
REFLECTION ZFE

User Guide

December 2016

© 2016 Attachmate Corporation, a Micro Focus company. All rights reserved.

No part of the documentation materials accompanying this Micro Focus software product may be reproduced, transmitted, transcribed, or translated into any language, in any form by any means, without the written permission of Micro Focus. The content of this document is protected under copyright law even if it is not distributed with software that includes an end user license agreement.

The content of this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Micro Focus. Micro Focus assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this document.

Attachmate, the Attachmate logo, and Reflection are registered trademarks of Micro Focus in the USA. All other trademarks, trade names, or company names referenced herein are used for identification only and are the property of their respective owners.

<http://www.attachmate.com> (<http://www.attachmate.com>)

Contents

| | |
|--|-----------|
| About Reflection ZFE | 5 |
| 1 Setting Installation Options | 7 |
| How to Adjust Session Timeout Values | 7 |
| How to Set Up the Terminal ID Manager for Reflection ZFE | 7 |
| How to Set Up Metering for Reflection ZFE | 8 |
| How to Start and Stop Services Automatically | 9 |
| How to Change Ports | 10 |
| How to set up Automated Single Sign-On for Mainframe | 11 |
| 2 Making Secure Connections | 13 |
| Securing the Web Browser to Reflection ZFE Session Server | 14 |
| Securing Reflection ZFE Session Server and Management Component to MSS | 15 |
| Securing Reflection ZFE Session Server to the Host | 16 |
| How to install unlimited strength policy jars | 16 |
| How to configure the keystore location on the Reflection ZFE Server | 16 |
| How to configure the keystore in MSS | 16 |
| How to configure a Reflection ZFE terminal session | 17 |
| Configuring X.509 Authentication | 18 |
| Configuring Single Sign-on through IIS | 20 |
| Enabling FIPS Level Security | 20 |
| To enable FIPS mode: | 20 |
| Enabling SSL | 21 |
| 3 Configuring a Connection | 23 |
| Connecting to the host | 23 |
| Common connection settings | 24 |
| 3270 and 5250 connection settings | 25 |
| VT connection settings | 27 |
| UTS connection settings | 28 |
| How to test Terminal ID Manager criteria | 29 |
| 4 Configuring Session Options | 31 |
| How to configure Display Settings | 31 |
| Customize host screen colors | 31 |
| Configure hotspots | 32 |
| Configure screen dimensions for VT and UTS hosts | 33 |
| Set cursor options | 33 |
| Set other display options | 34 |
| How to map keys | 34 |
| 3270, 5250, VT, and UTS Keyboard Mapping | 35 |
| How to configure macros for your users | 42 |
| How to transfer files to and from the host | 43 |
| How to specify copy and paste options | 43 |
| How to set user preference rules | 44 |

| | | |
|-----------|--|-----------|
| 5 | Using Sessions | 45 |
| | Managing your sessions | 45 |
| | Using Quick Keys | 45 |
| | Copying and Pasting | 47 |
| | Recording macros for the session | 47 |
| | Logging out | 48 |
| | | |
| 6 | Creating Macros | 49 |
| | How do I? | 49 |
| | Record a macro | 50 |
| | Edit a macro | 50 |
| | Run a macro | 51 |
| | Stop a macro | 51 |
| | Delete a macro | 51 |
| | View my macros | 52 |
| | Debug my macros | 53 |
| | | |
| 7 | Using the Macro API | 55 |
| | Macro API Objects | 55 |
| | Attribute | 56 |
| | AttributeSet | 58 |
| | Color | 59 |
| | ControlKey | 59 |
| | DataCell | 64 |
| | Dimension | 65 |
| | Field | 66 |
| | FieldList | 68 |
| | OIA | 69 |
| | OIAStatus | 69 |
| | AutoSignon | 70 |
| | Position | 71 |
| | PresentationSpace | 72 |
| | Session | 74 |
| | SessionType | 75 |
| | StatusSet | 76 |
| | User Interface | 77 |
| | Wait | 77 |
| | Sample Macros | 79 |
| | Basic Host Interaction | 80 |
| | User Interaction | 81 |
| | Paging Through Data | 83 |
| | Invoking a Web Service | 84 |
| | Working with DataCells and Attributes | 86 |
| | Using Fields and Field Lists | 88 |
| | Automatic Sign-On Macro for Mainframes | 89 |
| | | |
| 8 | File and Data Transfer | 91 |
| | Mainframe file transfer | 91 |
| | Transferring files | 91 |
| | Troubleshooting your file transfers | 92 |
| | | |
| 9 | Logging | 93 |
| | | |
| 10 | Troubleshooting | 95 |

About Reflection ZFE

The Reflection ZFE web client provides browser-based HTML5 access to 3270, 5250, VT, and UTS host applications. The Reflection ZFE product eliminates the need to touch the desktop; no software to deploy, patches to apply, or configurations to make. You can provide platform-independent user access to all your host applications.

The web client operates with complete session protection using SSL/TLS to secure communication with your mainframe systems.

Reflection ZFE consists of these components:

- ◆ **Micro Focus Host Access Management and Security Server**

Reflection ZFE uses the Management and Security Server (MSS) Administrative Server to create and manage host sessions. The Administrative Console is a centralized web site that contains administrative tools and its own documentation. You can configure, secure, deploy, and monitor terminal sessions from this central location. You can use a compatible version of MSS or install the version that is available with your Reflection ZFE installation.

- ◆ **Session Server**

The session server is an NT service or UNIX daemon that provides the engine that runs host sessions. You can have multiple session servers serving up multiple sessions to provide efficient and rapid access to your host data.

Reflection ZFE Components for MSS – The Reflection ZFE management components must be installed on the Management and Security Server to use Reflection ZFE. If you select a remote Management and Security Server, the installation will verify that these components have been installed. If they have not been installed, you will need to install them.

- ◆ **Web Client**

The web client is a terminal emulator that can be accessed using only a browser. Once assigned, users can easily access authorized sessions from any platform and from any location.

- ◆ **Documentation**

Both the Reflection ZFE Web Client and Management and Security Server Administrative Console have complete documentation sets available from the user interface. Each product's documentation is also available from the [Micro Focus Support Site](#) in both HTML or PDF formats.

Getting Started as an Administrator

For a list of supported platforms and system requirements, see [Technical Note 2837](#).

- ◆ After you've installed Reflection ZFE, if you have additional deployment questions, see [Setting Installation Options](#).
- ◆ To walk-through a sample workflow, read the [Getting Started Guide](#). This guide starts with an administrator logging in to Micro Focus Host Access Management and Security Server Administrative Console and ends with your authenticated end-user connecting to a host session.
- ◆ You create and authorize sessions for specific users using the Administrative Console, which is the administrative component to MSS and available from the Reflection ZFE Start menu.

- ♦ You can use an URL to access Reflection ZFE (for example `https://sessionserver:7443/zfe`). Alternatively, if you are in an administrator role, you can access the Reflection ZFE web client using the Administrative Console Session Manager.
- ♦ The web application initially displays a connection page in which you can enter host-specific information and set other preferences for you and your users.

Getting Started as an End User

- ♦ You have access to a Reflection ZFE session using the URL provided by your administrator. It will look something like this: `https://sessionserver:7443/zfe`.
- ♦ [Using Sessions](#) provides how-to information and instructions on navigating Reflection ZFE.
- ♦ End-user macros are created by individuals for sessions they are authorized to access. The administrator grants permission to create macros when the session is created.
- ♦ If granted permission by your administrator, you can customize screen colors, configure hotspots, map keys, and have access to other user preferences.

1 Setting Installation Options

There are a number of post-installation configurations that you can make to ensure that Reflection ZFE runs successfully.

- ◆ [How to Adjust Session Timeout Values](#)
- ◆ [How to Set Up the Terminal ID Manager for Reflection ZFE](#)
- ◆ [How to Set Up Metering for Reflection ZFE](#)
- ◆ [How to Start and Stop Services Automatically](#)
- ◆ [How to Change Ports](#)
- ◆ [How to set up Automated Single Sign-On for Mainframe](#)

Related Topics

[Management and Security Server Installation Guide](#)
[Troubleshooting Reflection ZFE Connections](#)
[Configuring Session Options](#)

How to Adjust Session Timeout Values

The default timeout value for an inactive Reflection ZFE session is 30 minutes. This means that a session that was not logged out and has had no activity will close after 30 minutes. You can configure this setting on the server.

- 1 Open `<install_location>Micro Focus\ReflectionZFE\sessionserver\webapps\zfe|WEB-INF\web.xml`.

- 2 Adjust the session timeout value:

```
<session-config>
  <session-timeout>30</session-timeout> <!--In minutes-Minimum values of 5-->
  <cookie-config>
    <max-age>604800</max-age> <!--1 week in seconds-->
  </cookie-config>
</session-config>
```

- 3 Restart the server.

How to Set Up the Terminal ID Manager for Reflection ZFE

The Management and Security Server provides a Terminal ID Manager to pool terminal IDs, track ID usage, and manage inactivity timeout values for specific users, thus conserving terminal ID resources and significantly reducing operating expenses.

The Terminal ID Manager Add-On requires a separate license.

Before you configure the Terminal ID Manager for Reflection ZFE, verify that you have this option enabled for MSS. There are complete instructions in the [MSS Installation Guide](#).

TIP: If MSS and Reflection ZFE are installed on the same machine and using port 80, no additional configuration is needed.

Configuring Terminal ID Manager for Reflection ZFE

To configure the Terminal ID Manager for Reflection ZFE, you must provide the correct address to the Terminal ID Manager.

- 1 Open the `sessionserver/conf/container.properties` file.
- 2 Update `id.manager.server.url=http://localhost:80/tidm` to reflect the address of the Terminal ID Manager configured in Management and Security Server.
- 3 Restart the Reflection ZFE Session Server.

How to Set Up Metering for Reflection ZFE

The Management and Security Server provides metering capabilities to monitor Reflection ZFE host sessions.

Before you configure metering for Reflection ZFE, verify that you have metering enabled for MSS. There are complete instructions in the [MSS Installation Guide](#).

In Reflection ZFE metering is set globally for all emulation sessions created by the Reflection ZFE session server. Settings are configured in the `sessionserver/conf/container.properties` file.

Table 1-1 Metering options

| Property | Description |
|-------------------------------------|---|
| <code>metering.enabled</code> | Turns metering on or off, with a value of "true" or "false". Any value other than "true" turns metering off. |
| <code>metering.host.required</code> | Determines whether the session can connect to the host even if the metering server cannot be contacted. "True" means that session connections will fail if the metering host is unavailable. "False" means that session connections will still work even if the metering host is unavailable. |
| <code>metering.server.url</code> | Specifies the name or address of the metering server, the port, the protocol, and the webapp context. The syntax is "host:port protocol context". This syntax is the same as that used by the MSS server in the <code>MssData/serverconfig.props</code> file to register metering servers. The host:port section of the URL must escape the "." character. For example, <code>test990.attachmate.com\:8080</code> . |

```
#Example additions to sessionserver/conf/container.properties
metering.enabled=true
metering.host.required=false
metering.server.url=10.10.11.55\:80|http|meter
```

NOTE: In the event that all licenses are in use and you attempt to make a connection, the session will be disconnected. To determine whether the host has disconnected or the metering service has stopped the connection, see the Reflection ZFE/sessionserver/logs/server.log file.

How to Start and Stop Services Automatically

All server components are installed as services and can be configured to start during installation.

If you are running on Linux, Solaris, or AIX platforms, follow these steps to set the session server to start automatically when your system first boots up.

Create a file called `zfe` containing the following and using your installation directory:

```
#!/bin/sh
#
#This script manages the service needed to run the session server
#chkconfig:235 19 08
#description:Manage the Reflection ZFE session server

###BEGIN INIT INFO
# Provides:          zfe
# Required-Start:    $all
# Required-Stop:     $all
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Description:       Start the Reflection ZFE Session Server
### END INIT INFO

INSTALL_DIR=<enter installation directory>
BIN_DIR=$INSTALL_DIR/sessionserver/bin
case "$1" in
start)
echo "Starting Reflection ZFE Session Server"
$BIN_DIR/server start

RETVAL=0
;;
stop)
echo "Stopping Reflection ZFE Session Server"
$BIN_DIR/server stop

RETVAL=0
;;
status) echo "Current Reflection ZFE Session Server status"
$BIN_DIR/server status
```

```

RETVAL=0
;;
restart) echo "Restart Reflection ZFE Session Server"
$BIN_DIR/server restart

RETVAL=0
;;
*)
echo "Usage: $0 (start|stop|status|restart) "

RETVAL=1
;;
esac
exit $RETVAL

```

Then select your platform and complete the relevant steps.

| Platform | Follow these steps |
|----------|---|
| Linux | <ol style="list-style-type: none"> 1. Copy the file to the <code>/etc/init.d</code> directory. 2. Set the file permission. Run <code>chmod</code> using the value 755. For example, <code>chmod 755 zfe</code> 3. Run <code>chkconfig</code> to add the initialization script. For example, <code>/sbin/chkconfig --add zfe</code> |
| Solaris | <ol style="list-style-type: none"> 1. Copy the file to the <code>/etc/init.d</code> directory. 2. Set the file permission. Run <code>chmod</code> using the value 755. For example, <code>chmod 755</code> 3. Create a symbolic link in <code>/etc/rc.d/rc3.d</code>. For example, <code>ln -s/etc/init.d/zfe/etc/rc3.d/S99zfe</code> |
| AIX | <ol style="list-style-type: none"> 1. Copy the file to <code>/etc/rc.zfe</code>. For example, <code>cp zfe /etc/rc.zfe</code> 2. Set the file permission. Run <code>chmod</code>. For example, <code>chmod +x /etc/rc.zfe</code> 3. Add these lines at the end of the <code>/etc/rc.tcpip</code> file: <code>start/etc/rc.zfe</code> <code>" " "start"</code> |

How to Change Ports

Both the Reflection ZFE session server and MSS ports can be modified depending on your network needs. The default ports used by Reflection ZFE are:

Table 1-2 Reflection ZFE and MSS Default Ports

| | |
|-----------------------------------|-----------------------------|
| Session server | HTTP - 7070 HTTPS - 7443 |
| Reflection ZFE Components for MSS | HTTP - 7080 HTTPS - 7444 |
| Management and Security Server | HTTP - 80 HTTPS - 443 |

To change the default ports:

Table 1-3 Changing default ports

| Component | Instructions |
|--------------------------------|---|
| Reflection ZFE session server | <p>The Reflection ZFE session server ports are set, and can be modified, in <code>sessionserver/conf/container.properties</code>.</p> <pre>servletengine.port=7070 servletengine.ssl.port=7443</pre> <p>To turn the port off, set the port value to 0. You can disable your non-secure SSL port by changing the value from 7070 to 0.</p> |
| Management and Security Server | <p>The SSL port MSS uses to make an HTTPS connection is set to 443 by default. If you need to change the port number, start the Management Server. This creates the default <code>PropertyDS.xml</code> file. Then, open <code>PropertyDS.xml</code> in the <code>MssData</code> directory. Change the value from 443 to the appropriate port number in the section below, and then restart the Management Server.</p> <pre><CORE_PROPERTY NAME="sslport"> <STRING>443</STRING></pre> |

How to set up Automated Single Sign-On for Mainframe

Automated Sign-On for Mainframe is an add-on product to Management and Security Server that enables an end user to authenticate to a terminal emulation client and be automatically logged on to a host application on the z/OS mainframe.

