HP OpenView Operations

HTTPS-based Java GUI Support on OVO Management Server

Software Version: A.08.14
Edition 2

UNIX

Manufacturing Part Number: none (PDF only)

October 2005

Legal Notices

Contents

Conventions

The following typographical conventions are used in this manual.

Table 1 **Typographical Conventions**

Font	Meaning	Example
Italic	Book or manual titles, and man page names	Refer to the *OVO Administrator's Reference* and the *opc(1M)* manpage for more information.
	Emphasis	You *must* follow these steps.
	Variable that you must supply when entering a command	At the prompt, enter **`rlogin`** ***`username`***.
	Parameters to a function	The *`oper_name`* parameter returns an integer response.
Bold	New terms	The **HTTPS agent** observes...
`Computer`	Text and other items on the computer screen	The following system message displays: `Are you sure you want to remove current group?`
	Command names	Use the `grep` command ...
	Function names	Use the `opc_connect()` function to connect ...
	File and directory names	`/opt/OV/bin/OpC/`
	Process names	Check to see if `opcmona` is running.
	Window/dialog-box names	In the `Add Logfile` window ...
	Menu name followed by a colon (`:`) means that you select the menu, then the item. When the item is followed by an arrow (`->`), a cascading menu follows.	Select `Actions: Filtering -> All Active Messages` from the menu bar.

Table 1 Typographical Conventions (Continued)

Font	Meaning	Example
`Computer Bold`	Text that you enter	At the prompt, enter **`ls -l`**
Keycap	Keyboard keys	Press **Return**.
`[Button]`	Buttons in the user interface	Click `[OK]`.

1 About the HTTPS-based Java GUI

About this Document

This document describes the HTTPS-based Java GUI as a solution for providing a secure communication between Java GUI and the OVO management server.

This chapter includes the following sections:

❑ **The HTTPS-based Java GUI Architecture**

Outlines the HTTPS-based Java GUI underlying concepts and architecture.

❑ **Establishing a Secure Communication**

Describes the process for establishing a secure communication.

❑ **Installing the HTTPS-based Java GUI**

Explains how to install and enable the HTTPS-based Java GUI.

❑ **Disabling Non-secure Communication**

Explains how to disable a non-secure communication between the Java GUI client and the OVO management server.

❑ **Configuring the HTTPS-based Communication on the Java GUI Client**

Details the parameters in the `ito_op` (`ito_op.bat`) startup script related to HTTPS-based Java GUI.

❑ **Configuring opcuihttps Process**

Explains how to configure `opcuihttps` settings and list the parameters related to HTTPS-based Java GUI.

❑ **Installing Core Functionality**

Outlines the role that Core functionality has in the process of establishing the secure communication and explains how to install it.

❑ **About Certificates**

Explains how to provide certificates for both, server and full, modes of authentication.

- ❏ **Configuring the HTTPS-based Java GUI Connection through Firewalls**

  Explains how to configure the connection between the OVO management server and the Java GUI client through the proxy server.

- ❏ **Known Problems and Workarounds**

  Describes the problems specific to running HTTPS-based Java GUI on the OVO management server.

The HTTPS-based Java GUI Architecture

The standard Java GUI supplied with OVO 8 has no secured link to the management server. This functionality is provided with the HTTPS-based Java GUI, that is the Java GUI which uses a HTTPS protocol with Secure Socket Layer (SSL) encryption for communication with OVO management server. The SSL encryption is based on the Core functionality components.

The HTTPS protocol acts as a guardian for applications, gauging which incoming communication requests are trustworthy for secure exchange of data. For details on how the secure communication is established, see “Establishing a Secure Communication” on page 13.

The HTTPS functionality provides three prerequisites for network security:

- ❏ Secrecy
- ❏ Data integrity
- ❏ Authentication

Once a user logs in to the HTTPS-based Java GUI, the communication using the HTTPS protocol is initiated between the client and the management server, based on the authentication of certificates. For more information about the certificates, see “About Certificates” on page 26.

Implementation of HTTPS-based communication also secures Service Navigator requests. The HTTPS protocol establishes a secure link between the OVO Java-based operator client and the OVO management server.

