IceWall SSO

Version Version 10.0

UserExit Routine Developer's Manual

August 2010

Printed in Japan

HP Part No. B1544-97011

Rev.111006A

Notice

The information contained in this document is subject to change without notice.

Meticulous care has been taken in the preparation of this document, however, if a questionable or erroneous item, or an omission of content is found, please contact us.

Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damage in connection with the furnishing, performance or use of this document.

This manual and media, such as the CD-ROM supplied as a part of the product's package, are only to be used with this product.

– Table of Contents –

1 Introduction

This manual provides information required to develop the UserExit routine integrated in to the Forwarder and Authentication Module.

1.1 Version designations in the text

The table below gives the meanings of the version designations added to the description.

Designation	Meaning
10.0	An item added to the version enclosed in the square. In this case, the designation indicates the item was added to 10.0.
	An item where the specification was changed or function added to the version enclosed in the oval mark. In this case, the designation indicates a specification change or function addition to 10.0.

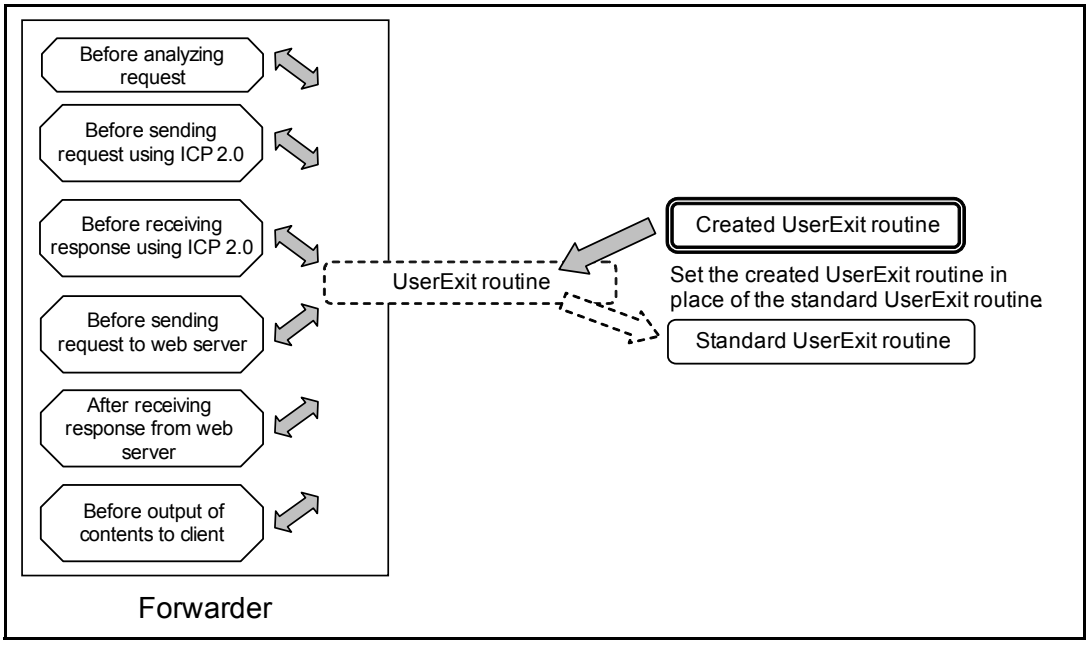
2 What is UserExit Routine?

The UserExit routine is a function that allows the addition of an arbitrary process for a specific process in the Forwarder and/or Authentication Module.
You can add a process for each process at the following locations.

Process	Location to add
Forwarder	Before analyzing a request from the client
	Before sending a login request to the Authentication Module using ICP 2.0
	After receiving a login response from the Authentication Module using ICP 2.0
	Before sending a login request (from an agent) to the Authentication Module using ICP 2.0
	After receiving a login response (to an agent) from the Authentication Module using ICP 2.0
	Before sending an access control request to the Authentication Module using ICP 2.0
	After receiving an access control response from the Authentication Module using ICP 2.0
	Before sending a logout request to the Authentication Module using ICP 2.0
	After receiving a logout response from the Authentication Module using ICP 2.0
	Before sending a password change request to the Authentication Module using ICP 2.0
	After receiving a password change response from the Authentication Module using ICP 2.0
	Before sending a request to the Backend Web Server
	After receiving a response from the Backend Web Server
	Before the output of contents to a client
Authentication Module	Before the user is authenticated
	After the user is authenticated
	After occurrence of a user authentication error
	Before access control
	After access control
	After occurrence of an- access control error
	Before changing the password
	After changing the password
	After occurrence of a password change error

Process	Location to add
Authentication Module	Before logging out
	After logging out
	After occurrence of a logout error

The UserExit routine is provided as a shared library. To add a function, you need to create a shared library for the required purpose.



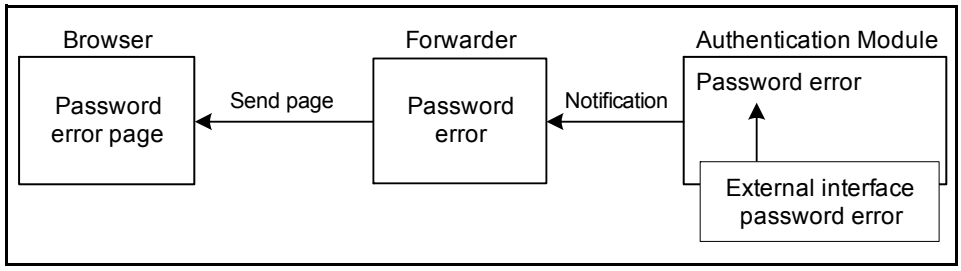
UserExit routine conceptual diagram (Forwarder)

2.1 Addition of original authentication process

By integrating an authentication process using another security product within the UserExit routine, you can implement an original and more powerful authentication function.

2.2 Display control

By returning a predefined value as the processing result from the UserExit routine, you can control pages displayed in the browser.



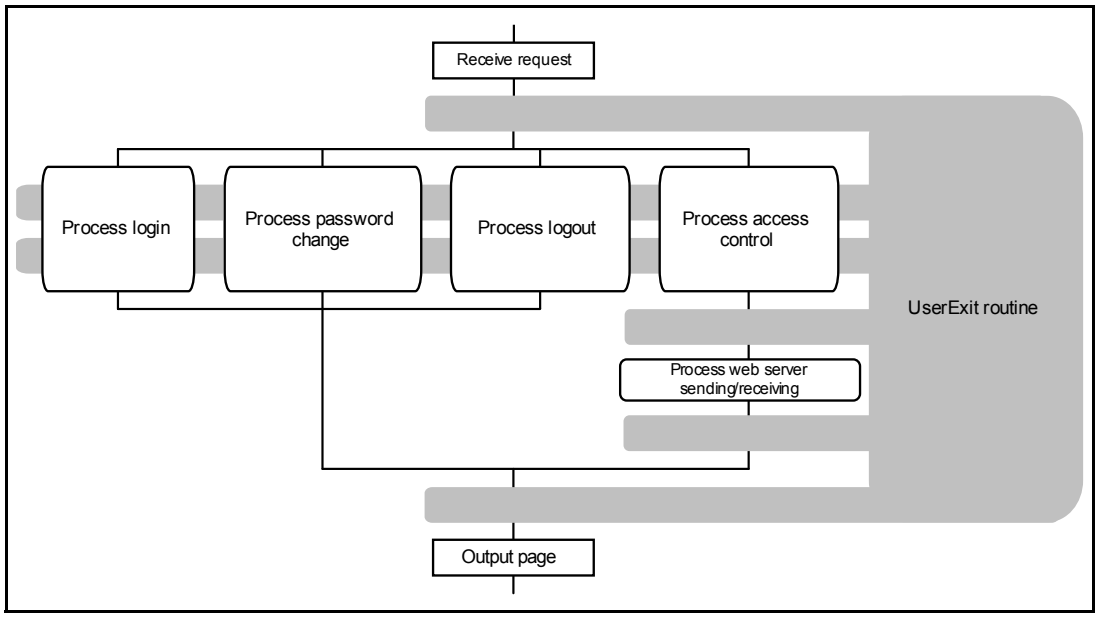
Display control example (for password error in Authentication Module)

3 Development Procedure

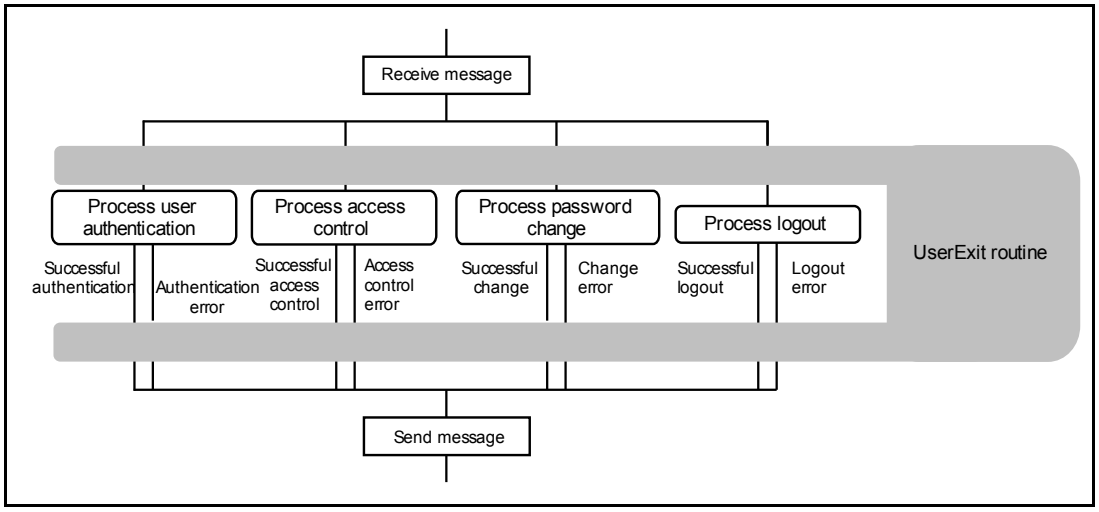
The following describes the general procedure for developing the UserExit routine.

3.1 Determining the Specifications

Determine the processing to be executed within the UserExit routine and the locations to be added.



UserExit routine overview



UserExit routine overview for Authentication Module

3.2 Coding and Building

According to the determined specifications, write the source code of the UserExit routine, and then build it into the shared library.

Use the skeleton source and makefile included in the development kit, for coding and building, respectively.

For Forwarder

```
$ cd /opt/icewall-sso/developkit/dfw/DfwExit
$ make -f Makefile
```

For Authentication Module

```
$ cd /opt/icewall-sso/developkit/certd/CertExit
$ make -f Makefile
```

3.3 Testing and Debugging

Test and debug the developed shared library using a test program and other tools. After the shared library has passed the test, integrate it into the Forwarder or Authentication Module and then perform an operation test.

Integrate the shared library into the Authentication Module in the following procedure.

For Forwarder
(1) Back up the UserExit routine installed by default.

```
$ cd /opt/icewall-sso/lib
$ cp -p libDfwExit.sl libDfwExit.sl.bak
```

(2) Copy the created UserExit routine to the target location.

```
$ cp -p /opt/icewall-sso/developkit/dfw/DfwExit/libDfwExit.sl
/opt/icewall-sso/lib
```

For Authentication Module
(1) Stop the Authentication Module.

```
$ /opt/icewall-sso/certd/bin/end-cert
```

(2) Back up the UserExit routine installed by default.

```
$ cd /opt/icewall-sso/lib
$ cp -p libCertExit.sl libCertExit.sl.bak
```

(3) Copy the created UserExit routine to the target location.

```
$ cp -p /opt/icewall-sso/developkit/certd/CertExit/
libCertExit.sl /opt/icewall-sso/lib
```

(4) Start the Authentication Module.

```
$ /opt/icewall-sso/certd/bin/start-cert
```

4 UserExit Routine Specifications

The specifications of the User Exit routine are different for the Forwarder and Authentication Module.
This chapter describes each type of the UserExit routine and available APIs.

4.1 Specifications of the UserExit routine for the Forwarder

The UserExit routine for the Forwarder defines the following:

- Interface function called by the Forwarder
- Codes returned from the interface function to control the page displayed to the client
- APIs available in the interface function

The following pages describe the specifications in detail.

IW_ExDFWInterFace

The following lists the specifications of the interface function of the User Exit routine for the Forwarder.

Format

int IW_ExDFWInterFace(EX_DFW_REQ* req, int kind)

Argument

req : User exit routine request structure. Information of the Forwarder is stored in this structure. APIs are used to manipulate the information in this structure.

kind : Location where the UserExit routine was called is stored in this structure. One of the following values is stored.