The [Management and Security Server Administrator Guide for Automated Sign-On for Mainframe](#) has complete information on configuring this option.

- 1 Install and configure the Automated Sign-On for Mainframe add-on for Management and Security Server. You can find complete instructions [here](#).
- 2 After the Management and Security Server setup is complete, open the Administrative Console to add sessions and map users to those sessions. During that process, you can complete the [additional configuration](#) needed to implement automated sign-on.
- 3 A Reflection ZFE macro sends the user's mainframe username and pass ticket to the host application. The user is then automatically logged in. To help create the macro:
 - ♦ The Macro API contains the [AutoSignon](#) object that provides the methods needed to create a Reflection ZFE login to use with the Automated Sign-On for Mainframe feature.
 - ♦ You can also reference the sample macro [Automatic Sign-On Macro for Mainframes](#) that uses the `AutoSignon` object to create a macro that uses the credentials associated with a user to obtain a pass ticket from the Digital Certificate Access Server (DCAS).

Related Topics

- ♦ [Using the Macro API](#)
- ♦ [Macro API Objects](#)
- ♦ [Sample Macros](#)

2 Making Secure Connections

When you open up your legacy hosts to users outside the corporate firewall--business partners, remote users, mobile sales personnel, and others--you need to shield your information from known security threats. With Reflection ZFE, you can provide secure web-to-host access to all your users. Reflection ZFE, along with the Management and Security Server, provides HTTPS connections and a variety of authorization and authentication options.

In a typical Reflection ZFE installation there are three main connection points that you need to consider in regard to security:

- ♦ [Web browser to the Reflection ZFE Session Server](#)
- ♦ [Reflection ZFE Session Server to the Reflection Host Access Management and Security Server \(MSS\)](#)
- ♦ [Reflection ZFE Session Server to your legacy host systems](#)

There are instructions for securing each of these connections.

About Java Keytool and certificates

Reflection ZFE and MSS use the Java Key and Management Tool to manage keys and certificates. Using it, you can manage public/private key pairs and certificates. Keys and certificates are stored in a keystore, which, by default, is implemented as a file.

| To..... | Use this..... |
|------------------------------|--|
| List certificates | <code>keytool -list -keystore keystore.jks</code> |
| Delete certificates | <code>keytool -delete-alias mydomain -keystore keystore.jks</code> |
| Export a certificate | <code>keytool -export -alias mydomain -file mydomain.cer -keystore keystore.jks</code> |
| Import a certificate | <code>keytool -importcert -file <path to certificate> -alias <some-alias> -keystore servletcontainer.jks -storetype jceks -storepass not-secure</code> |
| View stand alone certificate | <code>keytool -printcert -v -file mydomain.crt</code> |

For more information, see the [Java Key and Certificate Management Tool](#) documentation.

Related Topics

- ♦ [Securing the Web Browser to Reflection ZFE Session Server](#)
- ♦ [Securing Reflection ZFE Session Server and Management Component to MSS](#)
- ♦ [Securing Reflection ZFE Session Server to the Host](#)

Securing the Web Browser to Reflection ZFE Session Server

NOTE: The file paths noted here are for a default installation. If you have installed Reflection ZFE to another location, you must modify the path appropriately.

When an HTTPS connection is made to the session server, it authenticates itself to the client browser using a server certificate. The client checks the server certificate against its trusted certificate store. If the certificate or its root is in the trusted store, the connection proceeds. However, if the certificate is not trusted, you are warned by the browser and asked to agree to the connection.

By default, a self-signed certificate is generated and used by the session server to identify itself to the client. When you initiate a secure web browser connection to the session server using the HTTPS URL with the self-signed certificate in use, you are warned by the browser that the certificate is not trusted. At this point you can instruct the browser to trust the certificate and proceed with a secure connection.

You can avoid the need to instruct the browser to trust the default self-signed certificate by configuring the session server to use a trusted certificate. The necessary certificate, which most likely will be one signed by a certificate authority (CA) that is universally trusted, is provided by the administrator in charge of the ZFE installation.

The key and certificate chain must be in a keystore in either JCEKS format, or a PKCS12 format with strong encryption (PBE-SHA1-3DES). The password for the key and for the keystore must be the same.

To configure the Reflection ZFE session server to use the keystore:

1. Open the `sessionserver/conf/container.properties` file and add the following three lines, making sure to use either forward slashes or double backward slashes:

```
servletengine.ssl.keystore=full path to keystore
servletengine.ssl.keystoretype=format name of keystore, either JCEKS or PKCS12
servletengine.ssl.keystorepassword=password for the keystore file you specified
```

2. Restart the session server.

It is possible to change the default behavior and disable the client browser from making an insecure HTTP connection to the ZFE session server from the start. To do this:

Open `sessionserver/conf/container.properties` and set the `servletengine.port` property to 0 and restart the session server.

Related Topics

- ♦ [Securing Reflection ZFE Session Server and Management Component to MSS](#)
- ♦ [Securing Reflection ZFE Session Server to the Host](#)

Securing Reflection ZFE Session Server and Management Component to MSS

NOTE: The file paths noted here are for a default installation and assume that `Java\bin` is in your system path. If you have installed Reflection ZFE to another location, you must modify the path appropriately.

These instructions pertain to both the session server and management component and require a change to the `container.properties` file located here:

- ♦ `sessionserver/conf/container.properties`
- ♦ `managementserver/conf/container.properties`

The `<component-path>/container.properties` file contains the URL of the Management and Security Server (MSS) that will be used by both Reflection ZFE session server and management component:

```
management.server.url=http://my-company.com:80/mss
```

During the installation, you can specify that you want to configure a secure communication channel between both the Reflection ZFE session server and MSS, which means the install process will handle obtaining the MSS certificate and configure the Reflection ZFE session server. The management component must be configured manually.

To make this configuration manually after you complete the installation follow these steps:

1. Change the `management.server.url` property in `<component-path>/conf/container.properties` to use the HTTPS protocol and specify the correct MSS port.
2. Use the browser to connect to the HTTPS Management and Security Server URL and instruct the browser to save the certificate.
3. Import the certificate into the appropriate Reflection ZFE keystore by running the following command (the command may vary depending on specific values) in the `<component-path>/etc` directory: `keytool -importcert -file <path-to-the-MSS-certificate> -alias <some-alias> -keystore servletcontainer.jks -storetype jceks -storepass not-secure`
4. Restart the appropriate service.

These instructions use the default password, **not-secure** as the keystore password. You can change the keystore password by running the following command in the `<component-path>/etc` directory:

```
keytool -storepasswd -new new_password -keystore servletcontainer.jks -storetype jceks -storepass not-secure.
```

Related Topics

- ♦ [Securing the Web Browser to Reflection ZFE Session Server](#)
- ♦ [Securing Reflection ZFE Session Server to the Host](#)

Securing Reflection ZFE Session Server to the Host

Follow these basic steps to configure a TLS connection between the Reflection ZFE session server and a host that supports TLS:

1. Install unlimited strength policy jars from Oracle.
2. Configure a keystore location on the Reflection ZFE session server. (Optional)
3. Configure the keystore using the MSS Administrative Server.
4. Configure a Reflection ZFE terminal session for TLS.

How to install unlimited strength policy jars

TLS/SSL encryption between the Reflection ZFE session server and the host computer requires the unlimited strength policy files from Oracle or IBM. If you installed using the standard installation process, these files are already installed. However, if needed, you can find the files here:

- ♦ For Oracle Java 8 - <http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html>
- ♦ For IBM Java 8 - <https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk>

The Oracle readme file included with the download explains how to install their files.

How to configure the keystore location on the Reflection ZFE Server

This step is not required if you are using a default Reflection ZFE installation. However, some performance adjustments can require a local keystore directory.

By default, Reflection ZFE creates an MSS parameter that points to a local directory, `keystore`, located under the current working directory of the Reflection ZFE session server (typically, `sessionserver/bin/keystore`). You can specify an alternate keystore directory by defining the Java system property, `haapi.ecl.keystore.location=<path_to_dir>`.

How to configure the keystore in MSS

For a Reflection ZFE session to trust the TLS host it connects to, the public certificate of the host must be added to a trusted keystore using the Reflection Management and Security Server (MSS). The Reflection ZFE session retrieves this certificate the first time a session connects.

To configure the keystore with the public certificates of trusted TLS hosts:

1. Connect to the system where MSS is installed.
2. Copy the public certificate file of the new trusted host into the `MssData/certificates` folder. In an automated Windows installation this file is located in `C:\ProgramData\Microsoft\Focus\mss\MssData`. The file can be either a base64-encoded DER certificate or a binary Reflection Administrative Server certificate.
3. Log in to MSS For example, <http://mycompany.com/mss/AdminStart.html>.
4. Click **Administrative WebStation** at the bottom right of the links list panel.

5. In the Administrative WebStation, click Security Setup section, and then open the Certificates tab.
6. On the Certificates tab, click the link to **View or modify certificates trusted by the terminal emulator applet**.
7. On the Certificates page, the certificates that were imported are listed at the top, while trusted root certificates (CA certificates) are listed in the bottom section of the page. To import a certificate for a new trusted host, click **Import**.
8. On the Import Trusted Certificate page, enter the name of the certificate file that you copied to `MssData/certificates`, the password of the public cert file, if needed, and a friendly name for identifying the certificate on the MSS certificates page.
9. Click **Submit**.

When the certificate is successfully added to the MSS server's trusted keystore, you are returned to the list of certificates and you should see the new host.

How to configure a Reflection ZFE terminal session

To connect to the new trusted host, configure a Reflection ZFE terminal session as usual, and in the Settings dialog box, specify TLS/SSL as the security protocol. Make sure to specify the correct TLS port for the connection.

Using Secure Shell (SSH)

Secure shell provides encrypted communications between the client and a VT host.

MSS has a known hosts list that contains the public keys of hosts that you can connect to using SSH. SSH connections can be made only to hosts already trusted by an administrator.

The first time an SSH connection is made from a Reflection ZFE session to a host, the known hosts file is downloaded from the Management and Security Server to the Reflection ZFE session server.

When you attempt to create or edit a session using SSH in Session Manager, you will be notified if the key is not recognized as trusted and asked if you want to trust the key and continue.

- ♦ If you enter yes, the host will be trusted and added to the known host list, and you will be prompted for the SSH host password.
- ♦ If you do not answer yes, then the host will remain untrusted and the session will be disconnected.

You can also configure the SSH Known Hosts file manually by establishing an SSH connection from a Reflection ZFE session to the host, and adding the remote host's key fingerprint to the known hosts list in MSS.

Configure known hosts file for SSH connections using MSS

To configure the known hosts file for SSH connections in MSS:

1. Connect to the system where MSS is installed and navigate to the server's certificates folder: `C:\ProgramData\Micro Focus\Mss\MssData\certificates` (Windows) or `/var/opt/microfocus/mss/Mssdata/certificates` (UNIX).
2. Copy the public certificate file of the new SSH host into the `MssData/certificates` (Windows) or `/etc/ssh/ssh_host_rsa_key.pub` (UNIX) folder described above. Only `ssh-rsa` and `ssh-dss` are valid as public key types for MSS `known_hosts` entries.

The host's public key format can be OpenSSH, Base64-encode, .DER, or.PFX. The file should follow this format: hostname, IP-address key-type key. For example, a public key entry might look like this: alpsuse132, 10.117.16.232 ssh-rsa
AAAAB3NzaC1yc2EAAAADAQAB.....

3. Log in to MSS (for example, <http://mycompany.com/mss/AdminStart.html>).
4. Click **Administrative WebStation** at the bottom right of the links list panel.
5. In the Administrative WebStation, click Security Setup, and open the Secure Shell tab.
6. In the **Administer Secure Shell Known Hosts List**, click the **View or modify secure shell known hosts list** link. The **Secure Shell Known Hosts** page displays.
7. Click **Import**. The **Import Known Host** page displays.
8. Enter the name of the file containing the public key, the name of the host, optionally the password for the public key file, and the IP address of the host. The name of the host you enter must exactly match the name on the key; for example, if the name on the key is *hostname.example.com*, you cannot enter just *hostname*.
9. Click **Submit**.

After the public key is imported into the Reflection Known Hosts file, you will return to the Secure Shell Known Hosts page and the new host will appear in the list.

-
- ♦ [Securing the Web Browser to Reflection ZFE Session Server](#)
 - ♦ [Securing Reflection ZFE Session Server and Management Component to MSS](#)

Configuring X.509 Authentication

X.509 client authentication allows clients to authenticate to servers with certificates rather than with a username and password by leveraging the X.509 public key infrastructure (PKI) standard.

Prerequisite

- ♦ Using the procedure described for a manual configuration in [Securing Reflection ZFE Session Server and Management Component to MSS](#) verify that a trusted certificate has been installed in the certificate store. The procedure may vary depending on your operating system and browser.

Basic steps:

1. Add Reflection ZFE certificates to the MSS Trusted Subsystem.
2. Enable the MSS Trusted Subsystem.
3. Install the signing authorities certificate into MSS and Reflection ZFE.
4. Restart the servers.
5. Configure X.509 in the Management and Security Server Administrative Console.

Step 1. Add Reflection ZFE certificates to the MSS Trusted Subsystem

This step makes sure that MSS can trust connections from the Reflection ZFE session server and management components.

1. Add the Reflection ZFE session server certificate to the MSS Trusted Subsystem:

```
In <MSS_install_directory>\server\etc add the certificate: keytool -importcert -file  
<RZFE_install_directory>\sessionserver\etc\servletcontainer.cer -alias  
zfesessionserver -keystore system.jks -storetype jceks -storepass not-secure
```

2. Add the management component certificate to the MSS Trusted Subsystem:

```
In <MSS_install_directory>\server\etc add the certificate: keytool -importcert -file
<RZFE_install_directory>\managementserver\etc\servletcontainer.cer -alias
zfmgtserver -keystore system.jks -storetype jceks -storepass not-secure
```

Step 2. Enable the MSS Trusted Subsystem

1. Open <MSS_install_directory>\server\conf\container.properties.
2. Add `servletengine.system.ports= <unused network port>` and save the file.

Step 3. Install the signing authorities certificate into MSS and Reflection ZFE

MSS's trusted store may already contain your signing authority certificate. This is often the case with well-known certificate signing authorities, and if so, then you can skip this step.

To check:

Open the Administrative WebStation, click Security Setup, and open the Certificates tab. Open **View or modify certificates trusted by the Management and Security Server** to see a list of available certificates.

If your certificate is not listed you need to install your signing root CA into MSS and into the Reflection ZFE session server.