Establishing a Secure Communication

The process of establishing a secure communication is as follows:

Java GUI client connects to the `opcuihttps` process, which acts as a proxy between Java GUI client and OVO management server using the HTTPS protocol. For more information about configuring the `opcuihttps` process, see "Configuring opcuihttps Process" on page 23.

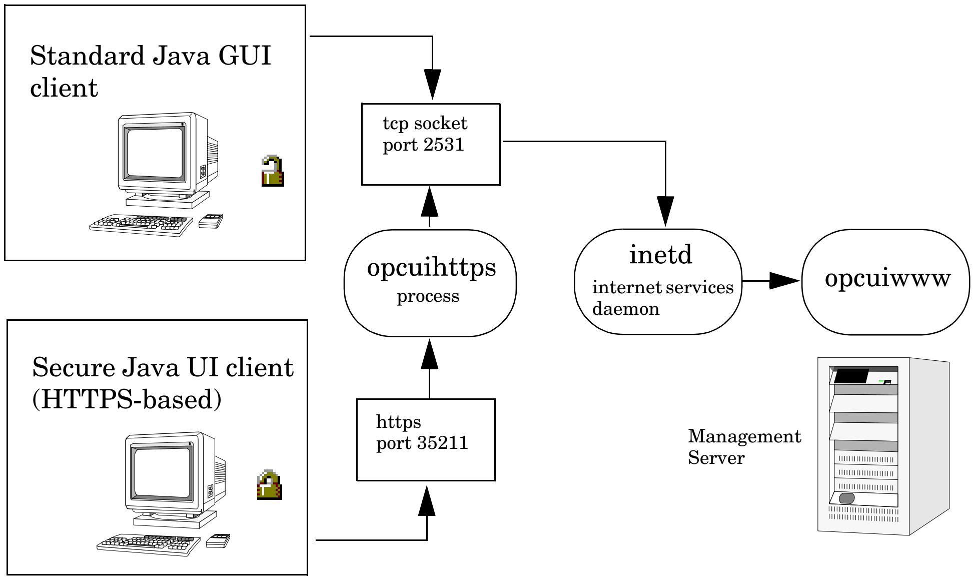
Java GUI communicates with `opcuihttps` process using a secure HTTPS protocol on the port 35211. The `opcuihttps` then redirects the https requests to the standard Java GUI port (2531) using socket communication. All forwarded https requests are then handled by `inetd` process, as well as the requests from non-secure Java GUI clients.

The `opcuihttps` also processes replies from the OVO management server and mediates them to the Java GUI using the HTTPS protocol.

This way all communication requests, from Java GUI to OVO management server and the other way round, become trustworthy for secure exchange of data.

Figure 1-1 on page 14 shows the client-server communication.

Figure 1-1 Client-server Communication



About the Authentication Process

The authentication process which ensure establishing a secure communication has four steps:

1. **Operator logs in.**

   The operator enters a username and a password at login.

   For the login to work, a certificate does not need be installed on the HTTPS-based Java GUI client. For details, see “About Certificates” on page 26.

2. **A certificate is generated.**

   If the Java GUI contacts the management server for the first time, a server certificate is generated.

   The management server then sends the certificate to authenticate itself to the Java GUI client.

NOTE If you choose to use the server certificate for more than a current session, it gets stored in the local Certificate Store. This certificate will then be used for each subsequent connection between Java GUI and the management server.

3. **Client identifies the server.**

   Based on the certificate it has received from the management server, the client identifies the management server.

   If you are using a full authentication mode, the client also authenticates itself with the client certificate. This way the higher level of security is achieved. See "About Certificates" on page 26 for more information about the authentication modes.

4. **Communication channel is opened.**

   If the authentication is successful, a communication channel is opened.

NOTE In case that HTTPS-based communication between Java GUI and the OVO management server fails to establish, you are prompted to use a non-secure communication type. If you press `Cancel`, the login window is displayed.

If you set the `https_only` parameter in the `ito_op` startup script to **`yes`**, you are not prompted to use a non-secure communication. See "Configuring the HTTPS-based Communication on the Java GUI Client" on page 21 for more information about the startup parameters for the HTTPS-based communication.

Figure 1-1 on page 14 shows what you see depending on the chosen communication type:

❏ **HTTPS-based communication**

  If you are using the HTTPS-based Java GUI communication, a *closed* padlock icon appears on the login window and on the status bar.

