCredential&cookie=a1d7e4.**

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<VFI:Response xmlns:VFI="urn:vfi-sapphire:np.domain.2001-07-01"/>
```

Bad GET Response for releaseCredential

The following fault document is returned in response to the URL command **http://10.64.128.181/cgi-bin/CGILink?cmd=releaseCredential&cookie=abc123** because the cookie was invalid.

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<VFI:Response xmlns:VFI="urn:vfi-sapphire:np.domain.2001-07-01">
  <VFI:Fault>
    <faultCode>CGIPortal.InternalError</faultCode>
    <faultString>CGIPortal Error</faultString>
    <detail>
      <e:vfiFault xmlns:e="urn:vfi-sapphire:np.domain.2001-07-01">
        <e:message>no credential for cookie abc123</e:message>
      </e:vfiFault>
    </detail>
  </VFI:Fault>
</VFI:Response>
```

Generalization

All GET and POST commands follow this pattern:

1. GET
 - a. Good response—The document you requested.
 - b. Bad response—A <VFI:Response/> with a fault explaining why.
2. POST
 - a. Good response—A <VFI:Response/> with an indication of success.
 - b. Bad response—A <VFI:Response/> with a fault explaining why.