- ◆ Installing into MSS:

1. Copy the certificate to the MSS server: `%PROGRAMDATA%\Micro Focus\MSS\MSSData\certificates`.
2. Open the Administrative WebStation. From the Start menu, open Micro Focus Host Access Management and Security Server | Administrative Server | Administrative WebStation.
3. In the **Administrative WebStation**, click **Security Setup** in the left panel, then click the **Certificates** tab. From the **Administer Management and Security Server Trusted Certificate List** section, click **View or modify certificates trusted by the Management and Security Server**.
4. Click **Import** in the Import Trusted Certificates section and fill out the fields to point to your certificate, specify its password and to give the certificate a friendly name
5. Click **Submit** and verify the certificate is listed.

- ◆ Installing the certificate into the Reflection ZFE session server and management component:

1. In <RZFE_install_directory>\sessionserver\etc import the certificate: `keytool -importcert -file <path to certificate> -alias zfesessionserver -keystore servletcontainer.jks -storetype jceks -storepass not-secure`
2. In <RZFE_install_directory>\managementserver\etc import the certificate: `keytool -importcert -file <path to certificate> -alias zfmgtserver -keystore servletcontainer.jks -storetype jceks -storepass not-secure`

Step 4. Restart all the servers

For the configuration to take affect, you must restart all servers.

Step 5. Configuring X.509 with LDAP fail over in the MSS Administrative Console

Once the certificates are in place, you can enable X.509 with LDAP fail over in **Management and Security Server Administrative Console | Access Control Setup**. See the Administrative Console online help for descriptions of the configuration options.

Configuring Single Sign-on through IIS

This option uses Microsoft IIS web server. This option requires no additional setup as long as you used the Management and Security Server automated installer and chose to integrate with IIS during the installation process. You can find more information on install configurations in the [Management and Security Server Installation Guide](#).

Enabling Reflection ZFE for use with single sign-on through IIS

To enable Reflection ZFE to work with this authentication method, add the following property in the `<install_dir>/sessionserver/conf/container.properties` file:

```
management.server.iis.url=<url>
```

The value of this property is the IIS web server address and port along with the / MSS path. For example: `http://server/mss`. The host name should be specified without the domain name in order for the domain credentials to be passed to IIS.

If you plan to proxy Reflection ZFE through IIS, when using IIS single sign-on, you need to set an additional property in the same `container.properties` file:

```
servletengine.iis.url=<url>
```

The value takes the same form as the URL above, but uses the Reflection ZFE address. For example: `http://server/zfe`. It is not necessary to use the short host name form in this URL. See [Technical Note 2859: Using the IIS Reverse Proxy with Reflection ZFE](#).

After you have completed this configuration, you choose this authentication option in [Management and Security Server Administrative Console | Access Control Setup](#). See the Administrative Console online help for descriptions of the configuration options.

Related Topics

- ♦ [Making Secure Connections](#)
- ♦ [Technical Note 2859: Using the IIS Reverse Proxy with Reflection ZFE](#)

Enabling FIPS Level Security

The Federal Information Processing Standards (FIPS) 140-2 validated cryptographic modules are used by the US federal government as a security regulation standard. Reflection ZFE supports this standard and you can easily enable FIPS mode by changing a property setting in the session server and in the management component.

To enable FIPS mode:

There is a `container.conf` file located in both the Reflection ZFE management component and the session server:

```
<install_directory>\sessionserver\conf\container.conf  
<install_directory>\managementserver\conf\container.conf
```

NOTE: You must modify the property in each location and restart the servers.

1. Open `container.conf`.

2. Modify the following property as follows:
`Dcom.attachmate.integration.container.BSAFE.FIPS.enabled=true.`
 3. Restart the server.
-

Related Topics

- ◆ [Technical Note 2400, Attachmate Products with FIPS 140-2 Validated Crypto Modules](#)
- ◆ [Technical Note 2783, Security Updates and Reflection ZFE](#)

Enabling SSL

SSL 3.0 has been deactivated by default in the JDK 8 `java.security` file due to discovered vulnerabilities.

However, for hosts that absolutely require SSL 3.0, you can follow these steps to enable the protocol:

- 1 Stop the applications or services that will be using SSL 3.0.
 - 2 Open `<install_dir>/jre/lib/security/java.security` in a text editor.
 - 3 Remove or comment out the line `jdk.tls.disabledAlgorithms=SSLv3`.
-

Related Topics

- ◆ [Configuring a Connection](#)

3 Configuring a Connection

Reflection ZFE supports IBM 3270 and 5250 hosts and VT and UTS host types.

Your users gain access to the host through sessions that you create and configure. Sessions are created by an administrator in the Management and Security Server Administrative Console. When you launch a session from the Administrative Console, the web client **Connection** panel opens in a separate browser window.

- ◆ [Connecting to the host](#)
- ◆ [Common connection settings](#)
- ◆ [3270 and 5250 connection settings](#)
- ◆ [VT connection settings](#)
- ◆ [UTS connection settings](#)
- ◆ [How to test Terminal ID Manager criteria](#)

Connecting to the host

The web client **Connection** panel is where you configure the connection options for your selected host type. These options vary depending on your host type.

After a session is created and launched in the Administrative Console, the web client interface displays. Here is where you set, not only connection options, but other preferences for both you and other session users so you can customize your interactions with the host.

NOTE: You can create a direct link to a specific Reflection ZFE session using the following format:

```
<rzfe-session-server:port>/?name=<session-name>
```

Users can access this link and directly launch a Reflection ZFE web session. A new session will not be launched if the specified session already exists when the user opens the link.

- 1 From the **Type** drop down list, select the type of host you are connecting to.
 - 2 Identify the host to which you want to connect. You can use a full host name or its IP address.
 - 3 Type the number of the port you want to use.
 - 4 Complete the information needed for the host connection.
 - 5 Save your connection settings.
-

Related Topics

- ◆ [VT connection settings](#)
- ◆ [UTS connection settings](#)
- ◆ [How to Set Up the Terminal ID Manager for Reflection ZFE](#)

Common connection settings

These options are common to all supported host types.

- ◆ **Connect at startup**

By default, sessions are configured to connect to the host automatically when you create or open a session. However, you can set up a session so that it doesn't automatically connect to the host. Choose **No** to manually connect to the host.

- ◆ **Protocol**

From the drop down list, select the protocol you want to use to communicate with the host. To establish a host connection, both the Reflection ZFE Web Client and the host computer must use the same network protocol. The available values are dependent on the host to which you are connecting. They are:

Table 3-1 Protocol Descriptions

| Protocol | Description |
|-------------------|--|
| TN3270 | TN3270 is a form of the Telnet protocol, which is a set of specifications for general communication between desktop and host systems. It uses TCP/IP as the transport between desktop computers and IBM mainframes. |
| TN3270E | TN3270E or Telnet Extended is for users of TCP/IP who connect to their IBM mainframe through a Telnet gateway that implements RFC 1647. The TN3270E protocol allows you to specify the connection device name (also known as LU name), and provides support for the ATTN key, the SYSREQ key, and SNA response handling. If you try to use Telnet Extended to connect to a gateway that doesn't support this protocol, standard TN3270 will be used instead. |
| TN5250 | TN5250 is a form of the Telnet protocol, which is a set of specifications for general communication between desktop and host systems. It uses TCP/IP as the transport between desktop computers and AS/400 computers. |
| Secure Shell (VT) | You can configure SSH connections when you need secure, encrypted communications between a trusted VT host and your computer over an insecure network. SSH connections ensure that both the client user and the host computer are authenticated; and that all data is encrypted |
| Telnet (VT) | Telnet Secure Socket Layer (SSL) and Transport Layer Security (TLS) are available to provide secure connections. |
| INT1 (UTS) | Provides access to Unisys 1100/1200 hosts using the TCP/IP network protocol. |

Related Topics

- ◆ [3270 and 5250 connection settings](#)
- ◆ [VT connection settings](#)
- ◆ [UTS connection settings](#)
- ◆ [How to Set Up the Terminal ID Manager for Reflection ZFE](#)

3270 and 5250 connection settings

In addition to the common configuration settings, 3270 and 5250 host types require these specific settings.

- ◆ **Device name**

If you selected TN3270, TN3270E, or TN5250 as the protocol, specify the device name to use when the session connects to the host. The device name is also known as the host LU or pool. If you do not specify a device name for the session, the host dynamically assigns one to the session. A device name that is set within a macro will override this setting.

If you selected **Terminal ID Manager** you can use it to provide IDs to client applications at runtime. You can use the Terminal ID Manager to manage pooled IDs for different host types. An ID is connection data that is unique for an individual host session. To use Terminal ID Manager, you must have a Terminal ID Manager server configured. See [Terminal ID Manager](#) in the Management and Security Server Installation Guide.

If you decide to use Terminal ID Manager and have configured the Terminal ID Manager server, then you can select from the options below to configure the criteria for acquiring an ID. All criteria must be met in order for an ID to be returned.

NOTE: Keep in mind that by specifying a criterion, you are indicating that the ID should be allocated only when an ID that has that specific value is found. The set of criteria selected here must be an exact match of the set of criteria specified on a least one Pool of IDs in Terminal ID Manager before the ID request can succeed.

Table 3-2 Terminal ID Manager Criteria

| Criterion | Description |
|-------------------|---|
| Pool name | Include this attribute and enter the name of the pool to limit the ID search to a specified pool. |
| Client IP address | The IP address of the client machine will be included as part of the request for an ID. |
| Host address | The address of the host configured for this session will be included as part of the request for an ID. |
| Host port | The port for the host configured for this session will be included as part of the request for an ID. |
| Session name | When selected, requires that the ID is configured to be used by this session exclusively. |
| Session type | The session type (for example, IBM 3270, IBM 5250, or UTS) is always included as part of any request for an ID. |

| Criterion | Description |
|-----------|-------------|
|-----------|-------------|

| | |
|-----------|--|
| User name | Use this criterion to ensure that only IDs created for exclusive use by specific users will be allocated. The current user's name, which must be found on an ID before it can be allocated, is the name of the user that the session is allocated to at runtime. |
|-----------|--|

To configure a session based on user names, a default place holder user name is available: **tidm-setup**.

For the administrator to configure sessions using **tidm-setup**, the Terminal ID Manager needs to have IDs provisioned for tidm-setup. You can override the default name with one of your own by modifying the `Micro Focus/ReflectionZFE/sessionserver/conf/container.properties` file as follows:

```
id.manager.user.name=custom-username
```

Where custom-username is replaced by the name you want to use.

| | |
|------------------------|---|
| Application name (UTS) | The name of the host application will be used as part of the request for an ID. |
|------------------------|---|

To determine the connection attempt behavior if Terminal ID Manager does not successfully allocate an ID to this session, use **If ID is not allocated**:

- ◆ **Fail connection attempt** -If selected, the session will not attempt to connect when an ID is not allocated.
- ◆ **Allow connection attempt** -If selected, the session will attempt to connect when an ID is not allocated. The attempt may be rejected by the host. There are some host types that permit a user to connect without an ID.

To confirm that Terminal ID Manager can provide an ID using the criterion and value selections you have made, click [Test](#).

- ◆ **Terminal model**

Choose the terminal model you are connecting to. (3270 host type only)

- ◆ **Host character set**

Select the 3270 or 5250 host character set you want to use. This setting chooses a conversion table to convert host characters (EBCDIC) into PC characters (ANSI). This setting should match the national character set used by your host system. If it doesn't match, then some characters, such as accents, may not display correctly. See your host documentation for definitions of the characters in each set. The default value is US English (037).

- ◆ **TLS/SSL Security**

SSL and TLS protocols allow a client and server to establish a secure, encrypted connection over a public network. When you connect using SSL/TLS, ZFE authenticates the server before opening a session, and all data passed between and the host is encrypted using the selected encryption level. The following options are available:

Table 3-3 TLS/SSL Descriptions

| Security options | Description |
|--------------------------------|--|
| TLS 1.2, 1.1, 1.0, and SSL 3.0 | Allow connection through TLS 1.2, TLS 1.1, TLS 1.0, or SSL 3.0, depending on the capabilities of the host or server to which you are connecting. |
| TLS 1.2, 1.1, 1.0 | Select this value to connect using TLS, but not SSL. As part of the TLS protocol, the client checks the server or host name against the name on the server certificate. Therefore, TLS connections require the common name on the server certificate to match the host or proxy server name. |
| SSL 3.0 | Select this value to connect using SSL, but not TLS. This is not recommended, but some hosts do not support TLS. If you plan on using this option, there is an additional configuration step to follow: SSL 3.0 has been deactivated by default in the JDK 8 java.security file. If SSL 3.0 is absolutely required, you can reactivate the protocol by removing "SSLv3" from the <code>jdk.tls.disabledAlgorithms</code> property in the java.security file or by dynamically setting this Security property to "true" before JSSE is initialized. For example: You must remove or comment out the line, <pre>jdk.tls.disabledAlgorithms=SSLv3, in <install_dir>\Micro Focus\ReflectionZFE\jre\jre\lib\security\java.security .</pre> |

NOTE: See the section on [Making Secure Connections](#) for information on adding trusted certificates, key stores, using SSH, and other advanced security information.