EX_DFW_KIND_REQ	: Before analyzing a request from the client
EX_DFW_KIND_LOGIN_SEND	: Before sending a login request to the Authentication Module
EX_DFW_KIND_LOGIN_RECV	: After receiving a login response from the Authentication Module
EX_DFW_KIND_ALOGIN_SEND	: Before sending a login request (from agent) to the Authentication Module
EX_DFW_KIND_ALOGIN_RECV	: After receiving a login response (to an agent) from the Authentication Module
EX_DFW_KIND_ACC_SEND	: Before sending an access control request to the Authentication Module
EX_DFW_KIND_ACC_RECV	: After receiving an access control response from the Authentication Module
EX_DFW_KIND_LOGOUT_SEND	: Before sending a logout request to the Authentication Module
EX_DFW_KIND_LOGOUT_RECV	: After receiving a logout response from the Authentication Module
EX_DFW_KIND_PWDCHG_SEND	: Before sending a password change request to the Authentication Module
EX_DFW_KIND_PWDCHG_RECV	: After receiving a password change response from the Authentication Module
EX_DFW_KIND_SEND	: Before sending a request to the Backend Web Server

EX_DFW_KIND_RECV : After receiving a response from the Backend Web Server
EX_DFW_KIND_END : Before the output of contents

Return value

Returns a code to control the page displayed to the client.
For the available codes, see “Client display control codes for UserExit routine for Forwarder” below.

Restrictions

The UserExit routine **may not be executed** depending on the contents of the configuration file. **The following lists conditions** where the routine is not executed and the location of the interface function.

Location	Condition when routine is not executed
EX_DFW_KIND_REQ	•Always executed
EX_DFW_KIND_LOGIN_SEND	•Except for login using ICP 2.0
EX_DFW_KIND_LOGIN_RECV	
EX_DFW_KIND_ALOGIN_SEND	•Except for login request from agent using ICP 2.0
EX_DFW_KIND_ALOGIN_RECV	
EX_DFW_KIND_ACC_SEND	•Except for access control using ICP 2.0
EX_DFW_KIND_ACC_RECV	
EX_DFW_KIND_LOGOUT_SEND	•Except for logout using ICP 2.0
EX_DFW_KIND_LOGOUT_RECV	
EX_DFW_KIND_PWDCHG_SEND	•Except for password change using ICP 2.0
EX_DFW_KIND_PWDCHG_RECV	
EX_DFW_KIND_SEND	•Except for access control
EX_DFW_KIND_RECV	•Access control with BUFFER parameter set to 0 and type of received contents (Content-Type) not set in CTYPE parameter
EX_DFW_KIND_END	

The contents of the req and kind arguments passed by this function cannot be changed. The operation is not guaranteed if they are changed.

Client display control codes for the UserExit routine for Forwarder

The client display control codes available for use in the UserExit routine for Forwarder are listed below. For more information on the pages displayed, see the "IceWall SSO Standard HTML Customization Guide" and "IceWall SSO Reference Manual."

Control code	Description	
EX_DFW_OK	Description	Used to begin a Forwarder process after the internal processing of the UserExit routine is complete.
	Displayed page	The page displayed when this code is used depends on the Forwarder operation that takes over.
EX_DFW_LOGIN	Description	Used to request the client for login by the user ID.
	Displayed page	Login page for user ID and password (login.html)
EX_DFW_LOGINCERT	Description	Used to request the client for login by the certificate. To use this control code, the "IceWall SSO Client Certificates Option" is required.
	Displayed page	Login page for client certificate and password (login_cert.html)
EX_DFW_LOGINFORCE	Description	Used to request the client for forced login.
	Displayed page	Forced Login page (login_force_tkt.html). Note that operation is not guaranteed if this control code is used other than before request analysis.
EX_DFW_LOGINUIDERR	Description	Used to treat a login request from the client as a user ID error.
	Displayed page	User ID Error page (login_userid_error.html)
EX_DFW_LOGINPWDERR	Description	Used to treat a login request from the client as a password error.
	Displayed page	Password Error page (login_pwd_error.html)
EX_DFW_LOGINLOCKERR	Description	Used to treat a login request from the client as an account lock error.
	Displayed page	Account Lock Error page (login_lock_error.html)
EX_DFW_LOGINCERTERR	Description	Used to treat a login request from the client as a pre-authenticated by client certificate error. To use this control code, the "IceWall SSO Client Certificates Option" is required.
	Displayed page	Pre-authenticated Error page (login_cert_error.html)

Control code	Description	
EX_DFW_LOGINSERIALERR	Description	Used to treat a login request from the client as a client certificate serial number error. To use this control code, the “IceWall SSO Client Certificates Option” is required.
	Displayed page	Certificate Serial Number Error page (login_error.html)
EX_DFW_LOGINGRPERR	Description	Used to treat a login request from the client as a no group error.
	Displayed page	No Group Error page (login_group_error.html)
EX_DFW_LOGINSTOP	Description	Used to treat a login request from the client as a login stop error.
	Displayed page	Login Stop Error page (login_stop.html)
EX_DFW_LOGINLIMITERR	Description	Used to treat a login request from the client as a login limit error. * Currently, the UserExit routine for the Authentication Module cannot display the Login Limit Error page.
	Displayed page	Login Limit Error page (login_limit_error.html)
EX_DFW_LOGOUT	Description	Used to request the client for logout.
	Displayed page	Logout page (logout.html)
EX_DFW_LOGOUTOK	Description	Used to treat a login request from the client to be successful.
	Displayed page	Logout Successful page (logout_ok.html)
EX_DFW_LOGOUTNO	Description	Used to treat a login request from the client as a logged out error.
	Displayed page	Logout Failed page (logout_no.html)
EX_DFW_LOGOUTERR	Description	Used to treat a login request from the client as a logout failed error.
	Displayed page	Logout error page (logout_error.html)
EX_DFW_ACCESS	Description	Used to treat an access control request from the client as an access permission error.
	Displayed page	Access Privileges Error page (access_error.html)
EX_DFW_SENDERR	Description	Used to treat data transmission from the client as a data send error.
	Displayed page	Data Send Error page (datasend_error.html)
EX_DFW_PWDCHG	Description	Used to request the client to change the password.
	Displayed page	Password Change page (pwdchg.html)

Control code	Description	
EX_DFW_PWDCHGOK	Description	Used to treat a password change request from the client to be successful.
	Displayed page	Password Change Success page (pwdchg_ok.html)
EX_DFW_PWDOLDERR	Description	Used to treat a password change request from the client as an old password entry error.
	Displayed page	Current Password Entry Error page (pwdchg_oldpasser.r.html)
EX_DFW_PWDREERR	Description	Used to treat a password change request from the client as a new password entry error.
	Displayed page	New Password Entry Error page (pwdchg__repasserr.html)
EX_DFW_PWDPCYERR	Description	Used to treat a password change request from the as a password policy error.
	Displayed page	Password Change Policy Error page (pwdchg_policy_err.html)
EX_DFW_PWDVIOERR	Description	Used to treat a password change request from the client as a password change prohibited error.
	Displayed page	Password Change Prohibited Error Page (pwdchg_pwvioerr.html)
EX_DFW_PWDNOLOGIN	Description	Used to treat a login request from the client as a logged out error.
	Displayed page	Password Change No Login Error page (pwdchg_nologin.html)
EX_DFW_PWDERR	Description	Used to treat a password change request from the client as a password change error.
	Displayed page	Password Change Error page (pwdchg_err.html)
EX_DFW_PWDWARNING	Description	Used to warn the client of password expiration.
	Displayed page	Password Expiration Warning page (pwdchg_warning.html)
EX_DFW_SYSERR	Description	Used to treat a request from the client as a system error.
	Displayed page	System Error page (system_error.html)
EX_DFW_ALIASNO	Description	Used to treat a request from the client as a no alias error.
	Displayed page	No Alias Error page (system_server_error.html)
EX_DFW_ALIASBAD	Description	Used to treat a request from the client as an undefined alias error.
	Displayed page	Undefined Alias Error page (system_alias_errorr.html)
EX_DFW_DOWNCERTD	Description	Used to treat a request from the client as an Authentication Module down error.
	Displayed page	Authentication Module Down Error page (system_cert_error.html)

Control code	Description	
EX_DFW_DOWNDB	Description	Used to treat a request from the client as an Authentication DB down error.
	Displayed page	Authentication DB Down Error page (system_ldap_error.html)
EX_DFW_DOWNBKEND	Description	Used to treat a request from the client as Backend Web Server down error.
	Displayed page	Backend Web Server Down Error page (system_backend_error.html)
EX_DFW_TOUTCERTD	Description	Used to treat a request from the client as an Authentication Module receive timeout error.
	Displayed page	Authentication Module Receive Timeout Error page (system_timeout_certd.html)
EX_DFW_TOUTBKEND	Description	Used to treat a request from the client as Backend Web Server receive timeout error.
	Displayed page	Backend Web Server Receive Timeout Error page (system_timeout_bkend.html)
EX_DFW_FILTERGE	Description	Used to treat a request from the client as a GET filter error.
	Displayed page	GET Filter Error page (filter_get_error.html)
EX_DFW_FILTERPOST	Description	Used to treat a request from the client as a POST filter error.
	Displayed page	POST Filter Error page (filter_post_error.html)
EX_DFW_FILTERHTML	Description	Used to treat a request from the client as an HTML filter error.
	Displayed page	HTML Filter Error page (filter_html_error.html)
EX_DFW_FILTERHOST	Description	Used to treat a request from the client as a host filter error.
	Displayed page	Host Filter Error page (filter_svr_error.html)
EX_DFW_LOGINTIMEERR	Description	Used when timeout has occurred before sending POST data to the Forwarder during the password change process.
	Displayed page	Password Change Send Timeout Error page (pwdchg_postlimit_err.html)
EX_DFW_PWDTIMEERR	Description	Used when timeout has occurred before sending POST data to the Forwarder during the password change process.
	Displayed page	Password Change Send Timeout Error page (pwdchg_postlimit_err.html)
EX_DFW_FILTERREQ 10.0	Description	Used when a request of an unaccepted type has been issued.
	Displayed page	Request Filter Error page (filter_request_error.html)

Control code	Description	
EX_DFW_DBBUSYERR	Description	Used when the Authentication Module cannot process the request due to congestion of database accesses.
	Displayed page	DB Busy Error page (system_busy_database.html)
EX_DFW_REQACLERR 10.0	Description	Used when connection to the Authentication Module is not permitted.
	Displayed page	Request ACL Error page (request_acl_error.html)
EX_DFW_POSTERR	Description	Used to treat a request from the client as a POST data maximum size error.
	Displayed page	Post Data Maximum Size Error page (max_postsize_error.html)
EX_DFW_USREX1	Description	Used to treat a request from the client as a user-defined error 1.
	Displayed page	User-Defined Error 1 page (usr_ext1.html)
EX_DFW_USREX2	Description	Used to treat a request from the client as a user-defined error 2.
	Displayed page	User-Defined Error 2 page (usr_ext2.html)
EX_DFW_USREX3	Description	Used to treat a request from the client as a user-defined error 3.
	Displayed page	User-Defined Error 3 page (usr_ext3.html)
EX_DFW_USREX4	Description	Used to treat a request from the client as a user-defined error 4.
	Displayed page	User-Defined Error 4 page (usr_ext4.html)
EX_DFW_USREX5	Description	Used to treat a request from the client as a user-defined error 5.
	Displayed page	User-Defined Error 5 page (usr_ext5.html)
EX_DFW_USREX6	Description	Used to treat a request from the client as a user-defined error 6.
	Displayed page	User-Defined Error 6 page (usr_ext6.html)

4.1.1 APIs in the interface function

The APIs available in the interface function are listed below.

API name	Description
IW_ExDFWChangeProtocol	Changes the operation protocol of the Forwarder.
IW_ExDFWGetMethod	Acquires the method requested to the Forwarder.
IW_ExDFWModifyMethod	Changes the method requested to the Forwarder.
IW_ExDFWGetPostdata	Acquires the POST data sent to the Forwarder.
IW_ExDFWModifyPostdata	Changes the POST data sent to the Forwarder to arbitrary POST data.
IW_ExDFWGetBuffer	Acquires the data buffer to communicate with the Backend Web Server.
IW_ExDFWModifyBuffer	Changes the data buffer to communicate with the Backend Web Server. Used for text data.
IW_ExDFWModifyBufferLength	Changes the data buffer to communicate with the Backend Web Server. Used for binary data.
IW_ExDFWModifyRequestPath	Changes the request path to the Backend Web Server.
IW_ExDFWModifyPathinfo	Changes the string after the alias of the URL requested for for the Forwarder.
IW_ExDFWGetUid	Acquires the user ID of the login user.
IW_ExDFWGetCertData	Acquires the data buffer to communicate with the Authentication Module.
IW_ExDFWCtrlCertData	Manipulates the data buffer to communicate with the Authentication Module.