❏ **Standard communication**

  If you are using the standard HTTPS Java GUI communication, an *open* padlock icon appears in the GUI.

Installing the HTTPS-based Java GUI

This section contains the following information:

- ❏ To Install and Enable the HTTPS-based Java GUI
- ❏ Required OVO Patches

To Install and Enable the HTTPS-based Java GUI

IMPORTANT The following installation procedure is applicable *only* for the OVO Java GUI A.08.14.

To install and enable the HTTPS Java GUI communication type, follow these steps:

1. Start the `opcuihttps` process on the OVO management server. Perform the following:

   a. Move the `opcuihttps` file from `/opt/OV/contrib/OpC/opcuihttps` to `/opt/OV/bin/OpC`.

   b. Start the `opcuihttps` process. Enter the following:
   `/opt/OV/bin/OpC/opcsv -start`

2. Enable HTTPS communication on the Java GUI client. Do one of the following:

   a. Start Java GUI client from the command line using the option `-https true`. For example, enter the following:

      — *On Windows systems*
      ```
      C:\Program Files\Hewlett-Packard\HP OVO Java
      Console>ito_op -https true
      ```

      — *On HP-UX and SOLARIS systems*
      ```
      /opt/OV/www/htdocs/ito_op/ito_op https=true
      ```

b. Edit the `ito_op` startup script. Perform the following:

— *On Windows systems*
In the `ito_op.bat` script, replace the line:
```
if "%HTTPS%" == "" set HTTPS=false
```
with the following line:
```
if "%HTTPS%" == "" set HTTPS=true
```

— *On HP-UX and SOLARIS systems*
In the `ito_op` script, replace the line:
```
https=false
```
with the following line:
```
https=true
```

c. Edit the `ito_for_activator.html` file to start Java UI as an applet.

— To start Java UI in Internet Explorer replace following line:
```
<PARAM NAME = https VALUE = "false">
```
with the following line:
```
<PARAM NAME = https VALUE = "true">
```

— To start Java GUI in Mozzila or Firefox web browser, locate and change the `https="false"` to `https="true"`in the line starting with:
```
else if (_ns == true) document.writeln...
```

NOTE A required Java runtime environment (JRE) version for running Java UI in the HTTPS communication mode is 1.4.2_09.
To set up the JRE on UNIX systems, export the `JAVA_DIR` variable to the base directory where the JRE is installed. For example, enter the following:

```
export JAVA_DIR=/opt/java1.4/jre/
```

Required OVO Patches

The following are OVO patches required for the HTTPS-based Java GUI:

OVO Server A.08.1x - A.08.11

OVO Java GUI A.08.1x - A.08.11

Core Function (L-Core) A.02.1x - A.08.11

Event/Action A.08.1x - A.08.11

OVO Server A.08.1x - A.08.14

OVO Java GUI A.08.1x - A.08.14

Disabling Non-secure Communication

To ensure the secure exchange of data between Java GUI and the OVO management server, it is recommended to disable the non-secure communication. This is achieved by disabling all non-localhost connections to the port 2531. To do so, perform the following:

- ❑ *On HP-UX systems*

  Edit the `/var/adm/inetd.sec` file. Enter the following line:

  ```
  ito-e-gui allow 127.0.0.1
  ```

- ❑ *On Solaris systems*

  Enable the `tcp_wrappers` package.

NOTE The `tcp_wrappers` package is installed on Solaris 9 by default, while on Solaris 8 it has to be installed additionally.

In the `/etc/default/inetd` file, specify the following parameter:

```
ENABLE_TCPWRAPPERS=YES
```

To control the connection, edit also the `/etc/hosts.allow` and `/etc/hosts.deny` files as follows:

- ❑ In the `/etc/hosts.allow`, enter the following:

  ```
  opcuiwww.sh : localhost,127.0.0.1
  ```

- ❑ In the `/etc/hosts.deny`, enter the following:

  ```
  opcuiwww.sh : ALL
  ```

Configuring the HTTPS-based Communication on the Java GUI Client

To configure the HTTPS-based communication between Java GUI and the management server, you can set some parameters in the `ito_op` (`ito_op.bat` on Windows) Java GUI startup script.