VT connection settings

In addition to the common connection settings, VT hosts require these additional settings:

Table 3-4 VT session configuration options

| VT Settings | Description |
|----------------------|--|
| Terminal ID | This setting determines the response that Reflection ZFE sends to the host after a primary device attributes (DA) request. This response lets the host know what terminal functions it can perform. The Reflection ZFE response for each Terminal ID is exactly the same as the VT terminal's response; some applications may require a specific DA response. This terminal ID setting is independent of the Terminal type setting. The options are: VT220, VT420, VT100, DEC-VT100, and VT52. |
| Local Echo | Automatic(default). How Reflection ZFE responds to remote echo from a Telnet host: Automatic attempts to negotiate remote echo, but does what the host commands. Yes means Reflection ZFE negotiates local echo with the host, but always echoes, while No means Reflection ZFE negotiates remote echo with the host, but does not echo. |
| Set Host Window Size | Yes (default). This setting sends the number of rows and columns to the Telnet host whenever they change. This enables the Telnet host to properly control the cursor if the window size is changed. |

| VT Settings | Description |
|--------------------|--|
| Request Binary | No (default). Telnet defines a 7-bit data path between the host and the terminal. This type of data path is not compatible with certain national character sets. Fortunately, many hosts allow for 8-bit data without zeroing the 8th bit, which resolves this problem. In some cases, however, it may be necessary to force the host to use an 8-bit data path by selecting this check box. |
| Send LF after CR | No (default). A "true" Telnet host expects to see a CrNu (carriage return/null) character sequence to indicate the end of a line sent from a terminal. There are some hosts on the Internet that are not true Telnet hosts, and they expect to see a Lf (linefeed) character following the Cr at the end of a line. If you're connecting to this type of Telnet host, select Yes. |
| Host Character Set | The default value for the Host character set depends on the type of terminal you are emulating. This setting reflects the current terminal state of VT Host Character Set, which can be changed by the host. The associated default setting, saved with the model is DEC Supplemental. |
| Auto Answerback | No (default). This setting specifies whether the answerback message (set with the Answerback property) is automatically sent to the host after a communications line connection. |
| Answerback String | This setting allows you to enter an answerback message if the host expects an answer in response to an ENQ character. |
| Display columns | Number of columns to display 80 (default) |
| Display rows | 24 rows (default). This setting specifies the number of rows to display. |

UTS connection settings

In addition to the common connection settings, UTS hosts require these additional settings:

Table 3-5 UTS INT1 session configuration options

| UTS INT1 options | Description |
|------------------|--|
| Application | <p>The name of the host application or host operating mode to be accessed.</p> <p>This is the word or phrase that the local machine sends to the host when you first establish communication with the host. If you were using a host terminal, this would be the \$\$OPEN name of the application. The application name is typically the same as the environment name. However, they can be different. For example, the environment name might be MAPPER, and the application might be UDSSRC. During a terminal emulation session, you would type \$\$OPEN MAPPER at the prompt, and INT1 would send UDSSRC to the host once the connection is established.</p> |
| TSAP | <p>The desired Transport Service Access Point (TSAP), up to 32 characters (such as TIPCSU for TIP connections, RSDCSU for Demand connections). A TSAP is required only if you are connecting to a Host LAN Controller (HLC) or to a Distributed Communications Processor (DCP) in IP router mode. If you're not sure which value to use, contact your host administrator.</p> |

| UTS INT1 options | Description |
|--------------------------|---|
| Initial transaction | The character, word, or phrase that the local machine will send to the host when communication with the host is first established (up to 15 characters). This parameter is optional and is primarily used with TIP. For example, you might type ^ to run MAPPER. This parameter can also be used to transmit passwords. |
| Start transaction | <p>When you configure an initial transaction, by default, the data is sent as soon as the session connection is established. You can decide when to send an initial transaction by using a particular string to trigger the initial transaction.</p> <p>For example, to wait for a successful login before sending initial transaction data, type in a string to be used to identify a successful login.</p> <p>You can use this setting in combination with Send initial transaction.</p> |
| Send initial transaction | <p>You can determine when the initial transaction is sent:</p> <ul style="list-style-type: none"> ◆ Immediately - Default. ◆ When start of entry (SOE) character is received - Useful when multi-line transactions must be completed before sending the string. ◆ After specified milliseconds |
| Terminal ID | <p>Choose whether to specify a terminal ID or use the Terminal ID Manager. To specify a terminal ID, type it in the Specify Terminal ID field.</p> <p>If you choose Use Terminal ID Manager, you are prompted to select the Terminal ID attributes you want to use to obtain an ID. See Terminal ID Manager Attributes.</p> <p>To test the attributes, click Click selected attribute.</p> |
| Specify Terminal ID | The Terminal ID, a terminal identifier (typically up to 8 alphanumeric characters) to use for the communication session associated with this path. Also known as a TID or PID, each terminal ID should be unique to the host. |

How to test Terminal ID Manager criteria

The Terminal ID Manager provides IDs to client applications at runtime. To confirm that Terminal ID Manager can provide an ID using the criteria and value selections you selected use this test option.

Criteria for the current session are specified on the Connection panel after selecting **Use Terminal ID Manager** from either the Device Name (3270, 5250 host types) or the Terminal ID (UTS) field. By default, the selected criteria for the current session are displayed.

Click **Get ID** to confirm that Terminal ID Manager can provide an ID matching the configured criterion and value selections. The test returns the name of an available ID that satisfies the selected attribute values.

Testing for other criteria and values

You can also use this panel to test criteria different from those associated with the current session.

1. Select any of the session types from the Session type list, and select the criteria you want to test. You can test alternate values that you want to use in a sample Terminal ID Manager request.
2. Click **Get ID** to confirm that Terminal ID Manager can provide an ID matching the criterion and value selections. The test returns the name of an available ID that satisfies the selected values.

Related topics

- ◆ [Configuring Session Options](#)
- ◆ [Terminal ID Manager Criteria](#)

4 Configuring Session Options

In addition to connection settings, you can personalize other display options, map keys, and specify different copy and paste functions.

- ◆ [How to configure Display Settings](#)
- ◆ [How to map keys](#)
- ◆ [How to configure macros for your users](#)
- ◆ [How to transfer files to and from the host](#)
- ◆ [How to specify copy and paste options](#)
- ◆ [How to set user preference rules](#)

How to configure Display Settings

You can customize host foreground and background colors and set up hotspots to make it easier to navigate the host application. These settings are specific to the session you are configuring.

How do I?

- ◆ [Customize host screen colors](#)
- ◆ [Set cursor options](#)
- ◆ [Set other display options](#)
- ◆ [Configure screen dimensions for VT and UTS hosts](#)
- ◆ [Configure hotspots](#)
- ◆ [How to map keys](#)
- ◆ [How to specify copy and paste options](#)

Customize host screen colors

You can customize the color of your screen and the appearance of different host attributes in the terminal window. For each item, you can select a color for the foreground and the background colors for 3270, 5250, VT and UTS host connections. Colors are specified using the color grid or by entering the Hex code format.

There are many web sites that list available Hex colors, for an example see [w3schools.com HTML Color Picker](#)

You may see different options depending on the type of host connection.

Options available to UTS hosts:

- ◆ **Use color information from the host** - To use the colors specified here rather than any colors specified by the host, clear this option.
- ◆ **Enable blink** - To disable blinking, clear this option.

- ◆ **Select attribute to edit** - In UTS emulation, colors are set directly by the host. You can specify colors for text associated with specific screen display options. Including the following and available combinations:
Plain, Underline(UND), Strikethru (STK), Left Column Separator (LCS), Control Page, and Status Line (OIA).
- ◆ **Video intensity**
The video intensities, Blink, Dim, Protected, and Reverse are combined with the attributes to create additional combinations. For example, you could map foreground or background colors to all cells with Dim + Blink + Underline or Reverse + Protected + Strikethru + Underline.
When you select a video intensity (or combination of intensities), those intensities are combined with the value of the attribute drop down list to form a single color mapping.

Options available to VT hosts:

- ◆ **Enable blink** - To disable blinking, clear this option.
- ◆ **Enable bold** - Displays text set with bold attributes as bold text in the terminal window. To display bold characters as plain text, clear this option.
- ◆ **Enable underline** - Displays text with underline.

To customize colors for all host types:

- 1 From the left navigation panel, click **Display**.
- 2 Under **Color Mappings**, click the background color field to open the color grid. From the color grid, select the color you want to use as the host background color. Alternatively, type the Hex color number for the color you want to use.
- 3 From the drop down list, select the default host color you want to change.
- 4 Open the color grid for the **Element Foreground** to choose a color to map the new color for the element text or type the Hex code you want to use. Select **Element Background** to map the new color to the element background field.
- 5 Click **Save** to close the Display panel and resume configuring your host connection.

Restore defaults clears any changes you made and resets the colors to the default host settings.

Configure hotspots

Hotspots are buttons that appear over common host commands in terminal sessions. When you use hotspots you can control the terminal session using a mouse or a finger-tap instead of the keyboard. The hotspot transmits a terminal key or command to the host. By default, hotspots are configured for the most common 3270, 5250, and VT commands.

Hotspots are enabled and visible by default, however you can disable hotspots for a particular session or choose to hide them.

- ◆ **Enable hotspots**
Choose **No** to disable hotspots for the session you are connecting to.
- ◆ **Show hotspots**
Choose **No** to hide hotspots on the screen. Hotspots remain functional.

Table 4-1 Hotspots for 3270 Hosts

| Hotspot | Description |
|------------------|--|
| PF1...PF24 | Transmits a PF1...PF24 to the host |
| PA1, PA2, or PA3 | Transmits a PA1, PA2, or PA3 to the host |
| enter | Transmits an Enter key to the host |
| more | Transmits a Clear key to the host |

Table 4-2 Hotspots for 5250 Hosts

| Hotspot | Description |
|------------|---|
| enter | Transmits an Enter key to the host |
| more... | Transmits a Roll Up key to the host (scrolls down one page) |
| PF1 - PF24 | Transmits a PF1...PF24 to the host |

Table 4-3 Hotspots for VT Hosts

| Hotspot | Description |
|----------|----------------------------------|
| F1 - F20 | Transmits a F1...F20 to the host |

Configure screen dimensions for VT and UTS hosts

As an administrator you can select the number of columns and rows for VT and UTS sessions.

- 1 Open the Display panel.
- 2 Under **Dimensions**, specify the number of columns and rows you want each screen to possess. The default values are 80 columns by 24 rows.
- 3 Click **Save**.

Set cursor options

Use the cursor options to configure the appearance and behavior of the cursor and ruler.

| This option | Does this.... |
|---------------|--|
| Cursor type | <ul style="list-style-type: none"> ◆ Underline displays the text cursor as an underline. ◆ Vertical bar displays the cursor as a vertical line. ◆ Block displays the text cursor as an inverse video block. |
| Ruler type | <ul style="list-style-type: none"> ◆ Vertical displays a vertical ruler at the cursor position. ◆ Horizontal displays a horizontal ruler at the cursor position. ◆ Crosshair displays both a horizontal and vertical ruler at the cursor position. |
| Cursor blinks | By default, the cursor (whether in block or underline mode) blinks. Clear this option to display a visible non-blinking cursor. |


Set other display options

| This option | Does this.... |
|---|--|
| Preserve aspect ratio | Select this option to maintain the host screen aspect ratio regardless of the size of the browser window. Aspect ratio describes the proportional relationship between the width of an image and its height. |
| Display OIA | Select this option to display the operation and status messages in the Operator Information Area (OIA) at the bottom of the terminal window . By default, OIA display is enabled. |
| Ignore mouse click on window activation | When a mouse click activates the terminal window, this option specifies whether actions such as updating the terminal cursor position, clearing a selection, or executing a hotspot are also performed. By default, these actions are not performed. |

How to map keys

You can create keyboard shortcuts that perform any assignable action during a session. The View Keyboard Map option on the left panel provides a view of the default keyboard map for each host type and the mapped custom keys for that session.

TIP: Browsers use keyboard shortcuts to save both time and mouse clicks. When mapping keystrokes it is important to keep this in mind. For example, Cntrl+F1 opens Internet Explorer help as well as the UTS control page. [Handy Keyboard Shortcuts](#) gives a brief overview of the keyboard shortcuts used by different browsers.

- 1 Open the **Manage Key Mappings** panel.
- 2 Under Key, click to add the key or key combination you want to use to trigger the assigned action in the Key field.
- 3 From the Mapped To drop down box, select the action you want to associate with the key selection.
- 4 Click the blue check mark to complete the association and add the key map to the session.
- 5 Click  to close the Key Mapping panel and resume configuring your host connection. Click **Connect** to save the configuration.

After you complete mapping keys for the session, click View Keyboard Map on the left panel to reference the updated map that includes the boldface custom key mappings. Click the column header to sort the list by Key or Mapped to action.

Different key stroke combinations are also used for copy/paste operations. For example, on a VT host screen, **CTL+ Shift + A** initiates a Select All action. See [How to specify copy and paste options](#) for a list of copy/paste key actions.

Related Topics

[IBM 3270 Keyboard Maps](#)

[IBM 5250 Keyboard Maps](#)

[VT Keyboard Maps](#)

[UTS Keyboard Maps](#)

[Using Quick Keys](#)

3270, 5250, VT, and UTS Keyboard Mapping

The following tables provide the default keys, key name, and key description for the different host keyboard mappings.

Table 4-4 IBM 3270 Keyboard Mapping

| Key | Maps to | Description |
|------------------|---------------------|--|
| Ctrl + F1 | Attention | Sends the ATTENTION key to the host |
| Shift + Tab | Backtab | Moves the cursor to the previous unprotected field |
| Ctrl + F2 | Clear | Clears the screen and sends the CLEAR key to the host |
| Alt + ArrowLeft | Cursor left double | Moves the cursor two positions to the left |
| Alt + ArrowRight | Cursor right double | Moves the cursor two positions to the right |
| Ctrl + F3 | Cursor select | Simulates a lightpen select in the current field |
| Alt + Delete | Delete word | Deletes three characters from the current field |
| Ctrl + 5 | Duplicate | Inserts the DUP character at the cursor location |
| Enter | Enter | Sends the ENTER key to the host |
| End | Erase end of field | Erases all data from the cursor location to the end of the current field |
| Alt + F5 | Erase input | Erases all data in all unprotected fields of the current screen. |
| Ctrl + Alt + F | Field delimiter | Toggles whether field delimiters are displayed on screen |
| Ctrl + 6 | Field mark | Inserts the Field Mark character at the cursor location |
| Home | Home | Moves the cursor to the first unprotected field on the screen |
| Insert | Insert | Toggles Insert mode |

| Key | Maps to | Description |
|---------------|----------------|--|
| Shift + Enter | New line | Moves to the next unprotected field |
| Ctrl + 1 | PA1 | Sends the PA1 key to the host |
| Pageup | PA1 | Sends the PA1 key to the host |
| Ctrl + 2 | PA2 | Sends the PA2 key to the host |
| Pagedown | PA2 | Sends the PA2 key to the host |
| Ctrl + 3 | PA3 | Sends the PA3 key to the host |
| F1 - F10 | PF1 - PF10 | Sends the PF1, PF2...PF10 key to the host |
| Alt + 1 | PF11 | Sends the PF11 key to the host |
| F11 | PF11 | Sends the PF11 key to the host |
| Alt + 2 | PF12 | Sends the PF12 key to the host |
| F12 | PF12 | Sends the PF12 key to the host |
| Shift + F1 | PF13 | Sends the PF13 key to the host |
| Shift + F2 | PF14 | Sends the PF14 key to the host |
| Shift + F3 | PF15 | Sends the PF15 key to the host |
| Shift + F4 | PF16 | Sends the PF16 key to the host |
| Shift + F5 | PF17 | Sends the PF17 key to the host |
| Shift + F6 | PF18 | Sends the PF18 key to the host |
| Shift + F7 | PF19 | Sends the PF19 key to the host |
| Shift + F8 | PF20 | Sends the PF20 key to the host |
| Shift + F9 | PF21 | Sends the PF21 key to the host |
| Shift + F10 | PF22 | Sends the PF22 key to the host |
| Alt3 | PF23 | Sends the PF23 key to the host |
| Shift + F11 | PF23 | Sends the PF23 key to the host |
| Alt4 | PF24 | Sends the PF24 key to the host |
| Shift + F12 | PF24 | Sends the PF24 key to the host |
| Ctrl +P | Print | Prints the contents of the screen to the printer |
| Escape | Reset | Resets keyboard error conditions |
| Ctrl + S | System request | Sends the SYSTEM REQUEST key to the host |