The following describes how to use these APIs.

IW_ExDFWChangeProtocol

Format **int IW_ExDFWChangeProtocol(EX_DFW_REQ* req, int kind, int flg)**

Function Changes the operation protocol of the Forwarder.
When the protocol was changed to SSL in the Forwarder configuration file
(DFW_PROTOCOL=1), it can be changed back to HTTP using this API.

Argument **req** : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

flg : This flag sets the operating protocol of the Forwarder. Set one of the following values:
EX_DFW_HTTP : Sets the protocol to HTTP.
EX_DFW_HTTPS: Sets the protocol to SSL.

Return value One of the following values is returned.
EX_DFW_SUCCESS : Normal termination
EX_DFW_ERROR : Abnormal termination
EX_DFW_BADKIND : The entry cannot be used by the API, or the kind value is invalid

Restrictions Available only when the value of the kind argument of the interface function is EX_DFW_KIND_REQ.

IW_ExDFWGetMethod

Format **char *IW_ExDFWGetMethod (EX_DFW_REQ* req, int kind)**

Function Acquires the method requested to the Forwarder.

Argument **req** : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

Return value One of the following values is returned.

Other than NULL: Pointer to a string representing a method. To edit the data, copy it to another area in advance.

NULL : Abnormal termination, or the entry cannot be used by the API

Restrictions Available only when the value of the kind argument of the interface function is EX_DFW_KIND_REQ.

The string pointed by the return value must not be changed. The operation is not guaranteed if it is changed.

IW_ExDFWModifyMethod

Format **int IW_ExDFWModifyMethod (EX_DFW_REQ* req, int kind, int method)**

Function Changes the method requested to the Forwarder.

Argument **req** : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

method : This flag specifies a method to change. Set one of the following values:
EX_DFW_METHOD_GET : GET method
EX_DFW_METHOD_POST : POST method

Return value One of the following values is returned.
EX_DFW_SUCCESS : Normal termination
EX_DFW_ERROR : Abnormal termination
EX_DFW_BADKIND : The entry cannot be used by the API, or the kind value is invalid

Restrictions Available only when the value of the kind argument of the interface function is EX_DFW_KIND_REQ.

IW_ExDFWGetPostdata

Format **char* IW_ExDFWGetPostdata (EX_DFW_REQ* req, int kind)**

Function Acquires the POST data sent to the Forwarder.
If the POST data is encoded into an URL when sent from the browser, it is obtained by this API as encoded.

Argument **req** : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

Return value One of the following values is returned.
Other than NULL: Pointer to a buffer which stores the POST data. To edit the data, copy it to another area in advance.
NULL : Abnormal termination, or the entry cannot be used by the API

Restrictions Available only when the value of the kind argument of the interface function is EX_DFW_KIND_REQ.

The string pointed by the return value must not be changed. The operation is not guaranteed if it is changed.

Note that when the FORCELOGIN_ENC parameter is used for encryption, the user ID and password in the POST data sent from the Forced Login page are also encrypted.

IW_ExDFWModifyPostdata

Format

int IW_ExDFWModifyPostdata (EX_DFW_REQ* req, int kind, char* buf)

Function

Changes the POST data sent to the Forwarder to arbitrary POST data.
If the POST data to be changed is encoded into an URL when sent from the browser, it must be URL-encoded even when using this API.
By changing the POST data, the Content-Length header is automatically modified.

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

buf : Set the pointer to a buffer that stores the POST data to change. You can destroy the buffer after using this API.

Return value

One of the following values is returned.

EX_DFW_SUCCESS	: Normal termination
EX_DFW_ERROR	: Abnormal termination
EX_DFW_BADKIND	: The entry cannot be used by the API, or the kind value is invalid

Restrictions

Available only when the value of the kind argument of the interface function is EX_DFW_KIND_REQ.

IW_ExDFWGetBuffer

Format

char* IW_ExDFWGetBuffer (EX_DFW_REQ* req, int kind)

Function

Acquires the request data sent to the Backend Web Server, response data received from the Backend Web Server, or contents just before being output to the browser.
One of the following types of data is obtained according to the kind argument:

EX_DFW_KIND_SEND : Request data sent to Backend Web Server
EX_DFW_KIND_RECV : Response data received from Backend Web Server
EX_DFW_KIND_END : Contents just before being output to browser

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

Return value

One of the following values is returned.

Other than NULL: Pointer to a buffer that stores the data. To edit the data, copy it to another area in advance.
NULL : Abnormal termination, or the entry cannot be used by the API

Restrictions

Available only when the value of the kind argument of the interface function is EX_DFW_KIND_SEND, EX_DFW_KIND_RECV, or EX_DFW_KIND_END.

The string pointed by the return value must not be changed. The operation is not guaranteed if it is changed.

IW_ExDFWModifyBuffer

Format

int IW_ExDFWModifyBuffer (EX_DFW_REQ* req, int kind, char* buf)

Function

Changes the request data sent to the Backend Web Server, response data received from the Backend Web Server, or contents just before being output to the browser to arbitrary data.
One of the following types of data is changed according to the kind argument:

EX_DFW_KIND_SEND : Request data sent to Backend Web Server
EX_DFW_KIND_RECV : Response data received from Backend Web Server
EX_DFW_KIND_END : Contents just before being output to browser

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

buf : Set the pointer to a buffer that stores the POST data to be changed. You can destroy the buffer after using this API.

Return value

One of the following values is returned.

EX_DFW_SUCCESS : Normal termination
EX_DFW_ERROR : Abnormal termination, the entry cannot be used by the API, or the kind value is invalid

Restrictions

Available only when the value of the kind argument of the interface function is EX_DFW_KIND_SEND, EX_DFW_KIND_RECV, or EX_DFW_KIND_END.

To change the request or response data, convert it to the correct format. The operation is not guaranteed if, for example, the request data is changed to the response data format buffer.

* If the buffer to be changed contains binary data, use IW_ExDFWModifyBufferLength() mentioned later instead of this API.

IW_ExDFWModifyBufferLength

Format

int IW_ExDFWModifyBufferLength (EX_DFW_REQ* req, int kind, char* buf, int len)

Function

Changes the request data sent to the Backend Web Server, response data received from the Backend Web Server, or contents just before being output to the browser to arbitrary data.
Binary data version of IW_ExDFWModifyBuffer().
One of the following types of data is changed according to the kind argument:

EX_DFW_KIND_SEND : Request data sent to Backend Web Server
EX_DFW_KIND_RECV : Response data received from Backend Web Server
EX_DFW_KIND_END : Contents just before being output to browser

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

buf : Set the pointer to a buffer that stores the POST data to change. You can destroy the buffer after using this API.

len : Specify the length of the data buffer to be changed.

Return value

One of the following values is returned.

EX_DFW_SUCCESS : Normal termination
EX_DFW_ERROR : Abnormal termination
EX_DFW_BADKIND : The entry cannot be used by the API, or the kind value is invalid

Restrictions

Available only when the value of the kind argument of the interface function is EX_DFW_KIND_SEND, EX_DFW_KIND_RECV, or EX_DFW_KIND_END.

To change the request or response data, convert it to the correct format. The operation is not guaranteed if, for example, the request data is changed to the response data format buffer.

IW_ExDFWModifyRequestPath

Format

int IW_ExDFWModifyRequestPath(EX_DFW_REQ* req, int kind, char *path)

Function

Changes the content path for the Backend Web Server.

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

path : Set the path to the new contents.

Return value

One of the following values is returned.

EX_DFW_SUCCESS : Normal termination
EX_DFW_ERROR : Abnormal termination

Restrictions

When the content path is changed using this API, the operation changes depending on the changed location.

Location	Operation after change
EX_DFW_KIND_REQ	•Performs access control using the changed content path. •If the Set-Cookie header is sent from the changed path without the path attribute, the cookie is set for the old path. •The changed path is recorded in the access log of the Authentication Module. •The changed path is recorded in the access log of the Forwarder.
EX_DFW_KIND_LOGIN_SEND	•Logs into the changed content path and then redirects the contents.
EX_DFW_KIND_LOGIN_RECV	•Logs into the changed content path and then redirects the contents.
EX_DFW_KIND_ALOGIN_SEND	•Can be changed but the operation is not affected.

Location	Operation after change
EX_DFW_KIND_ALOGIN_RECV	•Can be changed but the operation is not affected.
EX_DFW_KIND_ACC_SEND	•Performs access control using the old content path but actually obtains the contents from the changed path. •If the Set-Cookie header is sent from the changed path without the path attribute, the cookie is set for the old path. •The old path is recorded in the access log of the Authentication Module. •The changed path is recorded in the access log of the Forwarder.
EX_DFW_KIND_ACC_RECV	•Performs access control using the old content path but actually obtains the contents from the changed path. •If the Set-Cookie header is sent from the changed path without the path attribute, the cookie is set for the old path. •The old path is recorded in the access log of the Authentication Module. •The changed path is recorded in the access log of the Forwarder.
EX_DFW_KIND_LOGOUT_SEND	•Returns EX_DFW_ERROR because the SESSION=0 request path does not exist. •The SESSION=1 request path is changed but operation is not affected.
EX_DFW_KIND_LOGOUT_RECV	•Returns EX_DFW_ERROR because the SESSION=0 request path does not exist. •The SESSION=1 request path is changed but operation is not affected.
EX_DFW_KIND_PWDCHG_SEND	•Returns EX_DFW_ERROR because the SESSION=0 request path does not exist. •The SESSION=1 request path is changed but operation is not affected.
EX_DFW_KIND_PWDCHG_RECV	•Returns EX_DFW_ERROR because the SESSION=0 request path does not exist. •The SESSION=1 request path is changed but operation is not affected.

Location	Operation after change
EX_DFW_KIND_SEND	•Performs access control using the old content path but actually obtains the contents from the changed path. •If the Set-Cookie header is sent from the changed path without the path attribute, the cookie is set for the old path. •The old path is recorded in the access log of the Authentication Module. •The changed path is recorded in the access log of the Forwarder.
EX_DFW_KIND_RECV	•The changed path is recorded in the access log of the Forwarder.
EX_DFW_KIND_END	•The changed path is recorded in the access log of the Forwarder.

IW_ExDFWModifyPathinfo

Format

int IW_ExDFWModifyPathinfo(EX_DFW_REQ* req, int kind, char *pathinfo)

Function

Changes the string after the alias of the URL requested to the Forwarder.

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

pathinfo : Set the string after the alias for the new URL. You can delete the string after the alias by specifying NULL.

Return value

One of the following values is returned.
EX_DFW_SUCCESS : Normal termination
EX_DFW_ERROR : Abnormal termination
EX_DFW_BADKIND : The entry cannot be used by the API, or the kind value is invalid

Restrictions

Available only when the value of the kind argument of the interface function is EX_DFW_KIND_REQ.

IW_ExDFWGetUid

Format

char *IW_ExDFWGetUid(EX_DFW_REQ* req, int kind, int flg)

Function

Acquires the user ID of the accessing user.

This API is available with Version 8.0 or later.

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

flg : Uses the get user ID information flag. Set one of the following values:

EX_DFW_UID_LOGIN : Gets user ID of login user
EX_DFW_UID_INPUT : Gets user ID entered at login

Return value

One of the following values is returned.

Other than NULL: Pointer to a buffer that stores the user ID (this buffer is read-only. To edit the data, copy it to another area in advance.
NULL : The user ID is not stored, abnormal termination, or the entry cannot be used by the API

Restrictions

The login user ID (user ID of the currently logged in user) can be acquired only when the value of the kind argument of the interface function is EX_DFW_KIND_END. When the value is EX_DFW_KIND_SEND or EX_DFW_KIND_RECV, the acquired user ID can be used for access control only.

The value of the kind argument used for communication with the Authentication Module cannot be used to acquire the user ID.

When the client certificate is being used, the user ID for the client certificate is acquired.

An entered user ID (user ID entered from the client on the login page and the user may not be logged in) can be acquired only when the process is

login and the kind argument value of the interface function is EX_DFW_KIND_END.

This API cannot be used to acquire the user ID for the logout process.