Table 1-1 presents the available startup options for setting the HTTPS-based communication type.

NOTE

Upon starting `ito_op` (`ito_op.bat` on Windows) from the command line, you can specify *only* the `https` parameter. You can set all other startup parameters by adding them in the `ito_op` (`ito_op.bat` on Windows) script at the end of line where the Java GUI startup command is stated, or in the `itooprc` configuration file, which is located in the user's home directory.

Table 1-1 Startup Script Options for Setting the HTTPS Protocol

Option	Format	Default	Description
`https`	`yes|no`	`yes`[a]	Sets a HTTPS-based communication type to be used at the startup. If set to **`no`**, the standard non-secure communication is used.
`lcore_defaults`	`yes|no`	`no`	This option is necessary for setting the full SSL authentication. If set to **`yes`**, the default Core functionality directories are used.
`https_only`	`yes|no`	`no`	If set to **`yes`**, disables the communication fallback to the standard socket communication.
`https_port`	*<number>*	35211[b]	A port on which `opcuihttps` is listening.

a. In the OVO Java GUI A.08.14 `ito_op` script (`ito_op.bat` on Windows), the value of the `https` parameter (including the value set in the `itooprc` configuration file) is overridden and automatically set to `no` upon the Java GUI startup. This means that, by default, Java GUI is started with standard non-secure communication in the OVO Java GUI version A.08.14.

b. The port on which `opcuihttps` is listening, used to establish a secure HTTPS-based connection. The standard Java GUI uses the port 2531.

For information about configuring the connection between the OVO management server and the Java GUI client through the proxy server, see “Configuring the HTTPS-based Java GUI Connection through Firewalls” on page 29.

Configuring opcuihttps Process

The `opcuihttps` process acts as a proxy between Java GUI client and OVO management server. It is controlled by the Control Manager process `opcctlm`, which means that `opcuihttps` is started and stopped together with other server processes.

Configuring the Parameters for opcuihttps

Configuration parameters for `opcuihttps` are read at its startup.

NOTE In case any of the `opcuihttps` parameters are changed during the runtime, it is required to restart the `opcuihttps` process.

To change the `opcuihttps` parameters, enter the following command:

```
ovconfchg -ovrg server -ns opc.opcuihttps -set \
<parameter> <value>
```

Table 1-2 lists the parameters for configuring the `opcuihttps` process.

Table 1-2 The opcuihttps Parameters

Parameter	Format	Default value	Description
`SERVER_PORT`[a]	*<number>*	35211[b]	A port on which the Java GUI is listening.
`OPCUIWWW_PORT`	*<number>*	2531	The `opcuiwww` port number as defined in `/etc/services`, `ito-e-gui` entry.
`SSL_CLIENT_ VERIFICATION_MODE`	`Anonymous` | `RequireCertificate`	Anonymous	Specifies whether the `opcuihttps` server accepts anonymous connections from the clients. If set to `RequireCertificate`, the clients will require the certificate for (full) authentication[c].
`MAX_CONNECTIONS`	*<number>*	100	The maximum number of connections to `opcuihttps`.

a. For troubleshooting purposes, you can also set the port in the command line, by starting `opcuihttps` the with the *<server_port>* parameter specified.
b. The port on which `opcuihttps` is listening, used to establish a secure HTTPS-based connection. The standard Java GUI uses the port 2531.
c. For full authentication, set also the startup parameter `lcore_defaults` to **`yes`**.

NOTE You can check if it is possible to connect to the `opcuihttps` process using a web browser, such as Internet Explorer or Mozilla. To do so, enter the following:

`https://`*`<server>`*`:`*`<port>`*`/opcuihttps/info`

Where the `<server>` is an OVO management server hostname, and the `<port>` is the port on which `opcuihttps` is listening.

Installing Core Functionality

The Core functionality is required on the client system for the following reasons:

- ❏ If you have chosen to set the full authentication mode, but there is no HTTPS OVO agent installed on the Java GUI client.
- ❏ If you want to configure the connection between the OVO management server and the Java GUI client through the proxy server.