Table 4-5 IBM 5250 Keyboard Mapping

| Key | Maps to | Description |
|------------|----------------|--|
| Escape | Attention | Sends the ATTENTION key to the host |
| Ctrl + F2 | Clear | Clears the screen and send the CLEAR key to the host |

| Key | Maps to | Description |
|------------------|-----------------------|--|
| Ctrl + F3 | Cursor select | Simulates a lightpen select in the current field |
| Ctrl + Backspace | Destructive backspace | Moves the cursor one position to the left |
| Ctrl + 5 | Duplicate | Inserts the DUP character at the cursor location |
| Ctrl + End | End of field | Moves the cursor to the end of the field |
| End | Erase end of field | Erases all data from the cursor location to the end of the current field |
| Alt + End | Erase input | Erases all data in the all unprotected fields of the current screen |
| Alt + F5 | Erase input | Erases all data in all unprotected fields of the current screen. |
| Ctrl + Enter | Field exit | Moves the cursor out of an input field |
| KP + Subtract | Field exit minus | Moves the cursor out of a signed-numeric or numeric-only field |
| Ctrl + Subtract | Field exit minus | Moves the cursor out of a signed-numeric or numeric-only field |
| KP + Add | Field exit plus | Moves the cursor out of a signed-numeric or numeric-only field |
| Ctrl + Add | Field exit plus | Moves the cursor out of a signed-numeric or numeric-only field |
| Ctrl + 6 | Field mark | Inserts the field mark character at the cursor location |
| Ctrl + H | Help | Sends the Help key to the host |
| Ctrl + X | Hex mode | Places the terminal in Hex mode |
| Home | Home | Moves the cursor to the first unprotected field on the screen |
| Insert | Insert | Toggles Insert mode |
| Shift + Enter | New line | Moves to the next unprotected field |
| Ctrl + 1 | PA1 | Sends the PA1 key to the host |
| Ctrl + 2 | PA2 | Sends the PA2 key to the host |
| Ctrl + 3 | PA3 | Sends the PA3 key to the host |
| F1 - F11 | PF1 - PF11 | Sends the PF1, PF2....PF11 key to the host |
| Alt + 1 | PF11 | Sends the PF11 key to the host |
| Alt + 2 | PF12 | Sends the PF12 key to the host |
| F12 | PF12 | Sends the PF12 key to the host |
| Shift + 1 | PF13 | Sends the PF13 key to the host |
| Shift + F2 | PF14 | Sends the PF14 key to the host |
| Shift + F3 | PF15 | Sends the PF15 key to the host |
| Shift + F4 | PF16 | Sends the PF16 key to the host |

| Key | Maps to | Description |
|-------------|----------------|--|
| Shift + F5 | PF17 | Sends the PF17 key to the host |
| Shift + F6 | PF18 | Sends the PF18 key to the host |
| Shift + F7 | PF19 | Sends the PF19 key to the host |
| Shift + F8 | PF20 | Sends the PF20 key to the host |
| Shift + F9 | PF21 | Sends the PF21 key to the host |
| Shift + F10 | PF22 | Sends the PF22 key to the host |
| Alt + 3 | PF23 | Sends the PF23 key to the host |
| Shift + F11 | PF23 | Sends the PF23 key to the host |
| Alt + 4 | PF24 | Sends the PF24 key to the host |
| Shift + F12 | PF24 | Sends the PF24 key to the host |
| Ctrl + P | Print | Prints the contents of the screen to the printer |
| Control | Reset | Resets the keyboard error conditions |
| Pageup | RollDown | Sends the RollDown key to the host |
| Pagedown | RollUp | Sends the RollUp key to the host |
| Ctrl + Home | Start of field | Moves the cursor to the start of the field |
| Ctrl + S | System request | Sends the SYSTEM REQUEST key to the host |

Table 4-6 VT Keyboard Mapping

| Key | Maps to | Description |
|---------------|----------------|---------------------------------|
| Ctrl + Cancel | Break | Sends the Break key to the host |
| Ctrl + Enter | Enter | Send the Enter key to the host |
| Alt + F1 | F1 | Sends the F1 key to the host |
| Ctrl + F1 | F11 | Sends the F11 key to the host |
| Ctrl + F2 | F12 | Sends the F12 key to the host |
| Ctrl + F3 | F13 | Sends the F13 key to the host |
| Ctrl + F4 | F14 | Sends the F14 key to the host |
| Ctrl + F5 | F15 | Sends the F15 key to the host |
| Ctrl + F6 | F16 | Sends the F16 key to the host |
| Ctrl + F7 | F17 | Sends the F17 key to the host |
| Ctrl + F8 | F18 | Sends the F18 key to the host |
| Ctrl + F9 | F19 | Sends the F19 key to the host |
| Ctrl + F10 | F20 | Sends the F20 key to the host |
| Home | Find | Sends the Find key to the host |

| Key | Maps to | Description |
|-------------------|----------------|---|
| F1 | Hold | Sends the Hold Screen to the host |
| Pause | Hold | Sends the Hold Screen to the host |
| Insert | Insert | Sends the Insert key to the host |
| Ctrl + Insert | Keypad 0 | Sends the numeric keypad 0 key to the host |
| Ctrl + End | Keypad 1 | Sends the numeric keypad 1 key to the host |
| Ctrl + ArrowDown | Keypad 2 | Sends the numeric keypad 2 key to the host |
| Ctrl + Pagedown | Keypad 3 | Sends the numeric keypad 3 key to the host |
| Ctrl + ArrowLeft | Keypad 4 | Sends the numeric keypad 4 key to the host |
| Ctrl + Clear | Keypad 5 | Sends the numeric keypad 5 key to the host |
| Ctrl + ArrowRight | Keypad 6 | Sends the numeric keypad 6 key to the host |
| Ctrl + Home | Keypad 7 | Sends the numeric keypad 7 key to the host |
| Ctrl + ArrowUp | Keypad 8 | Sends the numeric keypad 8 key to the host |
| Ctrl + Pageup | Keypad 9 | Sends the numeric keypad 9 key to the host |
| Ctrl + Alt-add | Keypad comma | Sends the numeric keypad Comma key to the host |
| Ctrl + add | Keypad minus | Sends the numeric keypad Minus key to the host |
| Ctrl + decimal | Keypad period | Sends the numeric keypad Period key to the host |
| Ctrl + Delete | Keypad period | Sends the numeric keypad Period key to the host |
| Pagedown | Next | Sends the Next Screen key to the host |
| Ctrl + Pause | PF1 | Sends the PF1 key to the host |
| Ctrl + Divide | PF2 | Sends the PF2 key to the host |
| Ctrl + Multiply | PF3 | Sends the PF3 key to the host |
| Ctrl + Subtract | PF4 | Sends the PF4 key to the host |
| Pageup | Previous | Sends the Prev Screen key to the host |
| Delete | Remove | Sends the Remove key to the host |
| End | Select | Sends the Select key to the host |
| Shift + F6 | UDK6 | Sends the User Defined Key 6 to the host |
| Shift + F7 | UDK7 | Sends the User Defined Key 7 to the host |
| Shift + F8 | UDK8 | Sends the User Defined Key 8 to the host |
| Shift + F9 | UDK9 | Sends the User Defined Key 9 to the host |
| Shift + F10 | UDK10 | Sends the User Defined Key 10 to the host |
| Shift + Ctrl + F1 | UDK11 | Sends the User Defined Key 11 to the host |
| Shift + Ctrl + F2 | UDK12 | Sends the User Defined Key 12 to the host |
| Shift + Ctrl + F3 | UDK13 | Sends the User Defined Key 13 to the host |

| Key | Maps to | Description |
|--------------------|---------|---|
| Shift + Ctrl + F4 | UDK14 | Sends the User Defined Key 14 to the host |
| Shift + Ctrl + F5 | UDK15 | Sends the User Defined Key 15 to the host |
| Shift + Ctrl + F6 | UDK16 | Sends the User Defined Key 16 to the host |
| Shift + Ctrl + F7 | UDK17 | Sends the User Defined Key 17 to the host |
| Shift + Ctrl + F8 | UDK18 | Sends the User Defined Key 18 to the host |
| Shift + Ctrl + F9 | UDK19 | Sends the User Defined Key 19 to the host |
| Shift + Ctrl + F10 | UDK20 | Sends the User Defined Key 20 to the host |

Table 4-7 UTS Keyboard Mapping

| Key | Maps to | Description |
|-------------------|-----------------------------|---|
| F4 | Clear Change Bit | Sends the CLEARCHANGEBIT key to the host. |
| Keypad+Enter | Carriage Return | Sends a carriage return to the host. |
| Ctrl+PageDown | Clear End of Display | Clears text from the cursor location to the end of the display. |
| Ctrl+PageUp | Clear End of Display FCC | Clears all data (including FCC information) from the cursor to the end of the display |
| Ctrl+End | Clear End of Field | Clears text from the cursor location to the end of the field. |
| Ctrl+Shift+end | Clear End of Line | Clears text from the cursor location to the end of the row. |
| F7 | Clear FCC | Clears the field control character |
| Ctrl+Home | Clear Home | Sends the CLEAR_HOME key to the host. |
| Ctrl+H | Column Separator Right | Sends the COLUMN_SEP_RIGHT key to the host. |
| Ctrl+F1 | Control Page | Sends the CONTROL_PAGE key to the host. |
| Keypad+2 | Cursor Down | Moves the cursor one row down. |
| Keypad+4 | Cursor Left | Moves the cursor one column to the left. |
| Keypad+6 | Cursor Right | Moves the cursor one column to the right. |
| Keypad+8 | Cursor Up | Moves the cursor one row up. |
| Delete | Delete in Line | Sends the DELETE_IN_LINE key to the host. |
| Ctrl+Delete | Delete in Page | Sends the DELETE_IN_PAGE key to the host. |
| Ctrl+Shift+Delete | Delete Line | Deletes the row at the cursor location. |
| Ctrl+ArrowDown | Duplicate Line | Duplicates the row at the cursor location. |
| F8 | Enable FCC | Enables the field control character. |
| Keypad+- | End of Display and Transmit | Sends the EOD_AND_TRANSMIT key to the host. |
| Shift+End | End of Field | Moves the cursor to the end of the field. |

| Key | Maps to | Description |
|-------------------|-----------------|--|
| End | End of Line | Moves the cursor to the end of the row. |
| Ctrl+ArrowRight | End of Page | Moves the cursor to the end of the page. |
| Shift+Space | Erase Character | Erases the character at the cursor location. |
| Ctrl+Shift+E | Euro Character | Sends the Euro character to the host. |
| Ctrl+1 - Ctrl+9 | F1 - F9 | Sends the F1 - F9 key to the host |
| Ctrl+0 | F10 | Sends the F10 key to the host. |
| Ctrl+- | F11 | Sends the F11 key to the host. |
| Ctrl+= | F12 | Sends the F12 key to the host. |
| Ctrl+Q | F13 | Sends the F13 key to the host. |
| Ctrl+W | F14 | Sends the F14 key to the host. |
| Ctrl+E | F15 | Sends the F15 key to the host. |
| Ctrl+R | F16 | Sends the F16 key to the host. |
| Ctrl+T | F17 | Sends the F17 key to the host. |
| Ctrl+Y | F18 | Sends the F18 key to the host. |
| Ctrl+U | F19 | Sends the F19 key to the host. |
| Ctrl+I | F20 | Sends the F20 key to the host. |
| Ctrl+O | F21 | Sends the F21 key to the host. |
| Ctrl+P | F22 | Sends the F22 key to the host |
| Shift+F3 | FF | Sends a formfeed to the host. |
| F9 | Generate FCC | Generates a field control character. |
| Home | Home | Moves the cursor to the first field in the display. |
| Ctrl+Shift+Space | Insert in Line | Sends the INSERT_IN_LINE key to the host. |
| ICtrl+Space | Insert in Page | Sends the INSERT_IN_PAGE key to the host. |
| Ctrl+Shift+Insert | Insert Line | Inserts a new row into display memory. |
| Insert | Insert Mode | Toggles insert character mode. |
| F5 | Locate FCC | Disables the field control characters and moves to the first character of the next field to the right of the cursor. |
| F3 | Message Wait | Sends the MESSAGE_WAIT key to the host. |
| Shift+F2 | New Line | Moves the cursor to a new row. |
| Keypad+Shift+2 | Next Field | Moves the cursor to the next field. |
| Keypad+Shift+4 | Next Field | Moves the cursor to the next field |
| PageDown | Page Down | Sends the Page Down key to the host. |
| PageUp | Page Up | Sends the Page Up key to the host. |
| Keypad+Shift+6 | Previous Field | Moves the cursor to the previous field. |

| Key | Maps to | Description |
|----------------|--------------------------|--|
| Keypad+Shift+8 | Previous Field | Moves the cursor to the previous field. |
| Clear | SOE Character | Sends the SOE character to the host. |
| F12 | SOE Character | Sends the SOE character to the host. |
| Ctrl+Clear | Set Tab | Sends the SET_TAB key to the host. |
| Ctrl+Tab | Set Tab | Sends the SET_TAB key to the host. |
| Shift+Home | Start of Field | Moves the cursor to the start of the field. |
| Ctrl+ArrowLeft | Start of Line | Moves the cursor to the start of the row |
| Ctrl+[| System Mode | Sends the SYSTEM_MODE key to the host. |
| Ctrl+J | Toggle Column Separator | Toggles the column separator. |
| Ctrl+F12 | Toggle Message Wait Beep | Sends the TOGGLEMSGWAITBEEP key to the host. |
| Ctrl+L | Toggle Strike Thru | Toggles strike thru mode. |
| Ctrl+K | Toggle Underline | Toggles underline mode. |
| Ctrl+Enter | Transmit | Transmits the contents of the display to the host. |
| ScrollLock | Transmit | Transmits the contents of the display to the host. |
| Keypad++ | Transmit | Transmits the contents of the display to the host. |
| Keypad+Ctrl+ | Transmit | Transmits the contents of the display to the host. |
| Escape | Unlock | Sends the UNLOCK key to the host. |
| Ctrl+] | Workstation Mode | Sends the WORKSTATION_MODE key to the host. |

How to configure macros for your users

Use the Macro panel to select which macros to run and set when they should run.

- ◆ **Run macro on startup** - Choose a macro to run automatically when the session is opened.
- ◆ **Run macro on connect** - Choose a macro to run automatically when the session connects to the host.
- ◆ **Run macro on disconnect** - Choose a macro to run automatically when the session disconnects from the host.

Related Topics

[Using the Macro API](#)

[Sample Macros](#)

[Creating Macros](#)

How to transfer files to and from the host

You can transfer information between your computer and a 3270 host computer. From the drop down list, choose which IBM 3270 operating environment the host is running: CMS, TSO, or None (the default).

You cannot transfer files unless you are connected to the host. After establishing a connection, from the left panel, select **IND\$FILE** to open the File Transfer dialog box.

Related Topics

[File and Data Transfer](#)

How to specify copy and paste options

You can specify different options to use for copy and paste operations.

Copy options

Select text by dragging over it with the mouse. By default, different host types use different selection modes when copying text; IBM 3270, 5250 and UTS hosts use a block selection mode, while VT hosts use a linear selection mode. To toggle between block and linear selection modes, press and hold down the **Alt** key, then select the text.

- ◆ **Copy input fields only** - Select this option to only copy data from input fields. Data from protected fields is replaced with spaces when placed on the clipboard.
- ◆ **Use entire display when there is no selection** - This option applies the Copy command to the entire terminal display when nothing is selected.