The user ID stored in the Authentication DB is acquired as is as the login user ID; however, if it has been changed by the UserExit routine upon login with the Authentication Module, the changed user ID is acquired.

An entered user ID can be up to 64 characters in length. If a user ID exceeding 64 characters is entered, a login error occurs and the user ID cannot be acquired.

If the entered user ID was URL-encoded when it was sent, it is obtained as decoded. Character codes are obtained as they were entered.

IW_ExDFWGetCertData

Format

char *IW_ExDFWGetCertData(EX_DFW_REQ* req, int kind, char *name, int target)

Function

Acquires a value from request data sent to the Authentication Module or response data received from the Authentication Module using ICP 2.0 in the format "name: value."
One of the following buffers is acquired according to the kind argument:
EX_DFW_KIND_*_SEND: Request data sent to the Authentication Module
EX_DFW_KIND_*_RECV: Response data received from the Authentication Module

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

name : Specify the name of data to be acquired. The name is not case-sensitive.

target : Specify the target to acquire data. Set one of the following values:
EX_DFW_ICP_HEADER : ICP2.0 header
EX_DFW_ICP_BODY : ICP2.0 body

Return value

One of the following values is returned.
Other than NULL: Pointer to a buffer that stores the data (this buffer is read-only. To edit the data, copy it to another area in advance.
NULL : ICP 1.0 is used, the specified data name does not exist, abnormal termination, or the entry cannot be used by the API

Restrictions

Available only when the value of the kind argument of the interface function is EX_DFW_KIND_LOGIN_SEND, EX_DFW_KIND_LOGIN_RECV, EX_DFW_KIND_LOGOUT_SEND, EX_DFW_KIND_LOGOUT_RECV, EX_DFW_KIND_ACC_SEND, EX_DFW_KIND_ACC_RECV, EX_DFW_KIND_PWDCHG_SEND,

EX_DFW_KIND_PWDCHG_RECV, EX_DFW_KIND_ALOGIN_SEND, or EX_DFW_KIND_ALOGIN_RECV.

Available only when ICP 2.0 is used.

The status line contained in the ICP 2.0 header cannot be acquired.

IW_ExDFWCtrlCertData

Format

int IW_ExDFWCtrlCertData(EX_DFW_REQ* req, int kind, char *name, char *value, int target, int mode)

Function

Controls the request data sent to the Authentication Module and the response data received from the Authentication Module.
One of the following types of data is changed according to the kind argument:
EX_DFW_KIND_*_SEND: Request data sent to the Authentication Module
EX_DFW_KIND_*_RECV: Response data received from the Authentication Module

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

name : Specify the target data name ("name" in "name: value"). The name is not case-sensitive.

value : Specify the target data value ("value" in "name: value"). Specify "NULL" to delete the target.

target : Specify the target to control data. Set one of the following values:
EX_DFW_ICP_HEADER : ICP2.0 header
EX_DFW_ICP_BODY : ICP2.0 body

mode : Specify how to control data. Set one of the following values:
EX_DFW_ICP_ADD : Process to append "name: value
EX_DFW_ICP_MOD: Process to change "value" in "name: value"
EX_DFW_ICP_DEL : Process to delete the entire "name: value

Return value

One of the following values is returned.

EX_DFW_SUCCESS	: Normal termination
EX_DFW_ERROR	: Abnormal termination
EX_DFW_BADKIND	: The entry cannot be used by the API, or the kind value is invalid

EX_DFW_NOTFOUND: The target is not found (for change or deletion)

Restrictions

Available only when the value of the kind argument of the interface function is EX_DFW_KIND_LOGIN_SEND, EX_DFW_KIND_LOGIN_RECV, EX_DFW_KIND_LOGOUT_SEND, EX_DFW_KIND_LOGOUT_RECV, EX_DFW_KIND_ACC_SEND, EX_DFW_KIND_ACC_RECV, EX_DFW_KIND_PWDCHG_SEND, EX_DFW_KIND_PWDCHG_RECV, EX_DFW_KIND_ALOGIN_SEND, or EX_DFW_KIND_ALOGIN_RECV.

Available only when ICP 2.0 is used.

The status line contained in the ICP 2.0 header section cannot be controlled.

If the data sent to the Authentication Module contains binary data, communication cannot be performed correctly.

If you specify "" (no value) for "value" in an addition or change process, data does not
have a value as in "name:."

4.2 Specifications of the UserExit routine for the Authentication Module

The UserExit routine for the Authentication Module defines the following:

- Interface function called by the Authentication Module
- Code returned from the interface function to control the page displayed to the client
- APIs available in the interface function

The following pages describe the specifications in detail.

IW_ExInterFace 10.0

The following lists the specifications of the interface function of the User Exit routine for the Authentication Module.

Format

int IW_ExInterFace(int kind, char* userid, char* password, char* sessionid, char* data, EX_CS_REQ *req)

Argument

kind : Location where the UserExit routine was called is stored in this structure. One of the following values is stored.

EX_LOGIN_B	: Before user authentication
EX_LOGIN_A	: After user authentication
EX_LOGIN_A_ERR	: After user authentication error
EX_ACCESS_B	: Before access permission check
EX_ACCESS_A	: After access permission check
EX_ACCESS_A_ERR	: After access permission check error
EX_LOGOUT_B	: Before logout
EX_LOGOUT_A	: After logout
EX_LOGOUT_A_ERR	: After logout error
EX_PASSWD_B	: Before password change
EX_PASSWD_A	: After password change
EX_PASSWD_A_ERR	: After password change error

userid : The user ID entered upon logging-in from the browser. This value can be modified before user authentication (EX_LOGIN_B) only. If it is modified within the UserExit routine, the modified user ID is used for authentication. The Backend Web Server is also notified of the modified user ID.
When modifying the user ID, be sure to keep its length within 64 bytes.

password : The password entered upon logging-in from the browser. This value can be modified before user authentication (EX_LOGIN_B) only. If it is modified within the UserExit routine, the modified password is used for authentication. The Backend Web Server is also notified of the modified password.
When modifying the password, be sure to keep its length within 128 bytes.

sessionid : Session ID for the IceWall SSO. This argument cannot be modified.

data : Additional information for the location where the UserExit routine was called is stored in this structure. One of the following is set depending on where the UserExit routine is called. If it is called after an error in each process, an error code is added.

- Before user authentication, after user authentication (user authentication successful)

N_UID : Normal login
F_UID : Forced login (duplicate login forbidden and exclusive login)
N_CERT : Authentication by client certificate (normal login)
F_CERT : Authentication by client certificate (forced login)

- After user authentication error

N_UID,ErrCode : Error code to indicate the cause of the normal login error is added.
F_UID,ErrCode : Error code to indicate the cause of the forced login (duplicate login forbidden and exclusive login) error is added.
N_CERT,ErrCode : : Error code to indicate the cause of the error in authentication by the client certificate (normal login) is added.
F_CERT,ErrCode : : Error code to indicate the cause of the error in authentication by the client certificate (forced login) is added.

- Before access permission check, after access permission check

URL for access check

- After access permission check error

URL used for access check, error code

- Before logout, after logout

MANUAL : Logout by user operation
AUTO : Logout by session timeout
FORCED : Forced logout due to duplicated login by the same user
FORCEALL: Forced logout of all users 10.0

- After logout error

MANUAL,ErrCode : Error code to indicate the cause of the error during logout by user operation is added.
AUTO,ErrCode : Error code to indicate the cause of the error during logout by session timeout is added.
FORCED,ErrCode : Error code to indicate the cause of the error during forced logout due to duplicate login by the same user is added.
FORCEALL,ErrCode: Error code to indicate the cause of the error during forced logout of all users is added. 10.0

- Before password change
  New password entered from the browser to change the password
  Example: When the new password is "passwd"
  “passwd”

- After password change
  New password entered from the browser to change the password
  Example: When the new password is “passwd”
  “passwd”

- After password change error
  Error code to indicate the cause of the error is added to the new password entered from the browser to change the password
  Example: When the new password is “passwd”
  “passwd,14”

req : User exit routine request structure. Information of the Authentication Module is stored in this structure. APIs are used to manipulate the information in this structure.

Return value

Returns a code to control the page displayed to the client.
See the next section for the available codes.

Restrictions

Arguments passed by this interface function cannot be modified except for userid and password. The operation is not guaranteed if you modify other arguments.

If a UserExit routine developed with a version prior version 10.0 analyzes additional information for logout, pay attention to new arguments added to version 10.0. 10.0

Client display control codes for UserExit routine for Authentication Module

The client display control codes available for use in the UserExit routine for the Authentication Module are listed below. For more information on the pages that are displayed, see the “IceWall SSO Standard HTML Customization Guide” and “IceWall SSO Reference Manual.”

Control code	Description	
EXR_OK	Description	Used to begin an Authentication Module process after the internal processing of the UserExit routine is complete.
	Displayed page	The page displayed when this code is used depends on the Authentication Module operation that takes over.
	Available location	Available at all locations
EXR_UIDERR	Description	Used to treat a login request from the client as a user ID error.
	Displayed page	User ID Error page (login_userid_error.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR
EXR_PWDERR	Description	Used to treat a login request from the client as a password error.
	Displayed page	Password Error page (login_pwd_error.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR
EXR_PWDCHG	Description	Used to respond to a login or access request from the client with a request to change the password.
	Displayed page	Password Change page (pwdchg.html)
	Available location	EX_LOGIN_A, EX_ACCESS_B, EX_ACCESS_A, EX_ ACCESS_A_ERR
EXR_USRLOCK	Description	Used to treat a login request from the client as an account lock error.
	Displayed page	Account Lock Error page (login_lock_error.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR
EXR_LOGINNO	Description	Used to treat a login request from the client as a login prohibited error.
	Displayed page	Login Stop Error page (login_stop.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR

Control code	Description	
EXR_DLOGIN	Description	Used to respond to a login request from the client with a request for a forced login.
	Displayed page	Forced Login page (login_force_tkt.html).
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR
EXR_GRPERR	Description	Used to treat a login request from the client as a no group error.
	Displayed page	No Group Error page (login_lock_error.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR
EXR_PLOGIN	Description	Used to treat a login request using the user ID from the client as a pre-authenticated error. To use this control code, the “IceWall SSO Client Certificates Option” is required.
	Displayed page	Pre-authenticated Error page (login_cert_error.html)
	Available location	EX_LOGIN_B*1, EX_LOGIN_A*1, EX_LOGIN_A_ERR*1
EXR_PWDWARN	Description	Used to notify the client of password expiration.
	Displayed page	Password Expiration Warning page (pwdchg_warning.html)
	Available location	EX_LOGIN_A
EXR_SNOERR	Description	Used to treat a login request using the client certificate from the client as a certificate serial number error. To use this control code, the “IceWall SSO Client Certificates Option” is required.
	Displayed page	Certificate Serial Number Error page (login_error.html)
	Available location	EX_LOGIN_B*1, EX_LOGIN_A*1, EX_LOGIN_A_ERR*1
EXR_RLOGIN	Description	Used to respond to an access request from the client with a request for re-login.
	Displayed page	Login page (login.html, when using client certificate: login_cert.html)
	Available location	EX_ACCESS_B, EX_ACCESS_A, EX_ACCESS_A_ERR
EXR_ACCERR	Description	Used to treat an access request from the client as an access permission error.
	Displayed page	Access Privileges Error page (access_error.html)
	Available location	EX_ACCESS_B, EX_ACCESS_A, EX_ACCESS_A_ERR

Control code	Description	
EXR_LOGOUTNG	Description	Error code added as additional information upon an error after logout. This code cannot be used as a control code.
	Displayed page	—
	Available location	—
EXR_PWDPOERR	Description	Used to treat a password change request from the client as a password policy error.
	Displayed page	Password Policy Error page (pwdchg_policy_error.html)
	Available location	EX_PASSWD_B, EX_PASSWD_A, EX_PASSWD_A_ERR
EXR_PWDVIOERR	Description	Used to treat a password change request from the client as a no password change permission error.
	Displayed page	No Password Change Permission Error page (pwdchg_vio_error.html)
	Available location	EX_PASSWD_B, EX_PASSWD_A, EX_PASSWD_A_ERR
EXR_PWDVIOERR	Description	Used to respond to a login or password change request from the client with an Authentication DB down error.
	Displayed page	Authentication DB Down Error page (system_ldap_error.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERREX_PASSWORD_B, EX_PASSWORD_A, EX_PASSWORD_A_ERR
EXR_SYSERR	Description	Used to respond to a login, access, or password change request from the client with a system error.
	Displayed page	System Error page (system_error.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR, EX_ACCESS_B, EX_ACCESS_A, EX_ACCESS_A_ERR, EX_PASSWORD_B, EX_PASSWORD_A, EX_PASSWORD_A_ERR
EXR_LOGINLMT 10.0	Description	Used to treat a login request from the client as a login limit error.
	Displayed page	Login Limit Error page (login_limit_error.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR

Control code	Description	
EXR_ACLREQ 10.0	Description	Used to declare a no request permission error.
	Displayed page	No Request Permission Error (request_acl_error.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR, EX_ACCESS_B, EX_ACCESS_A, EX_ACCESS_A_ERR, EX_PASSWORD_B, EX_PASSWORD_A, EX_PASSWORD_A_ERR
EXR_SIDERR 10.0	Description	Used to declare a session ID generation error.
	Displayed page	System Error page (system_error.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR
EXR_DBBUSY 10.0	Description	Used to declare a DB busy error.
	Displayed page	DB Busy Error page (system_busy_database.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR, E X_PASSWD_B, EX_PASSWD_A, EX_PASSWD_A_ERR
EXR_PWDMINLEN 10.0	Description	Used to declare a minimum password length error.
	Displayed page	Password Policy Error page (pwdchg_policy_error.html)
	Available location	EX_PASSWD_B, EX_PASSWD_A, EX_PASSWD_A_ERR
EXR_PWDMAXLEN 10.0	Description	Used to declare a maximum password length error.
	Displayed page	Password Policy Error page (pwdchg_policy_error.html)
	Available location	EX_PASSWD_B, EX_PASSWD_A, EX_PASSWD_A_ERR
EXR_PWDALPHANUM 10.0	Description	Used to declare a password character type error.
	Displayed page	Password Policy Error page (pwdchg_policy_error.html)
	Available location	EX_PASSWD_B, EX_PASSWD_A, EX_PASSWD_A_ERR
EXR_PWDSAMEPASS 10.0	Description	Used to declare a same password as user ID error.
	Displayed page	Password Policy Error page (pwdchg_policy_error.html)
	Available location	EX_PASSWD_B, EX_PASSWD_A, EX_PASSWD_A_ERR

Control code	Description	
EXR_USREX1	Description	Used to respond to a login, access, or password change request from the client with a user-defined error 1.
	Displayed page	User-Defined Error page 1 (usr_ext1.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR, EX_ACCESS_B, EX_ACCESS_A, EX_ACCESS_A_ERR, EX_PASSWORD_B, EX_PASSWORD_A, EX_PASSWORD_A_ERR
EXR_USREX2	Description	Used to respond to a login, access, or password change request from the client with a user-defined error 2.
	Displayed page	User-Defined Error page 2 (usr_ext2.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR, EX_ACCESS_B, EX_ACCESS_A, EX_ACCESS_A_ERR, EX_PASSWORD_B, EX_PASSWORD_A, EX_PASSWORD_A_ERR
EXR_USREX3	Description	Used to respond to a login, access, or password change request from the client with a user-defined error 3.
	Displayed page	User-Defined Error page 3 (usr_ext3.html)
	Available location	EX_LOGIN_B, EX_LOGIN_A, EX_LOGIN_A_ERR, E X_ACCESS_B, EX_ACCESS_A, EX_ACCESS_A_ERR, EX_PASSWORD_B, EX_PASSWORD_A, EX_PASSWO RD_A_ERR

*1 If the IceWall SSO Client Certification Option is not installed, EXR_SYSERR is used.

Note that if you use a control code which cannot be used for an entry, the value is changed as follows.

Entry	Changed to	Displayed page
Before login After login Before access control After access control	EXR_SYSERR	System Error page (system_error.html)
Before password change After password change	EXR_SYSERR	Password Change Error page (pwdchg_error.html)

Entry	Changed to	Displayed page
After login error After access control error Before logging out After logging out After logout error After password change error	EXR_OK	The page displayed depends on the operation of the Authentication Module and Forwarder.

4.2.1 APIs in the interface function

The APIs available in the interface function are listed below.

API name	Description
IW_ExCSNoSelectDBClm	Sets a value to a column of the Authentication DB in advance.
IW_ExCSGetCacheUser	Acquires the user information from the cache.
IW_ExCSModifyCacheUser (10.0)	Changes the user information in the cache.
IW_ExCSRleaseRecord	IW_ExCSGetCacheUser, IW_ExCSModify Release the user information used by CacheUser.
IW_ExCSGetICP	Acquires elements from a request sent by using ICP 2.0 and a response received for it.
IW_ExCSAddICP	Adds elements to a request sent by using ICP 2.0 and a response received for it.
IW_ExCSModifyICP	Changes elements in a request sent by using ICP 2.0 and a response received for it.
IW_ExCSDeleteICP	Deletes elements from a request sent by using ICP 2.0 and a response received for it.
IW_ExCSAddCacheUserGroup	Add an arbitrary group name for group information of the user running the UserExit entry.
IW_ExCSDeleteCacheUserGroup	Deletes an arbitrary group name from group information of the user running the UserExit entry.
IW_ExCSGetLoginUserCount [10.0]	Acquires the number of login users.

The following describes the specifications of these APIs.

IW_ExCSNoSelectDBClm

Format

int IW_ExCSNoSelectDBClm (EX_CS_REQ* req, int kind, char* column, char* value)

Function

Sets the user information in advance to prevent the information from being read from the Authentication DB upon user authentication.

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

column : Specify the name of the column that is not read from the Authentication DB. Note that the User ID column cannot be specified.

value : Specify a value for the column that is not read from the Authentication DB. Specify numeric data as a string.

Return value

One of the following values is returned.

EXCS_SUCCESS	: Normal termination
EXCS_ERROR	: Abnormal termination
EXCS_BADKIND	: The entry cannot be used by the API, or the kind value is invalid
EXCS_BADCOLUMN	: The specified column does not exist
EXCS_BADUID	: The user ID column is specified

Restrictions

Available only when the value of the kind argument of the interface function is as follows:

EX_LOGIN_B: Before user authentication

IW_ExCSGetCacheUser

Format

EX_CS_RECORD *IW_ExCSGetCacheUser (EX_CS_REQ*req, int kind, char* sessionid, char* column)

Function

Acquires the user information of the login user from the internal cache.

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

sessionid : Specify the session ID of the login user to acquire the user information. Set the value of the sessionid argument passed from the interface function.

column : Specify the name of the column to acquire the value. In addition to an Authentication DB column, you can also acquire information stored in the internal cache. Specify one of the following reserved words as the column name.

EXCS_API_LOGINAUTH	: Authentication method (by user ID: 1, by client certificate: 2)
EXCS_API_USERID	: User ID (If the user ID is changed at EX_LOGIN_B, the changed ID is acquired.)
EXCS_API_PASSWORD	: Password (If the password is changed at EX_LOGIN_B, the changed password is acquired.)
EXCS_API_PASSWORDFLG	: Password expiration flag (not expired: 0, expired: 1)
EXCS_API_SESSIONTIME	: Login expiration date/time (YYYYMMDDhhmmss)
EXCS_API_LOGINTIME	: Last login date/time (YYYYMMDDhhmmss)
EXCS_API_GROUP	: Group information (When belonging to multiple groups, separate them with commas.)
EXCS_API_SOURCEADDR	: Request source IP address
EXCS_API_ICPVER	: ICP version number

[Arbitrary name] : Environment information of the request source sent upon login using ICP 2.0

Return value

One of the following values is returned.
Other than NULL: Pointer to record structure
NULL : Abnormal termination

The record structure has several member variables. It is composed as follows:

```
EX_CS_RECORD {
    int  result;
    char *name;
    char *value;
}
```

The result variable contains the processing result of the API.
The name variable stores the column name specified by the column argument.
The value variable stores the value for the column.

When a pointer to the record structure has been returned, the processing result is stored in the result variable. One of the following is stored:

EXCS_SUCCESS : Value obtained successfully
EXCS_ERROR : System error
EXCS_BADKIND : The entry cannot be used by the API, or the kind value is invalid
EXCS_BADCOLUMN : The specified column name is invalid
EXCS_BADSID : A user who is not logged in is specified

Restrictions

Available only when the value of the kind argument of the interface function is one of the following:

EX_LOGIN_A : After user authentication
EX_ACCESS_B : Before access permission check
EX_ACCESS_A : After access permission check
EX_ACCESS_A_ERR : After access permission check error
EX_LOGOUT_B : Before logout
EX_LOGOUT_A : After logout
EX_LOGOUT_A_ERR : After logout error
EX_PASSWD_B : Before password change
EX_PASSWD_A : After password change
EX_PASSWD_A_ERR : After a password change error

After using this API, destroy the user information from the structure acquired as the return value using the IW_ExCSReleaseRecord() API.

IW_ExCSModifyCacheUser 10.0

Format

EX_CS_RECORD* IW_ExCSModifyCacheUser (EX_CS_REQ* req, int kind, char* sessionid, char* column, char* value)

Function

Changes the user information of the login user from the internal cache. After changing the user information, the information of the group to which the user belongs is rebuilt.

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

sessionid : Specify the session ID of the login user to change the user information. Set the value of the sessionid argument passed from the interface function.

column : Specify the name of the column to change the value. In addition to an Authentication DB column, you can also change information stored in the internal cache. Specify one of the following reserved words as the column name. Note that some of the internal information that can be referenced cannot be changed.

EXCS_API_PASSWORDFLG	: Password expiration flag
EXCS_API_SESSIONTIME	: Login expiration date/time
EXCS_API_LOGINTIME	: Last login date/time
[Arbitrary name]	: Environment information of request source sent upon login using ICP 2.0

value : Specify a new value. Specify numeric data as a string.

Return value

One of the following values is returned.

Other than NULL	: Pointer to the record structure
NULL	: Abnormal termination

When a pointer to the record structure has been returned, the processing result is stored in the result variable. One of the following is stored:

EXCS_SUCCESS	: Value obtained successfully
EXCS_ERROR	: System error

EXCS_BADKIND : The entry cannot be used by the API, or the kind value is invalid
EXCS_BADCOLUMN : The specified column name is invalid
EXCS_BADSID : A user who is not logged in is specified
EXCS_BADGROUP : An error occurred during rebuilding of the target group information 10.0

Note that the member variable "name" of the record structure stores the column name specified as an argument to the API and "value" stores the value before change.

Restrictions

Available only when the value of the kind argument of the interface function is one of the following:

EX_LOGIN_A : After user authentication
EX_ACCESS_B : Before access permission check
EX_ACCESS_A : After access permission check
EX_ACCESS_A_ERR : After access permission check error
EX_LOGOUT_B : Before logout
EX_LOGOUT_A : After logout
EX_LOGOUT_A_ERR : After logout error
EX_PASSWD_B : Before password change
EX_PASSWD_A : After password change
EX_PASSWD_A_ERR : After a password change error

After using this API, destroy the user information from the structure acquired as the return value using the IW_ExCSReleaseRecord() API.

IW_ExCSReleaseRecord

Format **void IW_ExCSReleaseRecord (EX_CS_RECORD* rec)**

Function Releases the record structure.

Argument **rec** :Specify the record structure. Set the pointer acquired as the return value from IW_ExCSGetCacheUser() or IW_ExCSModifyCacheUser().

Return value None.

Restrictions The operation is not guaranteed if you pass data other than a pointer to the record structure.

IW_ExCSGetICP

Format

EX_CS_RECORD* IW_ExCSGetICP (EX_CS_REQ* req, int type, char* name)

Function

Acquires elements from a request message sent from the source using ICP 2.0 and a response message to it.

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

type : This flag specifies the storage area within the edit target message. Set one of the following values:

EX_TYPE_EXTRA : Acquire from the additional information
EX_TYPE_HEADER : Acquire from the header information
EX_TYPE_SUPLE : Acquire from the supplemental information

name : Specify the name of the element to acquire from the message.

Return value

One of the following values is returned.

Other than NULL: Pointer to the record structure
NULL : Abnormal termination

When a pointer to the record structure has been returned, the processing result is stored in the result variable. One of the following is stored:

EXCS_SUCCESS : Value acquired successfully
EXCS_ERROR : System error
EXCS_BADICP : Request of a different version is specified
EXCS_BADNAME : The specified element name does not exist in the message

Note that the member variable "name" of the record structure stores the column name specified as an argument to the API and "value" stores the value before change.

Restrictions

For details of the storage area, see the “IceWall Cert Protocol 2.0 Developer's Manual.”

This API is dedicated to ICP 2.0. Using it with ICP 1.0 causes an error.