To Install the Core Functionality

1. Copy the following packages:

   `HPOvXpl`

   `HPOvBbc`

   `HPOvSecCo`

   `HPOvSecCC`

   from the OVO management server vendor tree, at the following location:

   ```
   /var/opt/OV/share/databases/OpC/mgd_node/vendor/\
   <VENDOR>/<FAMILY>/<OS>/<OVO_VERSION>/RPC_BBC
   ```

   to the Java GUI client system.

   For example, for Windows XP, these packages are located in the following directory:

   ```
   /var/opt/OV/share/databases/OpC/mgd_node/vendor/ms/x86/\
   winnt/A.08.10.160/RPC_BBC/
   ```

2. Install the packages. On Windows XP, for example, these packages are as follows:

   `HPOvXpl.msi`

   `HPOvBbc.msi`

   `HPOvSecCo.msi`

   `HPOvSecCC.msi`

About Certificates

The HTTPS-based Java GUI provides network security through the exchange and authentication of electronic **certificates** between client and server. A certificate is a way of endorsing a public key, and includes, in an encrypted format, the username and public key of the sender.

Certificates are signed with the private key of the trusted **Certification Authority** (CA) who issued it. The CA then appends its public key to the certificate, which means that it can be verified by the person receiving it.

About Authentication Modes

The SSL encryption is provided for the following authentication modes:

❏ **Server authentication**

  This is a default authentication mode, where only server certificates are required. It is possible to connect to an OVO management server from an anonymous Java GUI client.

❏ **Full authentication**

  The full authentication mode requires that client certificates are installed on the client systems.

Providing Certificates for Server Authentication Mode

To use HTTPS-based Java GUI, it is not necessary to have any certificate installed, since it is possible to connect to the OVO management server anonymously.

The certificates are generated upon first connection to the OVO management server, and stored, together with the public key, in the local Certificate Store. The certificates are then used by the `opcuihttps` server in the process of authentication.

Depending on whether the certificate exists in the local Certificate Store, the following scenarios are possible:

❏ *Certificate is already stored on the client.*

The communication between Java GUI and the OVO management server is established without any notice.

❏ *Certificate does not exist on the client.*

The `OVO Server Certificate` dialog window is displayed. This dialog window prompts you whether you want to accept the HP OVO server certificate. The following selections are available:

- If you choose **`Yes`**, the HP OVO server certificate is accepted only for the current session. You will be prompted again to accept this certificate upon next login.
- If you choose **`No`**, the connection to the OVO management server is cancelled. In the newly displayed login window, you can choose either the other OVO management server, or cancel login procedure.
- If you choose **`Always`**, the HP OVO server certificate will be used for the current and all subsequent Java GUI sessions.

Providing Certificates for Full Authentication Mode

To provide the certificates for the full authentication mode, the following has to be performed:

❏ Enable the full authentication mode by properly configuring the `opcuihttps` process. To do so, enter the following and then restart the `opcuihttps` process:

```
ovconfchg -ovrg -ns opc.opcuihttps -set\
SSL_CLIENT_VERIFICATION_MODE RequireCertificate
```

See "Configuring the Parameters for opcuihttps" on page 23 for more information about configuring the `opcuihttps` parameters.

❏ Ensure that the client certificate is present on the client system.

In case that the HTTPS OVO agent is installed on the Java GUI client system, you can use its OVO client certificate for authentication; otherwise you should install the client certificate manually. See “Installing the Client Certificate Manually” on page 28 for details about the manual client certificate installation.

- ❑ Set the Java GUI startup parameter `lcore_defaults` to **yes**, so that Java GUI can use the default Core functionality. The Core functionality is either installed with the OVO agent in case it exists on the Java GUI client, or you have to install it additionally. See “Installing Core Functionality” on page 25 for more information.

Installing the Client Certificate Manually

IMPORTANT To install the client certificate manually, it is necessary to have installed the Core functionality beforehand. For details on the Core functionality installation, see “Installing Core Functionality” on page 25.

To install the client certificate manually on the client system, follow these steps:

1. On the Java GUI client system, create a client core id using the following command:

   ```
   ovcoreid
   ```

2. On the OVO management server, create a new certificate, associate the public key for this certificate and store it in a file. Enter the following:

   ```
   ovcm -issue -file <filename> -name <system_name> -pass
   <passphrase> -coreid <client_coreid>
   ```

   Where the `<system_name>` is the OVO management server hostname, `<passphrase>` is a password, `<client_coreid>` is the client core id, and the `<filename>` is the name of the file where the certificate is stored.