Paste options

Click Paste to paste the contents of the clipboard at the cursor location.

- ◆ **Restore starting cursor position after paste**- By default, the host cursor is positioned at the end of the data following a paste operation. Select this option to restore the host cursor to its starting position after the paste operation is complete.

Key combinations

There are certain key combinations that map to different copy/paste actions.

| Key Combination | Host type | Action |
|-------------------|---------------------|--|
| Ctrl + A | UTS, 3270, 5250 | Select all |
| Shift + Arrow key | UTS, 3270, 5250, VT | Change the extent of the current selection |
| Ctrl + C | UTS, 3270, 5250 | Copy |
| Ctrl + V | UTS, 3270, 5250 | Paste |
| Ctrl + Shift + A | VT | Select all |
| Ctrl + Shift + C | VT | Copy |
| Ctrl + Shift + V | VT | Paste |

Related Topics

[Managing your sessions](#)
[Configuring a Connection](#)
[How to map keys](#)

How to set user preference rules

As an administrator you can choose what options users can configure for their sessions. These options are set on a per session basis and all users who have access to a particular session can configure their session instance.

- 1 From the left navigation panel, choose **User Preference Rules**.
- 2 Select which options you want to allow your users to configure.
- 3 Click Save.

Each user's configurations are specific to their instance of the session and will not conflict with those of other users.

Related Topics

[How to configure Display Settings](#)
[Customize host screen colors](#)
[Configure hotspots](#)
[How to map keys](#)
[How to configure macros for your users](#)
[How to transfer files to and from the host](#)

5 Using Sessions

All the sessions you have access to are available in the **Available Sessions** list. Sessions are initially created and configured by your system administrator and accessed through a distributed URL (for example, `https://<sessionserver>:7443/zfe`).

To open a session

- 1 Select the session and click to open.
- 2 Interact with your host application using the open session.
- 3 You can create multiple instances of a configured session.

Related Topics

- ♦ [Managing your sessions](#)
- ♦ [Logging out](#)

Managing your sessions

You access your sessions through the **Available Sessions** list. These are sessions that your administrator has assigned to you. Click to open the session.

You can have multiple sessions open at a time and easily switch between them using the tabs arranged across the top of the screen. The current session is always the left-most tab and is indicated by a white background and bold text. Each session remains active for 30 minutes.

Use the toolbar to access the various options available to you as you interact with the session. You can disconnect from a session, close the session, turn on Quick Keys, and access other settings. Some options may be only available once your administrator has granted permission.

- ♦ [Using Quick Keys](#)
- ♦ [Copying and Pasting](#)
- ♦ [Recording macros for the session](#)
- ♦ [Logging out](#)

Using Quick Keys


Click  on the toolbar to display Quick Keys. The Quick Key terminal keyboard provides a graphical representation of the keys on a host keyboard and gives you quick access to terminal keys. Click the terminal keys on the Quick Key keyboard to send a terminal key to the host. Hovering over a key provides a tool tip that shows the mapping.

Table 5-1 3270 Quick Keys

| Keys | Description |
|-----------------------------------|--|
| Program Attention Keys PA1-PA3 | Select to send a program attention key to the host. |
| Program Function Keys PF1-PF24 | Select to send a program function key to the host. |
| Attention | Select to send an Attention key to the host. |
| Enter | Select to send an Enter key to the host. |
| Erase | Select to erase all characters, from the cursor to the end of the entry. |
| Clear | Set buffer locations for the active partition to nulls and the Reply mode to the default, transmit the Clear Aid key to the host, and move the cursor position to the top left corner. |
| Reset | Clear the Input Inhibited indicator and reset the Insert mode. |
| SYSRQ | Sends the System Request key to the host. |

Table 5-2 5250 Quick Keys

| Key | Description |
|-----------------------------------|---|
| Program Function Keys PF1-PF24 | Select to send a program function key to the host. |
| Attention | Select to send an Attention key to the host. |
| Enter | Select to send an Enter key to the host. |
| Erase | Select to erase all characters, from the cursor to the end of the entry. |
| Clear | Signal the host to erase all user-entered text from the current screen. |
| Reset | Exit insert mode, diacritical mode, or hex mode; end help and system request functions; clear operator errors; and remove the Input Inhibited indicator and reset the Insert mode. Select Reset twice (consecutively) to exit Plus CR mode. |
| Roll Up | Select to scroll down one page in the current host screen. This option is equivalent to Page Down. |
| Roll Down | Select to scroll up one page in the current host screen. This option is equivalent to Page Up. |

Table 5-3 VT Quick keys

| Key | Description |
|-----------------------------------|--|
| VT Function Keys F1-F20 | Select to send a VT function key to the host. |
| Program Function Keys PF1-PF24 | Select to send a program function key to the host. |
| Break | Sends the Break key to the host. |
| Enter | Select to send an Enter key to the host. |
| Find | Sends the Find key to the host. <ESC>[1~ |
| Insert | Toggles Insert mode. <ESC>[2~ |
| Remove | Sends the Remove key to the host. <ESC>[3~ |
| Select | Sends the Select key to the host. <ESC>[4~ |
| Previous | Sends the Previous key to the host. <ESC>[5~ |
| Next | Sends the Next key to the host. <ESC>[6~ |

Copying and Pasting

NOTE: Each browser handles copy and paste functions differently and in some cases will not support the use of copy and paste buttons. It is highly recommended that you use keyboard commands for those functions. Although keyboard commands vary depending on your operating system, in Windows they are: **CTRL+C** to copy and **CTRL+V** to paste.


To copy from the terminal

- 1 Highlight the area on the terminal screen that you want to copy.
- 2 Click **Copy** from the toolbar or use the keyboard command, **CTRL+C**.

To paste into the terminal screen

- 1 Position the cursor where you want to paste content.
- 2 Use the keyboard command, **CTRL+V**, or click **Paste** from the toolbar. If your browser does not support pasting from the toolbar, you will be prompted to use the keyboard command.

Recording macros for the session

Click the macro icon  on the toolbar to start creating your own macros. If you do not see a macro icon, ask your administrator to set the appropriate permissions.

See [Creating Macros](#) for instructions on how to record macros.

Related Topics

- ◆ [Creating Macros](#)
- ◆ [Using the Macro API](#)

Logging out

In the upper right corner of the screen, open the drop down list associated with your user name and select **Logout** to stop working with the host application.

Related Topics

- ◆ [Using Sessions](#)
- ◆ [Creating Macros](#)

6 Creating Macros

A macro is a series of keyboard actions that you record and then run. You can use these JavaScript macro programs to automate user interactions with the terminal. You can access and run macros from all supported devices.

Reflection ZFE records and saves advanced macros as JavaScript, making it easy to edit and enhance your recorded macros. You can record macros to playback later, run macros at startup or when the session connects or disconnects from the host. You can also write macros from scratch to perform complex tasks that the recorder cannot capture.

Macros are made available to users in two ways; created by an administrator or recorded by users for their own private use. All advanced macros are associated with a session and they all accomplish the same goal, automating host interaction. The only difference between the two flavors is simply who can access them and who manages their creation and availability:

- ◆ **Macros created by administrators**

Administrators create macros when they create the session. They are specific to a session and are available to all users who have access to the session from the Macro icon on the toolbar. Administrators can designate macros to run at startup or when the session connects or disconnects from the host.

- ◆ **Macros created by users**

End-user macros are created by individuals for sessions they are authorized to access. The administrator grants permission to create macros by setting a User Preference Rule. Users can access the session under their own credentials or in a **Guest** role. Macros that Guest users create are available to all Guest users. Users who are logged in using their own credentials can only see macros that they have created.

Advanced macros are listed in alphabetical order in the drop down list available from the toolbar. Macros created by the end-user are listed first and followed by an indicator of three vertical grey dots, which when selected, displays the Edit and Delete options. Macros created by the administrator are listed without the indicator as those macros cannot be modified by the end-user.

Related Topics

[How do I work with macros?](#)

How do I?

- ◆ [Record a macro](#)
- ◆ [Edit a macro](#)
- ◆ [Run a macro](#)
- ◆ [Stop a macro](#)
- ◆ [Delete a macro](#)

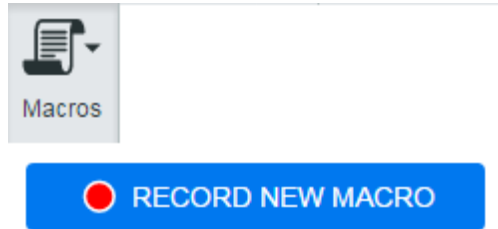
- ♦ [View my macros](#)
- ♦ [Debug my macros](#)


Record a macro

If you have been granted permission from your administrator, you can record macros to automate any series of host actions. Macros that were recorded by the administrator and associated with the current session are available to you from the Macro drop down list.

To record a macro:

1. Click the Macro icon on the toolbar, and then click Record New Macro.



2. Navigate the host application to record the series of steps you want included in the macro.
3. Click  on the toolbar to stop recording. The red dot pulses to indicate the recording is in process.
4. When prompted, type a name for the macro.

Related Topics

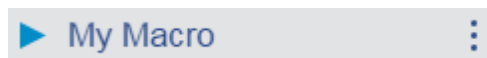
- ♦ [Run a macro](#)
- ♦ [Edit a macro](#)
- ♦ [Debug my macros](#)


Edit a macro

You can edit macros that you have recorded. These macros are listed under **My Macros**.

To edit an existing macro:

1. From the Macro drop down list, select the macro you want to edit.




2. Click the three vertical dots to expand the field.
3. Click  **Edit** to open the Macro Editor.
The Macro Editor opens in the left panel.
4. Use JavaScript to make whatever changes are necessary. You can run and save the modified macro using the toolbar icons in the upper panel of the editor.

Related Topics

- ♦ [Run a macro](#)
- ♦ [Record a macro](#)
- ♦ [Debug my macros](#)


Run a macro

To run a macro, choose the macro from the drop down list and click .

Related Topics

- ♦ [Edit a macro](#)
- ♦ [Record a macro](#)
- ♦ [Debug my macros](#)

Stop a macro

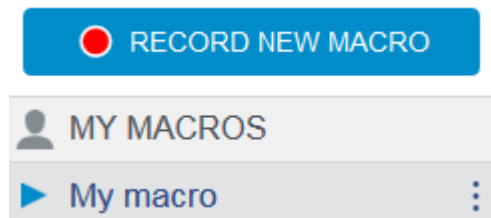
You can stop a macro before it completes from either the Macro Editor or the toolbar. Click  to stop the macro. To rerun the macro, navigate back to the macro starting screen.

Related Topics

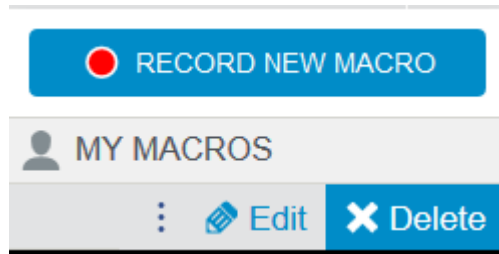
- ♦ [Edit a macro](#)
- ♦ [Record a macro](#)
- ♦ [Debug my macros](#)

Delete a macro

1. From the Macro drop down list, select the macro you want to delete.
2. Expand the field, by clicking the three vertical dot icon.



3. Click **Delete**.



Related Topics

- ◆ [Edit a macro](#)
- ◆ [Record a macro](#)
- ◆ [Debug my macros](#)

View my macros

The Macro drop down list is available from the toolbar to all users who have permission to record macros or are accessing a session where macros have been pre-recorded by the administrator for use with that session.

Macros are listed under either **MY MACROS** or **MACROS** depending on how they were recorded.

All users, whether they are logged in using their credentials or as Guest, can see the macros associated with the session. Macros listed under the MY MACROS heading are listed in alphabetical order by name and are visible to those users that recorded them. Macros recorded by the administrator and attached to a session are listed alphabetically under MACROS.



Related Topics

- ◆ [Edit a macro](#)
- ◆ [Record a macro](#)
- ◆ [Debug my macros](#)

Debug my macros

Since macros are written in JavaScript and executed in the browser, the best way to debug and troubleshoot them is by using your web browser's built-in tools. Modern browsers come with a very capable set of tools for debugging JavaScript code. You can place breakpoints, step through code, and output debug information.

TIP: JavaScript is case sensitive. Keep that in mind when editing JavaScript code.

To debug a macro:

1. Open the macro for editing. See [Edit a macro](#) for instructions.
2. Open your browser's development tools.

Table 6-1 Browser debugging support

| Browser | Open debugger |
|--------------------------------|---|
| Mozilla Firefox 40.0.3 | <ul style="list-style-type: none">◆ From the toolbar, open the Menu, and choose Developer.◆ From the Web Developer menu, choose Debugger. The debugger opens in a lower panel. |
| Google Chrome 45.0 | <ul style="list-style-type: none">◆ From the toolbar, open the Menu, and choose More tools.◆ Choose Developer Tools to open the Debugger. |
| Microsoft Internet Explorer 11 | <ul style="list-style-type: none">◆ From the toolbar, open Settings, and choose F12 Developer Tools.◆ Open the Debugger tab. |

These instructions are for supported browsers and are dependent on the versions used.

3. Use one of the these tools in your macro code, and run the code.

- ◆ *debugger*

The most thorough approach to debugging is to use the 'debugger;' statement. When you insert these statements into your macro code then run it, with the browser's development tools open, the execution will stop on those lines. You can step through your macro, view the value of local variables and whatever else you need to check.

You are encouraged to place multiple debugger; statements in your code to help get to the correct line. The asynchronous nature of JavaScript can make stepping through code challenging. This can be offset by using multiple, carefully placed debugger; statements.

Example 6-1 Debugger

```
-----  
var hostCommand = menuSelection + '[enter]';  
debugger; // <- Browser's debugger will stop here  
ps.sendKeys(hostCommand);  
-----
```

- ◆ `console.log()`, `alert()`

These two functions are commonly used for debugging JavaScript. While not as flexible as the debugger statement they provide a quick way to output debug information. These functions output the information to the JavaScript “Console” tab in the browser’s developer tools.

Example 6-2 *console.log(), alert()*

```
-----  
var hostCommand = menuSelection + `[enter]`;  
console.log('Command:' + hostCommand); // <- Will output the string to  
"Console" tab  
alert('Command:' + hostCommand); // Will pop up a small window containing  
the data  
ps.sendKeys(hostCommand);  
-----
```

◆ `ui.message()`

The Reflection ZFE Macro API provides an `ui.message()` function that is very similar to JavaScript’s `alert()` function. You can also use `ui.message()` to output debug information.

Example 6-3 *ui.message()*

```
-----  
var hostCommand = menuSelection + `[enter]`;  
ui.message('Command:' + hostCommand); // <- Will pop up a ZFE message  
window  
ps.sendKeys(hostCommand);  
-----
```

Notes to keep in mind when creating macros

◆ Stepping and “yields”

While the `yield` statements in macros make them easier to understand, they can make the code more challenging to step through with the debugger. Consider either using multiple debugger statements or carefully placed debugger statements of `console.log()` calls to output the right debug information.