After using this API, destroy the user information from the structure acquired as the return value using the IW_ExCSReleaseRecord() API.

IW_ExCSAddICP

Format

int IW_ExCSAddICP (EX_CS_REQ* req, int type, char* name, char* value)

Function

Adds elements to a request message sent from the source using ICP 2.0 and a response message to it.

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

type : This flag specifies the storage area within the edit target message. Set one of the following values:

EX_TYPE_EXTRA : Add to the additional information
EX_TYPE_HEADER : Add to the header information
EX_TYPE_SUPLE : Add to the supplemental information

name : Specify the name of the element to add to the message.

value : Specify the value of the element to add to the message. Specify numeric data as a string.

Return value

One of the following values is returned.

EXCS_SUCCESS : Value obtained successfully
EXCS_ERROR : System error

Restrictions

This API is dedicated to ICP 2.0. Using it with ICP 1.0 causes an error.

IW_ExCSModifyICP

Format

EX_CS_RECORD* IW_ExCSModifyICP (EX_CS_REQ* req, int type, char* name, char* value)

Function

Changes elements of a request message sent from the source using ICP 2.0 and a response message to it.

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

type : This flag specifies the storage area within the edit target message. Set one of the following values:

EX_TYPE_EXTRA : Change the elements in additional information
EX_TYPE_HEADER : Change the elements in header information
EX_TYPE_SUPLE : Change the elements in supplemental information

name : Specify the name of the element in the message to change the value.

value : Specify the new value of the element in the message. Specify numeric data as a string.

Return value

One of the following values is returned.

Other than NULL: Pointer to the record structure
NULL : Abnormal termination

When a pointer to the record structure has been returned, the processing result is stored in the result variable. One of the following is stored:

EXCS_SUCCESS : Value obtained successfully
EXCS_ERROR : System error
EXCS_BADICP : Request of a different version is specified
EXCS_BADNAME : The specified element name does not exist in the message

Note that the member variable "name" of the record structure stores the column name specified as an argument to the API and "value" stores the value before change.

Restrictions

For details of the storage area, see the “IceWall Cert Protocol 2.0 Developer's Manual.”

After using this API, destroy the user information from the structure acquired as the return value using the IW_ExCSReleaseRecord() API.

This API is dedicated to ICP 2.0. Using it with ICP 1.0 causes an error.

IW_ExCSDeleteICP

Format

EX_CS_RECORD* IW_ExCSDeleteICP (EX_CS_REQ* req, int type, char* name)

Function

Deletes elements from a request message sent from the source using ICP 2.0 and a response message to it.

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

type : This flag specifies the storage area within the edit target message. Set one of the following values:
EX_TYPE_EXTRA : Delete the elements from additional information
EX_TYPE_HEADER : Delete the elements from header information
EX_TYPE_SUPLE : Delete the elements from supplemental information

name : Specify the name of the element in the message to change the value.

Return value

One of the following values is returned.
Other than NULL: Pointer to record structure
NULL : Abnormal termination

When a pointer to the record structure has been returned, the processing result is stored in the result variable. One of the following is stored:
EXCS_SUCCESS : Value obtained successfully
EXCS_ERROR : System error
EXCS_BADICP : Request of a different version is specified
EXCS_BADNAME : The specified element name does not exist in the message

Note that the member variable "name" of the record structure stores the column name specified as an argument to the API and "value" stores the value before change.

Restrictions

For details of the storage area, see the “IceWall Cert Protocol 2.0 Developer's Manual.”

After using this API, destroy the user information from the structure acquired as the return value using the IW_ExCSReleaseRecord() API.

This API is dedicated to ICP 2.0. Using it with ICP 1.0 causes an error.

IW_ExCSAddCacheUserGroup 10.0

Format

int IW_ExCSAddCacheUserGroup (EX_CS_REQ *req, int kind,char *sessionid, char *groupname)

Function

Add an arbitrary group name for group information of the user running the UserExit entry.

This API is available with Version 8.0 or later.

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

sessionid : Specify the session ID of the login user to obtain the user information. Set the value of the sessionid argument passed from the interface function.

groupname : Specify a group name to add. The group name is not case-sensitive. Specify a group name to which the target user does not belong. You can also specify a group not defined in the group configuration file (cert.grp) or a group for which the target user does not match the belonging condition.

Return value

One of the following values is returned.

EXCS_SUCCESS : Group information added successfully
EXCS_ERROR : System error
EXCS_BADKIND : The entry cannot be used by the API, or the kind value is invalid
EXCS_BADSID : The session ID of a user who is not logged in is specified
EXCS_BADNAME : The specified group name is invalid

Restrictions

Available only when the the kind argument value is one of the following:

EX_LOGIN_A : After user authentication
EX_ACCESS_B : Before access permission check
EX_ACCESS_A : After access permission check
EX_PASSWD_B : Before password change

EX_PASSWD_A : After password change

Running IW_ExCSModifyCacheUser() after running this API invalidates the group information added with this API. Therefore, pay attention to the order of running APIs.

IW_ExCSDeleteCacheUserGroup

Format

int IW_ExCSDeleteCacheUserGroup (EX_CS_REQ *req, int kind,char *sessionid, char *groupname)

Function

Deletes an arbitrary group name from group information of the user running the UserExit entry.

This API is available with Version 8.0 or later.

Argument

req : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function.

kind : Uses the UserExit routine location flag. Set the value of the kind argument passed from the interface function.

sessionid : Specify the session ID of the login user to obtain the user information. Set the value of the sessionid argument passed from the interface function.

groupname : Specify a group name to delete. The group name is not case-sensitive. Specify a group name to which the target user does not belong.

Return value

One of the following values is returned.

EXCS_SUCCESS	: Group information deleted successfully
EXCS_BADKIND	: The entry cannot be used by the API, or the kind value is invalid
EXCS_BADSID	: The session ID of a user who is not logged in is specified
EXCS_BADNAME	: The specified group name is invalid
EXCS_BADGROUP	: The user belongs only to the group to be deleted

Restrictions

Available only when the value of kind argument of the interface function is one of the following:

EX_LOGIN_A	: After user authentication
EX_ACCESS_B	: Before access permission check
EX_ACCESS_A	: After access permission check
EX_PASSWD_B	: Before password change
EX_PASSWD_A	: After password change

Running IW_ExCSModifyCacheUser() after running this API invalidates the group information added with this API. Therefore, pay attention to the order of running APIs.

IW_ExCSGetLoginUserCount 10.0

Format **int IW_ExCSGetLoginUserCount(EX_CS_REQ *req)**

Function Obtains the number of login users.

This API is available with Version 10.0 or later.

Argument **req** : Uses the UserExit routine request structure. Set the value of the req argument passed from the interface function. The API terminates abnormally if the req argument value is NULL or invalid.

Return value One of the following values is returned.
Number of login users (0 or higher): Normal termination
EXCS_ERROR : Abnormal termination

Restrictions This API can be used with all entries.

5 Remarks

Pay attention to the following when developing and integrating the UserExit routine.

5.1 Scope of UserExit routine

Because the process executed within the UserExit routine is closely related to the internal processing of the Forwarder or Authentication Module, when a system error occurs within the UserExit routine, the module may abort. Be sure to thoroughly test the developed library before integrating it into the module.

Do not call exit() from the UserExit routine because doing so terminates the running process of the module.

5.2 Performance degrading

When a new process is added to the UserExit routine, the performance of the entire process is degraded because the new process is executed in addition to the normal processing. Keep the process as simple as possible.

5.3 Memory leak

Be sure to release the memory area reserved by the UserExit routine. Failure to do so may cause memory leak.

5.4 Notes on upgrading

When the previous version was configured to generate 64-byte session IDs, the UserExit routine must also be able to support 64-byte session IDs. 10.0

5.5 Support

Support for the product is available only when the standard is used. Support is not available if a custom UserExit routine is used. Be sure to back up the standard library because it is necessary when you request for support.

6 Restrictions

Note the following restrictions when developing the UserExit routine:
If you do not observe them during development, each process may become unable to run or unstable.

6.1 Restrictions common to all modules

The following restrictions are common to the Forwarder and Authentication Module.

6.1.1 Standard library version to link

The Forwarder and Authentication Module is designed to operate using libraries dynamically.
However, if an archive library is linked with the UserExit routine, the library linked with the UserExit routine is used and a mismatch due to different library versions may occur. (A trouble which cannot be resolved by applying patches to the OS may occur.)

Note) In particular, you should use extra care about the state of the linked library when using a library of which organization (i.e., organization of libclntsh.sl) differs depending on the installation environment (e.g., Oracle) is used within the UserExit routine.

6.1.2 Compatibility

The UserExit routine is not compatible with an upper or lower version of the IceWall and must be rebuilt when using the routine with a version other than the one used to create it. Therefore, when you have upgraded IceWall SSO, be sure to rebuild the UserExit routine.

6.2 Restriction on Forwarder

The following restrictions are related to Forwarder.

6.2.1 Restrictions common to all OSs

Build the code as 64-bit binary.

6.2.2 HP-UX edition (Itanium)

Be sure to add "+DD64" to the compile option.
Build the UserExit routine using the complier with the same architecture as that of the processor that runs the Forwarder to integrate the UserExit routine with.

6.2.3 Linux edition

Be sure to add "-m64" to the compile option.

6.3 Restriction on Authentication Module

The following restrictions are related to the Authentication Module.

6.3.1 Restrictions common to all OSes

Build the code as 64-bit binary.
The Authentication Module uses threads for the internal processing and the UserExit routine is called from a thread. Therefore, be sure to use the thread-safe versions of the standard library functions. Also, when creating a user function, be sure to create it as a thread-safe version.

6.3.2 HP-UX 11i v3 edition (Itanium)

Be sure to add "+DD64" to the compile option.
Build the UserExit routine using the complier with the same architecture as that of the processor that runs the Authentication Module to integrate the UserExit routine with. Link 64-bit libraries only.

6.3.3 Linux edition

Be sure to add "-m64" to the compile option.
Both the "Linux Threads" and "Native POSIX Library Thread" threads are supported for the Authentication Module.

7 Tips on Developing UserExit Routine

This section introduces a tip for developing the UserExit routine.

7.1 UserExit routine for the Forwarder

7.1.1 Request judgment method

Whether a request from a client is intended for the IceWall or Backend Web Server is judged by the method and POST data contents.
The judgment method is described below.

(1) Check the value of the kind argument of the IW_ExDFWInterFace().
(2) If the value of the kind argument is EX_DFW_KIND_REQ, use the IW_ExDFWGetMethod() function to obtain the method.
(3) If the method is POST, use the IW_ExDFWGetPostdata() function to obtain the POST data.
(4) If the POST data contains the value defined in the POSTKEY_LOGIN parameter in the Forwarder configuration file (dfw.conf), login is requested. If it contains the POSTKEY_LOGOUT parameter value, logout is requested. And if it contains the POSTKEY_PWDCHG parameter value, password change is requested.

7.1.2 How to control request to the Backend Web Server

You can change a value of the HTTP header included in a request sent to the Backend Web Server or add a new HTTP header.
The method is described below.

(1) Check the value of the kind argument of the IW_ExDFWInterFace().
(2) If the value of the kind argument is EX_DFW_KIND_SEND, use the IW_ExDFWGetBuffer() function to obtain the request data buffer.
(3) Copy the contents of the request data buffer to the work buffer. You need to allocate the work buffer in advance.
(4) Modify the HTTP header data in the work buffer as necessary.
(5) Use the IW_ExDFWModifyBuffer() function to set the contents of the work buffer as the request data. Use the IW_ExDFWModifyBufferLength() function if the POST data sent to the Backend Web Server is binary.

7.1.3 How to control response from the Backend Web Server

You can change a value of the HTTP header included in a response received from the Backend Web Server or add a new HTTP header.
The method is described below.

(1) Check the value of the kind argument of the IW_ExDFWInterFace().
(2) If the value of the kind argument is EX_DFW_KIND_RECV, use the IW_ExDFWGetBuffer() function to obtain the response data buffer.
(3) Copy the contents of the response data buffer to the work buffer. You need to allocate the work buffer in advance.
(4) Modify the HTTP header data in the work buffer as necessary.

(5) Use the IW_ExDFWModifyBuffer() function to set the contents of the work buffer as the response data. Use the IW_ExDFWModifyBufferLength() function if the contents received from the Backend Web Server is binary.

7.2 UserExit routine for the Authentication Module

7.2.1 Error judgment after each process

In the UserExit routine called after error occurrence in a process , determine which error occurred according to the argument.