3. Transfer the file containing the certificate to the client system by means of floppy disk or using the ftp service.

4. Install the certificate on the client system. Enter the following:

   ```
   ovcert -importcert -file <filename> -pass <passphrase>
   ```

Configuring the HTTPS-based Java GUI Connection through Firewalls

To be able to configure the connection between HTTPS-based Java GUI and the OVO management server through firewalls, you have to perform two major procedures:

1. Install Core functionality.

NOTE If the HTTPS OVO agent is present on the Java GUI client system, there is no need to install the Core functionality, since it is already installed with the OVO agent.

See "Installing Core Functionality" on page 25 for more information.

2. Properly configure the Core functionality on the client system. To do so, perform the following:

   a. In the `bbc.http` namespace, set the `PROXY` parameter using the `ovconfchg` command. Enter the following:

   ```
   ovconfchg -ns bbc.http -set PROXY <proxy_config>
   ```

   Where `<proxy_config>` is the configuration of the proxy server, which includes its full hostname, port on which is running, and the name of the OVO management server to which the connection is made.

   The `PROXY` parameter defines which proxy and port to use for a certain hostname.
   Format: `proxy:port +(a)-(b);proxy2:port2+(a)-(b);` ... a: comma or semicolon divided list of hostnames for which this proxy shall be used b: comma or semicolon divided list of hostnames for which the proxy shall not be used BBC chooses the first matching proxy. Example:
   `PROXY=web-proxy:8088-(*.hp.com)+(*.bbn.hp.com;*)`
   Meaning: the proxy 'web-proxy' will be used with port 8088 for every server (*) except hosts that match *.hp.com, e.g. www.hp.com. If the hostname matches *.bbn.hp.com, e.g. merlin.bbn.hp.com the proxy server will be used to.

It is also possible to use IP addresses instead of hostnames so 15.*.*.* or 15:*.*:*.*:*.*:* would be valid as well but you have to specify the correct number of dots or colons! Default is an empty string; no proxy specified.

For example, to enable connection of OVO management server `barney.hp.com` through proxy server `proxy.hp.com`, which runs on the port 8088, enter the following:

```
ovconfchg -ns bbc.http -set PROXY \
proxy.hp.com:8088+(barney.hp.com)
```

b. Set the Java GUI parameter `lcore_defaults` to **yes** in the `ito_op` (`ito_op.bat`) startup script.

Known Problems and Workarounds

This section describes only the problems that are specific to running HTTPS-based Java GUI on the OVO management server. Recommended workarounds are provided wherever possible.

1. **Symptom**

   The `opcuihttps` process stops in any of the following situations:

   - The following `ito-e-gui` entry is missing in the `/etc/services`:

     ```
     ito-e-gui 2351/tcp        # ITO Enterprise Java GUI
     ```

   - The port number is set to the value other than 2531.

   Solution

   Make sure the port number is set to the value 2531. The option for connecting the `opcuihttps` process to other than default `opcuiwww` port is currently *not* available.

2. **Symptom**

   When exiting or logging off from Java GUI, the following error message is displayed:

   ```
   ERROR MSG, 7:42:47 AM,
   com.hp.ov.it.comm.OvEmbHttpsClient:

   https status - InternalServerError:text/html,

   Message = HTTP/1.1 500 Internal Server Error

   Date: Wed, 11 May 2005 05:41:57 GMT

   Transfer-Encoding: chunked

   Server: BBC 05.20.010; opcuihttps 01.00.000

   senderid: e6979118-aca1-750b-1f6a-de6eb9cfe391

   Cache-Control: no-cache

   Content-Type: text/html
   ```

   Solution

   This message can be safely ignored.

3. **Symptom**

   If the full SSL authentication mode is set on the OVO management server for the `opcuihttps` process, but no valid client certificate is present on the client system, Java GUI fails to connect. There are no errors or warnings shown.

   Solution

   In case that full SSL authentication mode is enabled for the `opcuihttps` process on the OVO management server, make sure that client certificate is properly installed on the client system when connecting to that OVO management server.