◆ Internet Explorer

Debugging in Internet Explorer involves transformed code and may be more difficult than on other browsers.

Related Topics

- ◆ [Debug my macros](#)
- ◆ [Creating Macros](#)

7 Using the Macro API

In Reflection ZFE macros are recorded and written using JavaScript. JavaScript is a popular and prevalent programming language. There are a wide variety of learning resources and tools available to you.

The Reflection ZFE Macro API consists of a set of objects which you can use to interact with the host, wait for screen states, and interact with the user.

Important things to keep in mind

Because JavaScript is single threaded and uses 'callback functions' and 'promises' to help manage the flow of execution through code, often code can be difficult to follow. Reflection ZFE combines the concept of 'promises' with the 'yield' keyword so macro code can be organized in a more linear fashion.

- ◆ **Promises**

Promises are patterns to help simplify functions that return their result asynchronously, at some point in the future. All 'wait' and 'ui' functions in the Reflection ZFE Macro API return promise objects.

- ◆ **Yield**

Reflection ZFE macros use the yield keyword to block the execution of the macro until a promise is resolved, or done. So putting yield in front of any 'wait' or 'ui' function makes the macro execution pause until that function has finished executing. You can place the yield keyword in front of any function that returns a promise, even your own custom functions.

NOTE: The ability to make macro execution block by combining yield with promises is enabled by the `createMacro()` function.

Errors

Errors are handled in macros using a try / catch statement. Some API functions may throw errors if, for example, conditions can't be met or a timeout occurs. The thrown error is 'caught' in the catch statement. You can wrap smaller blocks of code in a try / catch statement to handle errors at a more granular level. Macro developers can also throw errors with `throw new Error('Helpful error message');`

- ◆ [Macro API Objects](#)
- ◆ [Sample Macros](#)

Macro API Objects

You can create macros using the Macro API. By default for use in macros, there are four primary objects available:

- ◆ **Session**

Session is the main entry point for access to the host. You use the Session object to connect, disconnect and provide access to the PresentationSpace object.

- ◆ [PresentationSpace](#)

The PresentationSpace object represents the screen and provides many common capabilities such as getting and setting the cursor location, sending data to the host and reading from the screen. It is obtained by calling `session.getPresentationSpace()`.

- ◆ [Wait](#)

Provides a simple way to wait for various host states to occur before continuing to send more data or read from the screen. For example, you can wait for the cursor to be located at a certain position, text to be present in a position on the screen or simply wait for a fixed amount of time. All 'Wait' function calls require the `yield` keyword, which is explained below.

- ◆ [User Interface](#)

The UI object is made automatically available in your macro as the "ui" variable. It provides basic user interface capabilities. You can use this object to display data to the user or prompt them for information. All 'UI' function calls require the `yield` keyword.

Other available objects

- ◆ [Attribute](#)
- ◆ [AttributeSet](#)
- ◆ [AutoSignon](#)
- ◆ [Color](#)
- ◆ [ControlKey](#)
- ◆ [DataCell](#)
- ◆ [Dimension](#)
- ◆ [Field](#)
- ◆ [FieldList](#)
- ◆ [OIA](#)
- ◆ [OIAStatus](#)
- ◆ [Position](#)
- ◆ [SessionType](#)
- ◆ [StatusSet](#)

Attribute

Use the Attribute, along with the AttributeSet, to decode the formatting information present on the data cell.

Table 7-1 Attributes

| Attribute | Description |
|----------------|---|
| PROTECTED | Indicates a protected data cell. |
| MODIFIED | Indicates a modified data cell. |
| NUMERIC_ONLY | Indicates the beginning of a numeric only data cell. |
| ALPHA_NUMERIC | Indicates an alpha numeric data cell. |
| HIGH_INTENSITY | Indicates whether the data cell contains high intensity text. |

| Attribute | Description |
|----------------------------|---|
| HIDDEN | Indicates whether the data cell contains hidden text |
| PEN_DETECTABLE | Indicates whether the data cell is pen detectable |
| ALPHA_ONLY | Indicates an alpha only data cell. |
| NUMERIC_SHIFT | Indicates the beginning of a numeric shift. field |
| NUMERIC_SPECIAL | Indicates the data cell marks the beginning of a numeric special field |
| KATAKANA_SHIFT | Indicates a section of Katakana text. |
| MAGNETIC_STRIPE | Indicates the data cell marks the beginning of a magnetic strip field. |
| SIGNED_NUMERIC_ONLY | Indicates the data cell is a signed numeric field. |
| TRANSMIT_ONLY | Indicates the data cell is a transmit only field |
| FIELD_END_MARKER | Indicates the data cell marks the end of a modified field. |
| FIELD_START_MARKER | Indicates the data cell marks the start of a modified field. |
| SPECIAL_EMPHASIS_PROTECTED | Indicates a special emphasis protected field. |
| TAB_STOP | Indicates that the data cell contains a tab stop. |
| REVERSE | Indicates the data cell displays in reverse video mode. |
| BLINKING | Indicates the data cell contains blinking text |
| RIGHT_JUSTIFIED | Indicates the data cell marks the beginning of a right justified field. |
| LEFT_JUSTIFIED | Indicates the data cell marks the beginning of a left justified field. |
| LOW_INTENSITY | Indicates the data cell contains low intensity text |
| UNDERLINE | Indicates the data cell contains underlined text. |
| DOUBLE_BYTE | Indicates the data cell contains double byte text. |
| COLUMN_SEPARATOR | Indicates the data cell contains a column separator. |
| BOLD | Indicates the data cell contains bold text. |
| DOUBLE_WIDTH | Indicates the data cell marks a double width field. |
| DOUBLE_HEIGHT_TOP | Indicates a double height top data cell. |
| DOUBLE_HEIGHT_BOTTOM | Indicates a double height bottom data cell. |
| CONTROL_PAGE_DATA | Indicates the data cell contains control page data. |
| RIGHT_COLUMN_SEPARATOR | Indicates the data cell contains a right column separator. |
| LEFT_COLUMN_SEPARATOR | Indicates a data cell containing a left column separator. |
| UPPERSCORE | Indicates the data cell contains an upperscore. |
| STRIKE_THROUGH | Indicates the data cell contains strike through text. |

AttributeSet

The AttributeSet object allows the user to decode the attributes that are present on the data cell. The AttributeSet object returns values defined in the [Attribute](#) object and when used together, you can get formatting information from the data cell.

Table 7-2 AttributeSet

METHODS

| | |
|--|--|
| <code>contains(attribute)</code> | Determines if the set contains the specified Attribute . |
| | Parameters |
| | {Number} attribute to check |
| | Returns |
| | {Boolean} True if the attribute is in the set. |
| <code>isEmpty()</code> | Determines if the attribute set is empty. |
| | Returns |
| | {Boolean} True if the set is empty. |
| <code>size()</code> | Indicates the number of attributes in a set. |
| | Returns |
| | {Number} The attribute count. |
| <code>toArray()</code> | Converts the internal attribute set to an array. |
| | Returns |
| | {Number[]} Array of values of attributes in the set. |
| <code>toString()</code> | Converts the internal attribute set to a string. |
| | Returns |
| | {String} Space-delimited names of attributes in the set. |
| <code>forEach(callback, thisArg)</code> | Function to iterate over each element in the attribute set. |
| | Parameters |
| | {forEachCallback} Callback to perform the specific operation. Called with the name of each attribute in the set. |
| | {Object} this Arg optional pointer to a context object. |
| <code>forEachCallback(string, object)</code> | A user provided callback function where you provide the behavior, to be used as the callback parameter to <code>forEach</code> . |
| | Parameters |
| | {String} String name of an attribute in the attribute set. |
| | {Object} thisArg optional pointer to a context object. |

Color

Color constants to use for the DataCell object foreground and background colors.

Table 7-3 Color constants

| Color | Description | Numeric Value |
|------------------------|------------------------|---------------|
| BLANK_UNSPECIFIED | No color specified | 0 |
| BLUE | Blue | 1 |
| GREEN | Green | 2 |
| CYAN | Cyan | 3 |
| RED | Red | 4 |
| MAGENTA | Magenta | 5 |
| YELLOW | Yellow | 6 |
| WHITE_NORMAL_INTENSITY | Normal intensity white | 7 |
| GRAY | Gray | 8 |
| LIGHT_BLUE | Light blue | 9 |
| LIGHT_GREEN | Light green | 10 |
| LIGHT_CYAN | Light cyan | 11 |
| LIGHT_RED | Light red | 12 |
| LIGHT_MAGENTA | Light magenta | 13 |
| BLACK | Black | 14 |
| WHITE_HIGH_INTENSITY | High intensity white | 15 |
| BROWN | Brown | 16 |
| PINK | Pink | 17 |
| TURQUOISE | Turquoise | 18 |

ControlKey

The ControlKey object defines constants for sending cursor control keys and host commands using the sendKeys method. Constants are available for these host types:

- ◆ [IBM 3270](#)
- ◆ [IBM 5250](#)
- ◆ [VT](#)
- ◆ [UTS](#)

IBM 3270

Table 7-4 IBM 3270

| Key word | Description |
|-----------------|--------------------------|
| ALTVIEW | Alternate view |
| ATTN | Attention |
| BACKSPACE | Back space |
| BACKTAB | Back tab |
| CLEAR | Clear or clear display |
| CURSOR_SELECT | Cursor select |
| DELETE_CHAR | Delete, delete character |
| DELETE_WORD | Delete word |
| DEST_BACK | Destructive backspace |
| DEV_CANCEL | Device cancel |
| DOWN | Cursor down |
| DSPSOSI | display SO/SI |
| DUP | Duplicate field |
| END_FILE | End of field |
| ENTER | Enter |
| ERASE_EOF | Erase end of field |
| ERASE_FIELD | Erase field |
| ERASE_INPUT | Erase input |
| FIELD_MARK | Field mark |
| HOME | Cursor home |
| IDENT | Ident |
| INSERT | Insert |
| LEFT_ARROW | Cursor left |
| LEFT2 | Left two positions |
| NEW_LINE | New line |
| PA1 - PA3 | PA1 - PA3 |
| PF1 - PF24 | PF1 - PF24 |
| PAGE_DOWN | Page down |
| PAGE_UP | Page up |
| RESET | Reset, reset terminal |
| RIGHT2 | Right 2 positions |
| RIGHT_ARROW | Cursor right, right |

| Key word | Description |
|-----------------|--------------------|
| SYSTEM_REQUEST | System request |
| TAB | Tab key |
| UP | Cursor up |

IBM 5250

Table 7-5 IBM 5250

| Key word | Description |
|-----------------|--------------------------|
| ALTVIEW | Alternate view |
| ATTN | Attention |
| AU1 - AU16 | AU1 - AU16 |
| BACKSPACE | Back space |
| BACKTAB | Back tab |
| BEGIN_FIELD | Begin field |
| CLEAR | Clear |
| DELETE_CHAR | Delete, delete character |
| DEST_BACK | Destructive backspace |
| DOWN | cursor down |
| DSPSOSI | Display SO/SI |
| DUP | Duplicate field |
| END_FILE | End of field |
| ENTER | Enter |
| ERASE_EOF | Erase end of field |
| ERASE_FIELD | Erase field |
| ERASE_INPUT | Erase input |
| FIELD_EXT | Field exit |
| FIELD_MINUS | Field minus |
| FIELD_PLUS | Field plus |
| FIELD_MARK | Field mark |
| HELP | Help request |
| HEXMODE | Hex mode |
| HOME | cursor home |
| INSERT | Insert |
| LEFT_ARROW | Cursor left |

| Key word | Description |
|-----------------|-----------------------|
| NEW_LINE | New line |
| PA1 - PA3 | PA1 - PA3 |
| [PF1 - PF24 | PF1 - PF24 |
| [print] | Print |
| RESET | Reset, reset terminal |
| RIGHT_ARROW | Cursor right, right |
| PAGE_UP | Page up |
| PAGE_DOWN | Page down |
| SYSTEM_REQUEST | System request |
| TAB | Tab |
| UP | Cursor up |

VT

Table 7-6 VT

| Keywords | Description |
|-----------------|--------------------------|
| BACKSPACE | Back space |
| BREAK | Break |
| CLEAR | Clear or clear display |
| CURSOR_SELECT | Cursor select |
| DELETE_CHAR | Delete, delete character |
| DOWN | Cursor down |
| EK_FIND | Edit keypad find |
| EK_INSERT | Edit keypad insert |
| EK_NEXT | Edit keypad next |
| EK_PREV | Edit keypad previous |
| EK_REMOVE | Edit keypad remove |
| EK_SELECT | Edit keypad select |
| ENTER | Enter |
| END_FILE | End of field |
| F1 - F24 | F1 - F24 |
| HOLD | Hold |
| HOME | Home |
| INSERT | Insert |

| Keywords | Description |
|-------------------|--|
| KEYPAD_COMMA | Keypad comma |
| KEYPAD_DOT | Keypad decimal |
| KEYPAD_MINUS | Keypad minus |
| KEYPAD_ENTER | Keypad enter |
| KEYPAD0 - KEYPAD9 | Keypad 0 - Keypad 9 |
| LEFT_ARROW: | Cursor left |
| PF1 - PF20 | PF1 - PF20 |
| PAGE_DOWN | Page down |
| PAGE_UP | Page up |
| RESET | Reset, reset terminal |
| RETURN | Return, carriage return |
| RIGHT_ARROW | Cursor right, right |
| TAB | Tab key |
| UDK16 - UDK20 | User defined key 6 - User defined key 20 |
| UP | Cursor up |

UTS

Table 7-7 UTS

| Key word | Description |
|-----------------|---|
| BACKSPACE | Moves the cursor to the previous tab position on the screen. |
| BACKTAB | Back tab <Shift> <Tab> |
| CHAR_ERASE | Erases character at the cursor and advances the cursor. |
| CLEAR_DISPLAY | Clear display |
| CLEAR_EOD | Clear to end of display |
| CLEAR_EOF | Clear to end of field |
| CLEAR_EOL | Clear to end of line |
| CLEAR_FCC | Clear Field Control Character |
| CLEAR_HOME | Clear display and cursor home |
| CONTROL_PAGE | Toggles the control page |
| DELETE_LINE | Deletes the line containing the cursor and shifts remaining lines up one row |
| DOWN | Moves the cursor down one line. Wraps at bottom. |
| DELIN_LINE | Deletes character under cursor and shifts remaining characters on line to the left. |

| Key word | Description |
|--------------------|---|
| DELIN_PAGE | Deletes character under cursor and shifts remaining characters on page to the left. |
| DUP_LINE | Creates a copy of the current line and overwrites the next line with the duplicate. |
| EURO | Inserts the Euro character |
| END_FIELD | Moves the cursor to the end of the current field. |
| END_PAGE | Moves the cursor to the end of the current page. |
| F1 - F22 | Function keys F1-F22 |
| HOME | Moves the cursor to beginning of current page (row 1, col 1) |
| INSERT | Toggles insert/overwrite mode. |
| INSERT_IN_LINE | Inserts space at cursor position and shifts the remaining characters on the line to the right. The character in the far right column on the line is discarded. |
| INSERT_IN_PAGE | Inserts space at cursor position and shifts the remaining characters on the page to the right. The character in the far right column on each line is discarded. |
| INSERT_LINE | Inserts a new line at the cursor row and shifts the remaining lines down. The last row on the page is discarded. |
| LEFT_ARROW | Moves the cursor one position to the left wrapping if necessary. |
| LOCATE_FCC | Finds the next field control character on the screen. |
| MSG_WAIT | Retrieves messages queued to the terminal. |
| RETURN | Carriage return |
| RIGHT_ARROW | Moves the cursor one position to the right, wrapping if necessary. |
| SOE | Inserts the Start of Entry character |
| START_OF_FIELD | Moves the cursor to the beginning of the field. |
| START_OF_LINE | Moves the cursor to column 1 of current line. |
| TAB | Moves the cursor to the next tab position of the screen. |
| TOGGLE_COL_SEP | Toggles the column separator attribute. |
| TOGGLE_STRIKE_THRU | Toggles the strike-through attribute on the current data cell. |
| TOGGLE_UNDERLINE | Toggles the underline attribute on the current data cell. |
| TRANSMIT | Transmits changed field data to the host. |
| UNLOCK | Sends the UNLOCK key to the host. |
| UP | Moves the cursor up one row, wrapping if necessary. |

DataCell

The DataCell object provides information about a particular position on a terminal screen.