The judgment method is described below. (This example assumes a login error.)

(1) Check the value of the kind argument of the IW_ExInterFace().
(2) If the value of the kind argument is EX_LOGIN_A_ERR, look for a comma contained in the data argument of the IW_ExInterFace() function.
(3) Obtain the string following the comma and convert it into a numeric value.
(4) Compare the converted value with the control code to determine the error.

8 Sample Source Code

The following gives the sample source of each UserExit routine.

8.1 Sample UserExit routine for the Forwarder

8.1.1 Changing the GET method to the POST method

```
int IW_ExDFWInterFace( req, kind )
EX_DFW_REQ *req;
int       kind;
{

   char postdata[128];

   if ( kind == EX_DFW_KIND_REQ ) {
      if ( strcmp( IW_ExDFWGetMethod( req, kind ),  "GET" ) ) {
         IW_ExDFWModifyMethod( req, kind, EX_DFW_METHOD_POST );
         strcpy( postdata,  "Name1=Value1&Name2=Value2" );
         IW_ExDFWModifyPostdata( req, kind, postdata );
      }
   }

   return( EX_DFW_OK );

}
```

8.1.2 Changing the HTTP header in a request

```
int IW_ExDFWInterFace( req, kind )
EX_DFW_REQ *req;
int       kind;
{

   char *req1, *req2;

   if ( kind == EX_DFW_KIND_SEND ) {
      req1 = IW_ExDFWGetBuffer( req, kind );
      req2 = (char*)malloc( strlen( req1 ) + 1 );
      strcpy( req2, req1 );
      HttpHeaderModify( req2,  "User-Agent" ,  "MyBrowser2.1" ); ※ 1
      IW_ExDFWModifyBuffer( req, kind, req2 );
      free( req2 );
   }

   return( EX_DFW_OK );

}
```

∗1 The HttpHeaderModify() function is not provided as an API.
It is a virtual function to change the values of the HTTP header.

8.2 Sample UserExit routine for the Authentication Module

8.2.1 Forcibly changing a particular user ID to guest

```
int  IW_ExInterFace( kind, userid, password, sessionid, data, req )
int  kind;
char *userid;
char *password;
char *sessionid;
char *data;
EX_CS_REQ *req;
{

   if ( kind == EX_LOGIN_B ) {
      if ( strcmp( userid,  "user01" ) ) {
         strcpy( userid,  "guest" );
         strcpy( password,  "guest" );
      }
   }

   return( EXR_OK );

}
```

8.2.2 Changing the request for a particular URL to a login request

```
int  IW_ExInterFace( kind, userid, password, sessionid, data, req )
int  kind;
char *userid;
char *password;
char *sessionid;
char *data;
EX_CS_REQ *req;
{

   if ( kind == EX_ACCESS_B ) {
      if ( strcmp( data,  "http://www.hp.com/members" ) ) {
         return( EXR_RLOGIN );
      }
   }

   return( EXR_OK );

}
```

9 Reference

The following shows the contents of the the standard UserExit routine development kit installed by default.
The development kit consists of the following directories and files.

Directories and files					Description
/opt/icewall-sso	/developkit	/dfw	/DfwExit	/dfwinterface.c	For the Forwarder UserExit Routine Development Kit
				/dfwinterface.h	
				/Makefile	
		/certd	/CertExit	/iwintface.c	For the Authentication Module UserExit Routine Development Kit
				/iwintface.h	
				/Makefile	

Note that the name of the library created using this development kit is the same as that of the library installed by default. Be sure to back up the standard library before creating and integrating a new library.

9.1 UserExit Routine Development Kit for Forwarder

9.1.1 Skeleton source (dfwinterface.c)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "dfwinterface.h"

int IW_ExDFWInterFace( req, kind )
EX_DFW_REQ *req;
int        kind;
{

    switch( kind ) {
        case EX_DFW_KIND_REQ: break;
        case EX_DFW_KIND_SEND: break;
        case EX_DFW_KIND_RECV: break;
        case EX_DFW_KIND_END: break;
        case EX_DFW_KIND_LOGIN_SEND: break;
        case EX_DFW_KIND_LOGIN_RECV: break;
        case EX_DFW_KIND_LOGOUT_SEND: break;
        case EX_DFW_KIND_LOGOUT_RECV: break;
        case EX_DFW_KIND_ACC_SEND: break;
        case EX_DFW_KIND_ACC_RECV: break;
        case EX_DFW_KIND_PWDCHG_SEND: break;
        case EX_DFW_KIND_PWDCHG_RECV: break;
        case EX_DFW_KIND_ALOGIN_SEND: break;
        case EX_DFW_KIND_ALOGIN_RECV: break;
    }

    return( EX_DFW_OK );
```

```
}
```

9.1.2 Header file (dfwinterface.h)

```
#ifndef DFWINTFACE_H
#define DFWINTFACE_H

/* Kind */
#define EX_DFW_KIND_REQ             0       /* After obtaining request information          */
#define EX_DFW_KIND_SEND            1       /* Before sending data                          */
#define EX_DFW_KIND_RECV            2       /* After receiving data                         */
#define EX_DFW_KIND_END             3       /* Before sending contents (6.0SP1)             */
#define EX_DFW_KIND_LOGIN_SEND      4       /* Before sending login request (8.0R3)         */
#define EX_DFW_KIND_LOGIN_RECV      5       /* After receiving login response (8.0R3)       */
#define EX_DFW_KIND_LOGOUT_SEND     6       /* Before sending logout request (8.0R3)        */
#define EX_DFW_KIND_LOGOUT_RECV     7       /* After receiving logout response (8.0R3)      */
#define EX_DFW_KIND_ACC_SEND        8       /* Before sending access request (8.0R3)        */
#define EX_DFW_KIND_ACC_RECV        9       /* After receiving access response (8.0R3)      */
#define EX_DFW_KIND_PWDCHG_SEND     10      /* Before sending password change request
(8.0R3)                                     */
#define EX_DFW_KIND_PWDCHG_RECV     11      /* After receiving password change response
(8.0R3)                                     */
#define EX_DFW_KIND_ALOGIN_SEND     12      /* Before sending agent login request (8.0R3)*/
#define EX_DFW_KIND_ALOGIN_RECV     13      /* After receiving agent login response (8.0R3)
*/
/* API Return */
#define EX_DFW_ERROR                -1      /* Abnormal termination                         */
#define EX_DFW_SUCCESS              0       /* Normal termination                           */
#define EX_DFW_BADKIND              1       /* Invalid kind                                 */
#define EX_DFW_NOTFOUND             2       /* No control target (8.0R3)                    */
/* API Switch */
#define EX_DFW_HTTP                 0       /* HTTP operation                               */
#define EX_DFW_HTTPS                1       /* HTTPS operation                              */
#define EX_DFW_METHOD_GET           0       /* GET method                                   */
#define EX_DFW_METHOD_POST          1       /* POST method                                  */
#define EX_DFW_UID_LOGIN            0       /* Login user ID (7.0SP2)                       */
#define EX_DFW_UID_INPUT            1       /* Input user ID (7.0SP2)                       */
#define EX_DFW_ICP_HEADER           0       /* Header section (8.0R3)                       */
#define EX_DFW_ICP_BODY             1       /* Body section (8.0R3)                         */
#define EX_DFW_ICP_DEL              0       /* Deletion process (8.0R3)                     */
#define EX_DFW_ICP_ADD              1       /* Addition process (8.0R3)                     */
#define EX_DFW_ICP_MOD              2       /* Change process (8.0R3)                       */
/* Return */
#define EX_DFW_OK                   0       /* No page (normal termination)                 */
#define EX_DFW_LOGIN                10      /* Login page                                   */
#define EX_DFW_LOGINCERT            17      /* Certificate login page                       */
#define EX_DFW_LOGINFORCE           6       /* Forced login page                            */
#define EX_DFW_LOGINUIDERR          1       /* User ID error page                           */
#define EX_DFW_LOGINPWDERR          18      /* Login password error page                    */
#define EX_DFW_LOGINLOCKERR         4       /* Account lock error page                      */
#define EX_DFW_LOGINCERTERR         7       /* Pre-authenticated error error page           */
#define EX_DFW_LOGINSERIALERR       8       /* Serial No. error page                        */
#define EX_DFW_LOGINGRPERR          9       /* No group error page                          */
#define EX_DFW_LOGINSTOP            5       /* Login stop error page                        */
#define EX_DFW_LOGINLIMITERR        19      /* Login limit error page                       */
#define EX_DFW_LOGINTIMEERR         46      /* Login send timeout page                      */
```

```
#define EX_DFW_LOGOUT           20      /* Logout page                          */
#define EX_DFW_LOGOUTOK         21      /* Logout successful page               */
#define EX_DFW_LOGOUTNO         22      /* Logout failed page                   */
#define EX_DFW_LOGOUTERR        23      /* Logout error page                    */
#define EX_DFW_ACCESS           11      /* Access permission error page         */
#define EX_DFW_SENDERR          24      /* Post send error page                 */
#define EX_DFW_REQACLERR        50      /* Request ACL error page (10.0)        */
#define EX_DFW_POSTERR          45      /* POST data maximum value error page
(8.0R3)                         */
#define EX_DFW_PWDCHG           3       /* Password change page                 */
#define EX_DFW_PWDCHGOK         25      /* Password change successful page      */
#define EX_DFW_PWDOLDERR        26      /* Old password error page              */
#define EX_DFW_PWDREERR         27      /* New password change error page       */
#define EX_DFW_PWDPCYERR        28      /* Password change policy error page    */
#define EX_DFW_PWDVIOERR        29      /* No password change permission page   */
#define EX_DFW_PWDNOLOGIN       30      /* Password change no login error page  */
#define EX_DFW_PWDERR           2       /* Password change error page           */
#define EX_DFW_PWDWARNING       44      /* Password expiration warning page     */
#define EX_DFW_PWDTIMEERR       47      /* Password change send timeout page    */
#define EX_DFW_SYSERR           13      /* System error page                    */
#define EX_DFW_ALIASNO          31      /* No alias error page                  */
#define EX_DFW_ALIASBAD         32      /* Undefined alias error page           */
#define EX_DFW_DOWNCERTD        33      /* Certd down error page                */
#define EX_DFW_DOWNDB           12      /* DB down error page                   */
#define EX_DFW_DOWNBKEND        34      /* Backend down error page              */
#define EX_DFW_TOUTCERTD        35      /* Certd timeout page                   */
#define EX_DFW_TOUTBKEND        36      /* Backend timeout page                 */
#define EX_DFW_DBBUSYERR        49      /* DB busy error page (10.0)            */
#define EX_DFW_FILTERGET        37      /* GET filter error page                */
#define EX_DFW_FILTERPOST       38      /* POST filter error page               */
#define EX_DFW_FILTERHTML       39      /* HTML filter error page               */
#define EX_DFW_FILTERHOST       40      /* Host filter error page               */
#define EX_DFW_FILTERREQ        48      /* Request filter error page (10.0)     */
#define EX_DFW_USREX1           14      /* User-defined error 1 page            */
#define EX_DFW_USREX2           15      /* User-defined error 2 page            */
#define EX_DFW_USREX3           16      /* User-defined error 3 page            */
#define EX_DFW_USREX4           41      /* User-defined error 4 page            */
#define EX_DFW_USREX5           42      /* User-defined error 5 page            */
#define EX_DFW_USREX6           43      /* User-defined error 6 page            */

typedef struct ex_dfw_req EX_DFW_REQ;

extern int  IW_ExDFWInterFace( EX_DFW_REQ *req, int kind );
extern int  IW_ExDFWChangeProtocol( EX_DFW_REQ *ex_rec, int kind, int flg );
extern char *IW_ExDFWGetMethod( EX_DFW_REQ *ex_rec, int kind );
extern int  IW_ExDFWModifyMethod( EX_DFW_REQ *ex_rec, int kind, int method );
extern char *IW_ExDFWGetPostdata( EX_DFW_REQ *ex_rec, int kind );
extern int  IW_ExDFWModifyPostdata( EX_DFW_REQ *ex_rec, int kind, char *buf );
extern char *IW_ExDFWGetBuffer( EX_DFW_REQ *ex_rec, int kind );
extern int  IW_ExDFWModifyBuffer( EX_DFW_REQ *ex_rec, int kind, char *buf );
extern int  IW_ExDFWModifyBufferLength( EX_DFW_REQ *ex_rec, int kind, char *buf, int len );
extern int  IW_ExDFWModifyRequestPath( EX_DFW_REQ *ex_rec, int kind, char *path );
extern int  IW_ExDFWModifyPathinfo( EX_DFW_REQ *ex_rec, int kind, char *path );
extern char *IW_ExDFWGetUid( EX_DFW_REQ *ex_rec, int kind, int flg );
extern char *IW_ExDFWGetCertData( EX_DFW_REQ *ex_rec, int kind, char *name, int target );
extern int  IW_ExDFWCtrlCertData( EX_DFW_REQ *ex_rec, int kind
    ,char *name, char *value, int target, int mode );

#endif /* #ifndef DFWINTFACE_H */
```

9.1.3 Makefile (HP-UX Itanium 64-bit edition: Makefile)

```
LD=cc

LDFLAGS=+z +DD64 +noobjdebug -Ae

LIBS=

INCLUDE=-I./

MAKEFILE=Makefile

OBJS=dfwinterface.o

PROGRAM=DfwExit

INST_LIB=DfwExit

SRCS=dfwinterface.c

all:$(PROGRAM)

$(PROGRAM):$(OBJS)
ld -b -o lib$(PROGRAM).sl $(OBJS) $(LIBS)

.c.o:
$(LD) $(CC_CMD) $(LDFLAGS) $(INCLUDE) -c $(SRCS)

debug:
make CC_CMD="-DDEBUG -g"

clean:
rm -f $(OBJS) lib$(PROGRAM).sl core

install:
cp -p lib$(PROGRAM).sl ../lib/lib$(INST_LIB).sl

###
dfwintface.o: ./dfwinterface.h
```

9.1.4 Makefile (Linux 64-bit edition: Makefile)

```
LD              =   gcc
LDFLAGS         =   -DLinux -m64 -fPIC
LIBS            =
INCLUDE         =   -I./
MAKEFILE        =   Makefile
OBJS            =   dfwinterface.o
PROGRAM         =   DfwExit
INST_LIB        =   DfwExit
SRCS            =   dfwinterface.c
all:            $(PROGRAM)
$(PROGRAM):     $(OBJS)
                gcc -shared -o lib$(PROGRAM).sl $(OBJS) $(LIBS)
.c.o:
                $(LD) $(CC_CMD) $(LDFLAGS) $(INCLUDE) -c $(SRCS)
debug:
                make CC_CMD="-DDEBUG -g"
clean:
                rm -f $(OBJS) lib$(PROGRAM).sl core
install:
                cp -p lib$(PROGRAM).sl ../lib/lib$(INST_LIB).sl
                rm -f ../lib/lib$(INST_LIB).so
                ln -s ../lib/lib$(INST_LIB).sl ../lib/lib$(INST_LIB).so
###
dfwintface.o: ./dfwinterface.h
```

9.2 UserExit Routine Development Kit for Authentication Module

9.2.1 Skeleton source (iwintface.c)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "iwintface.h"

int IW_ExInterFace( kind, userid, password, sessionid, data, req )
int       kind;
char      *userid;
char      *password;
char      *sessionid;
char      *data;
EX_CS_REQ *req;
{

   switch( kind ) {
      case EX_LOGIN_B : break;
      case EX_LOGIN_A : break;
      case EX_LOGIN_A_ERR : break;
      case EX_ACCESS_B : break;
      case EX_ACCESS_A : break;
      case EX_ACCESS_A_ERR : break;
      case EX_LOGOUT_B : break;
      case EX_LOGOUT_A : break;
      case EX_LOGOUT_A_ERR : break;
      case EX_PASSWD_B : break;
      case EX_PASSWD_A : break;
      case EX_PASSWD_A_ERR : break;
   }

   return( EXR_OK );
}
```

9.2.2 Header file (iwintface.h)

```
/*-------------------------------------------------------------------------------------------------------------------*/
/*  UserExit Interface                                                                                               */
/*-------------------------------------------------------------------------------------------------------------------*/
#ifndef IWINTFACE_H
#define IWINTFACE_H

/*-------------------------------------------------------------------------------------------------------------------*/
/*  Define                                                                                                           */
/*-------------------------------------------------------------------------------------------------------------------*/
#define EX_LOGIN_B          1           /* Before user authentication          */
#define EX_LOGIN_A          2           /* After user authentication           */
#define EX_ACCESS_B         3           /* Before access permission check      */
#define EX_ACCESS_A         4           /* After access permission check       */
#define EX_LOGOUT_B         5           /* Before logout                       */
#define EX_LOGOUT_A         6           /* After logout                        */
#define EX_LOGIN_A_ERR      7           /* After user authentication error     */
#define EX_ACCESS_A_ERR     8           /* After access permission check error */
#define EX_LOGOUT_A_ERR     9           /* After logout error                  */
#define EX_PASSWD_B         10          /* Before password change              */
#define EX_PASSWD_A         11          /* After password change               */
#define EX_PASSWD_A_ERR     12          /* After password change error         */

#define EX_N_UID            "N_UID"     /* Authentication by user ID           */
#define EX_F_UID            "F_UID"     /* Forced authentication by user ID    */
#define EX_N_CERT           "N_CERT"    /* Authentication by certificate       */
#define EX_F_CERT           "F_CERT"    /* Forced authentication by certificate */
#define EX_N_SAML           "N_SAML"    /* Authentication by SAML              */
#define EX_F_SAML           "F_SAML"    /* Forced authentication by SAML       */
#define EX_N_FEDE           "N_FEDE"    /* Authentication by federation        */
#define EX_F_FEDE           "F_FEDE"    /* Forced authentication by federation */
#define EX_MANUAL           "MANUAL"    /* Logout by user                      */
#define EX_AUTO             "AUTO"      /* Automatic logout                    */
#define EX_FORCED           "FORCED"    /* Forced logout                       */
#define EXR_OK              0           /* Process succeeded                   */
#define EXR_UIDERR          1           /* User ID error                       */
#define EXR_PWDERR          2           /* Password error                      */
#define EXR_PWDCHG          3           /* Password change request             */
#define EXR_USRLOCK         4           /* Account lock error                  */
#define EXR_LOGINNO         5           /* Login lock error                    */
#define EXR_DLOGIN          6           /* Duplicate login error               */
#define EXR_PLOGIN          7           /* Concurrent usage error              */
#define EXR_SNOERR          8           /* Serial No. error                    */
#define EXR_GRPERR          9           /* No group error                      */
#define EXR_RLOGIN          10          /* Relogin request                     */
#define EXR_ACCERR          11          /* Access permission error             */
#define EXR_DBERR           12          /* Authentication DB error             */
#define EXR_SYSERR          13          /* System error                        */
#define EXR_USREX1          14          /* User-defined error 1                */
#define EXR_USREX2          15          /* User-defined error 2                */
#define EXR_USREX3          16          /* User-defined error 3                */
#define EXR_LOGOUTNG        17          /* Logout error                        */
#define EXR_PWDPOERR        18          /* Password policy error               */
#define EXR_PWDVIOERR       19          /* Password permission error           */
#define EXR_PWDWARN         20          /* Password expiration warning         */
#define EXR_LOGINLMT        21          /* Login limit error                   */
```

```
#define EXR_ACLREQ          22          /* Request execution permission error */
#define EXR_SIDERR          23          /* Session ID generation error        */
#define EXR_DBBUSY          24          /* DB busy error                      */
#define EXR_PWDMINLEN       25          /* Minimum password length error      */
#define EXR_PWDMAXLEN       26          /* Maximum password length error      */
#define EXR_PWDALPHANUM     27          /* Password character type error      */
#define EXR_PWDSAMEPASS     28          /* User ID used as password           */

#define EX_TYPE_EXTRA       1           /* ICP - Additional information       */
#define EX_TYPE_HEADER      2           /* ICP - Header information           */
#define EX_TYPE_SUPLE       3           /* ICP - Supplemental information     */

/*-------------------------------------------------------------------------------------------------------------------------*/
/*  API Define                                                                                                             */
/*-------------------------------------------------------------------------------------------------------------------------*/
#define EXCS_ERROR          -1          /* Abnormal termination               */
#define EXCS_SUCCESS        0           /* Normal termination                 */
#define EXCS_BADKIND        1           /* Invalid kind error                 */
#define EXCS_BADSID         2           /* No user error                      */
#define EXCS_BADCOLUMN      3           /* No column error                    */
#define EXCS_BADGROUP       4           /* No group error                     */
#define EXCS_BADREPLI       5           /* Replication error                  */
#define EXCS_BADUID         6           /* User ID unreferenced setting error */
#define EXCS_BADICP         7           /* ICP protocol error                 */
#define EXCS_BADNAME        8           /* No name error                      */

/*-------------------------------------------------------------------------------------------------------------------------*/
/* API Structure                                                                                                           */
/*-------------------------------------------------------------------------------------------------------------------------*/
typedef struct ex_cs_req EX_CS_REQ;     /* Total information structure        */

typedef struct _EX_CS_RECORD{           /* Record structure for API           */
int  result;                            /* Return value (EXCS_SUCCESS-)       */
char *name;                             /* Column name                        */
char *value;                            /* Column value (referenced value or old
column value)                   */
}EX_CS_RECORD;

/*-------------------------------------------------------------------------------------------------------------------------*/
/* API Prtotype                                                                                                            */
/*-------------------------------------------------------------------------------------------------------------------------*/
extern void IW_ExCSReleaseRecord( EX_CS_RECORD *record );
extern int  IW_ExCSNoSelectDBClm( EX_CS_REQ *req, int kind, char *column,
                                  char *value );
extern EX_CS_RECORD *IW_ExCSGetCacheUser( EX_CS_REQ *req, int kind,
                                  char *sessionid, char *column );
extern EX_CS_RECORD *IW_ExCSModifyCacheUser( EX_CS_REQ *req, int kind,
                                  char *sessionid, char *column, char *value );
extern EX_CS_RECORD *IW_ExCSGetICP( EX_CS_REQ *req, int type, char *name );
extern EX_CS_RECORD *IW_ExCSModifyICP( EX_CS_REQ *req, int type, char *name,
                                  char *value );
extern EX_CS_RECORD *IW_ExCSDeleteICP( EX_CS_REQ *req, int type, char *name );
extern int IW_ExCSAddICP( EX_CS_REQ *req, int type, char *name, char *value );
extern int IW_ExCSAddCacheUserGroup( EX_CS_REQ *req, int kind,
                                  char *sessionid, char *groupname );
extern int IW_ExCSDeleteCacheUserGroup( EX_CS_REQ *req, int kind,
                                  char *sessionid, char *groupname );
extern int IW_ExCSGetLoginUserCount( EX_CS_REQ *req );
```

```
extern EX_CS_RECORD *IW_ExCSModifyCacheUser_V2( EX_CS_REQ *req, int kind,
                                        char *sessionid, char *column,
                                        char *value);
extern int IW_ExCSAddCacheUserGroup_V2( EX_CS_REQ *req, int kind,
                                        char *sessionid, char *groupname);
extern int IW_ExCSDeleteCacheUserGroup_V2( EX_CS_REQ *req, int kind,
                                        char *sessionid, char *groupname);

#endif /* #ifndef IWINTFACE_H */
```

9.2.3 Makefile (HP-UX Itanium 64-bit edition: Makefile)

```
CC          = cc

CCFLAGS           = -D_UNIX -D_HPUX_SOURCE -D_POSIX_C_SOURCE=199506L -Aa +e -
D_FILE_OFFSET_BITS=64 -z +DD64 +z

LIBS        =

INCLUDE           = -I./

MAKEFILE      = Makefile

OBJS        = iwintface.o

PROGRAM       = CertExit

SRCS        = iwintface.c

all:         $(PROGRAM)

$(PROGRAM):     $(OBJS)
            ld -b -o lib$(PROGRAM).sl $(OBJS) $(LIBS)
.c.o:
            $(CC) $(CCFLAGS) $(INCLUDE) -c $(SRCS)

clean:;             rm -f $(OBJS) lib$(PROGRAM).sl core
```

9.2.4 Makefile (Linux 64-bit edition: Makefile)

```
CC          = gcc

CCFLAGS          = -m64 -fPIC -DLinux -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE
-D_GNU_SOURCE

LIBS        =

INCLUDE          = -I./

MAKEFILE     = Makefile

OBJS        = iwintface.o

PROGRAM      = CertExit

SRCS        = iwintface.c

all:         $(PROGRAM)

$(PROGRAM):     $(OBJS)
            gcc -shared -o lib$(PROGRAM).sl $(OBJS) $(LIBS)
.c.o:
            $(CC) $(CCFLAGS) $(INCLUDE) -c $(SRCS)

clean:;             rm -f $(OBJS) lib$(PROGRAM).sl core
```