Table 7-8 *DataCell*

METHODS

| | |
|-----------------------------------|--|
| <code>getPosition()</code> | Returns the position of this data cell on the screen. Returns {Position} the position of the data cell on the screen |
| <code>getChar()</code> | Obtains the character associated with the cell. Returns {String} The character associated with the cell. |
| <code>getAttributes()</code> | Returns the set of attributes specified for this data cell instance. See AttributeSet . Returns {AttributeSet} Of attributes for this data cell instance. |
| <code>getForegroundColor()</code> | Returns the foreground color, as defined in the Color object, for this data cell. Returns {Number} Foreground color for this data cell. The color is defined in the Color object. |
| <code>getBackgroundColor()</code> | Returns the background color, as defined in the Color object, for this data cell. Returns {Number} Background color for this data cell. The color is defined in the Color object. |
| <code>toString</code> | Converts the internal data cell to a string. Returns {String} The string representation of a data cell. |
| <code>isFieldDelimiter()</code> | Tests if this cell represents a field delimiter. Returns {Boolean} True if this cell is a field delimiter, false if otherwise. |

Dimension

Represents the size of the screen or screen area.

Table 7-9 *Dimension*

| Method | |
|------------------------------------|---|
| <code>Dimension(rows, cols)</code> | Creates a new Dimension instance. |
| Parameters | |
| | <code>{Number}</code> rows screen rows dimension |
| | <code>{Number}</code> cols screen columns dimension |

Field

Use the Field object, along with [FieldList](#), to obtain the information present in a field on the screen.

Table 7-10 *Field*

| Method | |
|-----------------------------------|---|
| <code>getAttributes()</code> | Returns the set of attributes specified for this field instance. See AttributeSet . |
| Returns | |
| | <code>{AttributeSet}</code> The set of attributes for this field |
| <code>getForegroundColor()</code> | Returns the foreground color of the field. |
| Returns | |
| | <code>{Number}</code> the foreground color for this field. These values are defined in the Color object. |
| <code>getBackgroundColor()</code> | Returns the background color of the field. |
| Returns | |
| | <code>{Number}</code> the background color for this field. These values are defined in the Color object. |
| <code>getStart()</code> | Returns the starting position of the field. The starting position is the position of the first character of the field. Some host types use a character position to store field level attributes. In this case, the attribute position is not considered the start position. |
| Returns | |
| | <code>{Position}</code> Starting position of the field. |
| Throws | |
| | <code>{RangeError}</code> For zero length fields. |

Method

| | |
|-------------------------------------|---|
| <code>getEnd()</code> | <p>Returns the ending position of the field. The ending position is the position in the presentation space containing the last character of the field.</p> <p>Returns</p> <p>{Position} Ending position of the field.</p> <p>Throws</p> <p>{RangeError} For zero length fields.</p> |
| <code>getLength()</code> | <p>Returns the length of the field. For host types that use a character position to store the field attributes, the field length does not include the field attribute position.</p> <p>Returns</p> <p>{Number} Length of the field.</p> |
| <code>getDataCells()</code> | <p>Obtains the data cells that comprise this field. See DataCell.</p> <p>Returns</p> <p>{DataCell[]} Data cells that comprise this field.</p> |
| <code>getText()</code> | <p>Obtains the text from the field.</p> <p>Returns</p> <p>{String} field text.</p> |
| <code>setText()</code> | <p>Sets the field text. For certain host types, like VT, the text is transmitted to the host right away, but in other host types, the text is not transmitted to the host until an Aid key is invoked. If the text is shorter than the field, the text is placed in the host field, and the remainder of the field is cleared. If the text is longer than the host field, then as much text as will fit is placed in the field.</p> <p>Parameters</p> <p>{String} Text to set on the field.</p> <p>Throws</p> <p>{Error} If the field is protected.</p> |
| <code>clearField()</code> | <p>Clears the current field in an emulation-specific manner.</p> <p>Throws</p> <p>{Error} If the field is protected or clear is not supported.</p> |
| <code>getPresentationSpace()</code> | <p>Obtains the presentation space which created this field.</p> <p>Returns</p> <p>{PresentationSpace} Parent of this field instance.</p> |
| <code>toString()</code> | <p>Creates a user-friendly description of the field.</p> <p>Returns</p> <p>{String} A user readable rendition of the field.</p> |

FieldList

Use the FieldList object, along with Field object, to obtain field list information.

Table 7-11 FieldList

| Method | |
|---|--|
| <code>getPresentationSpace()</code> | <p>Obtains the presentation space which created this field list.</p> <p>Returns</p> <p>{PresentationSpace} Parent of this field list instance.</p> |
| <code>findField(position, text, direction)</code> | <p>Returns the field containing the specified text. The search starts from the specified position and proceeds either forward or backward. If the string spans multiple fields, the field containing the starting position is returned. When searching forward the search will not wrap to the top of the screen. When searching backward the search will not wrap to the bottom of the screen.</p> <p>Parameters</p> <p>{Position} Position from which to start the search. See Position object.</p> <p>{String} The text to search for (optional). If not provided, returns the next field to the right of or below the specified position.</p> <p>{Number} direction of the search (optional). Use PresentationSpace.SearchDirection constants for this parameter. For example, <code>PresentationSpace.SearchDirection.FORWARD</code> or <code>PresentationSpace.SearchDirection.BACKWARD</code>. If not provided, searches forward.</p> <p>Returns</p> <p>{Field} containing the string or null if a field meeting the given criteria is not found.</p> <p>Throws</p> <p>{RangeError} If the position is out of range.</p> |
| <code>get(index)</code> | <p>Obtains the field at the given index.</p> <p>Parameters</p> <p>{Number} index into the field list.</p> <p>Returns</p> <p>{Field} located at the specified index.</p> <p>Throws</p> <p>{RangeError} If the index is out of range.</p> |
| <code>isEmpty()</code> | <p>Determines if the field list is empty.</p> <p>Returns</p> <p>{Boolean} True if the list is empty.</p> |

| Method | |
|------------|---|
| size() | Indicates the number of fields in the list. Returns {Number} The field count |
| toString() | Creates a user-friendly description of the field list. Returns {String} User readable rendition of the field list. |

OIA

Operator Information Area (OIA) interface. The OIA object returns values which are defined in the [OIAStatus](#) object.

Table 7-12 OIA

| Method | |
|---------------------|---|
| getStatus () | Returns the set of enabled status flags. See StatusSet . Parameters Returns {StatusSet} Containing the status information. |
| getCommErrorCode () | Returns the current communication error code. Returns {Number} the current communication error code. If one doesn't exist, it will be 0. |
| getProgErrorCode () | Returns the current program error code Returns {Number} the current program error code. If one doesn't exist, it will be 0. |

OIAStatus

Table 7-13 OIAStatus

| OIAStatus | Description |
|------------------|----------------------------------|
| CONTROLLER_READY | Controller ready |
| A_LINE | Online with a non-SNA connection |
| MY_JOB | Connected to a host application |
| OP_SYS | Connected to a SSCP (SNA) |

| OIAStatus | Description |
|------------------|--|
| UNOWNED | Not connected |
| TIME | Keyboard inhibited |
| SYS_LOCK | System lock following AID key |
| COMM_CHECK | Communication check |
| PROG_CHECK | Program check |
| ELSEWHERE | Keystroke invalid at cursor location |
| FN_MINUS | Function not available |
| WHAT_KEY | Keystroke invalid |
| MORE_THAN | Too many characters entered in the field |
| SYM_MINUS | Symbol entered not available |
| INPUT_ERROR | Operator input error (5250 only) |
| DO_NOT_ENTER | Do not enter |
| INSERT | Cursor in insert mode |
| GR_CURSOR | Cursor in graphics mode |
| COMM_ERR_REM | Communications error reminder |
| MSG_WAITING | Message waiting indicator |
| ENCRYPT | Session is encrypted |
| NUM_FIELD | Invalid character in numeric only field |

AutoSignon

Some mainframe hosts have a Digital Certificate Access Server (DCAS). You can request a temporary, one-time pass ticket from DCAS for logging into a host application. Using this object, you can write and configure a macro to run when the session starts and to automatically log you in using the credentials of the currently logged in Reflection ZFE user.

Table 7-14 *AutoSignon*

| Method | |
|------------------------------|---|
| <code>getPassTicket()</code> | <p>Obtains a pass ticket to be used for signing onto a mainframe application. Multiple pass tickets may be requested using different application IDs.</p> <p>Parameters</p> <p>{String} application ID tells the host which application the sign on is for</p> <p>Returns</p> <p>{Promise} fulfilled with the pass ticket key or rejected if the operation fails. The pass ticket obtained from DCAS only works with the current host session and is valid for ten minutes.</p> |
| <code>sendUserName()</code> | <p>Applies the user name contained in the pass ticket to the field at the current cursor location on the current host screen. The user name must be sent before the password. Sending the password first will invalidate the pass ticket, and you will need to get another one.</p> <p>Parameters</p> <p>{String} passTicketKey obtained from getPassTicket</p> <p>Returns</p> <p>{Promise} fulfilled if the user name is successfully sent. Rejected if the operation fails.</p> |
| <code>sendPassword()</code> | <p>Applies the password contained in the pass ticket to the field at the current cursor location on the current host screen. The user name must be sent before the password. Sending the password first will invalidate the pass ticket, and you will need to get another one.</p> <p>Parameters</p> <p>{String} passTicketKey obtained from getPassTicket</p> <p>Returns</p> <p>{Promise} fulfilled if the password is successfully sent. Rejected if the operation fails.</p> |

Position

Represents a row and column on the screen.

Table 7-15 *Position*

| Method | |
|---------------------------------|--|
| <code>Position(row, col)</code> | Creates a new Position instance. |
| Parameters | |
| | <code>{Number}</code> row screen row coordinate |
| | <code>{Number}</code> col screen column coordinate |

PresentationSpace

Use the PresentationSpace object to interact with the terminal screen. Setting and getting the cursor position, sending keys, and reading text are some of the interactions available.

Table 7-16 *PresentationSpace*

METHODS

| | |
|---|---|
| <code>getCursorPosition()</code> | Returns a Position instance representing the current cursor position. An unconnected session has a cursor position of 0,0. |
| | Returns |
| | <code>{Position}</code> current cursor location |
| <code>setCursorPosition(position)</code> | Moves the host cursor to the specified row and column position. For some hosts, such as VT, the host may constrain the movements of the cursor. |
| | Parameters |
| | <code>{Position}</code> Position new cursor position. |
| | Returns |
| | None |
| | Throws |
| | <code>{RangeError}</code> If the position is not valid on the current screen. |
| <code>isCursorVisible()</code> | Tests that the cursor is currently visible in the presentation space. The cursor is considered not visible if the session is not connected. |
| | Returns |
| | <code>{Boolean}</code> True if the cursor is visible. False if the cursor is not visible. |

METHODS

| | |
|---|---|
| <code>sendKeys (keys)</code> | <p>Transmits a text string or ControlKeys to the host at the current cursor position in the presentation space. If the cursor is not in the desired position, then use <code>setCursorPosition</code> function first.</p> <p>The text string can contain any number of characters and ControlKeys</p> <p>For example: <code>"myname" + ControlKey.TAB + "mypass" + ControlKey.ENTER</code> will transmit a user ID, tab to the next field, transmit a password, and then transmit the Enter key.</p> <p>If you need to transmit a square bracket, double the brackets (<code>[[</code> or <code>]]</code>).</p> <p>Parameters</p> <p><code>{String}</code> keys text and/or control keys to transmit</p> |
| <code>getText (position, length)</code> | <p>Returns a string representing a linear area of the presentation space. No new line characters are inserted if row boundaries are encountered.</p> <p>Parameters</p> <p><code>{Position}</code> start position from which to retrieve text</p> <p><code>{Number}</code> length the maximum number of characters to return. If the length parameter causes the last position of the presentation space to be exceeded then only those characters up to the last position will be returned.</p> <p>Returns</p> <p><code>{String}</code> representing a linear area of the presentation space which may be empty if the session is not connected.</p> <p>Throws</p> <p><code>{RangeError}</code> If the position or length are not valid on the current screen.</p> |
| <code>getSize ()</code> | <p>Gets the dimensions of the screen as a <code>Dimension</code> object.</p> <p>Returns</p> <p><code>{Dimension}</code> Containing the number of rows and columns. The screen size is <code>[row:0, col:0]</code> if the session is not connected.</p> |

METHODS

`getDataCells(start, length)`

Returns [DataCell](#) instances where the first member will be for the position specified by the start parameter. The maximum number of DataCell instances in the list is specified by the length parameter.

Parameters

{Position} start the first position on the host screen in which to retrieve DataCell instances. See [Position](#).

{Number} length of the maximum number of DataCell instance to be retrieved. If not specified, returns DataCells from the start position to the end of the screen.

Returns

{DataCell []} Instances which may be empty if the session is not connected. If position is not specified, returns all DataCells. If length is not specified, returns DataCells from the start position to the end of the screen.

Throws

{RangeError} if start or length are out of range.

`getFields()`

Returns a list of the fields in the presentation space. If the host type does not support fields or the current screen is not formatted then the return value will always be an empty list. See [FieldList](#).

Returns

{FieldList} of host defined fields in the presentation space.

Session

The session object is the main entry point for interacting with the host. It contains functions for connecting, disconnecting, and obtaining the PresentationSpace object.

Table 7-17 Session object functions

METHODS

`connect()`

Connects to the configured host. If needed, use `wait.forConnect()` to block macro execution until the session is connected.

Returns

None

`disconnect()`

Disconnects from the configured host. If needed, use `wait.forDisconnect()` to block macro execution until the session is connected.

Returns

None

`isConnected()`

Determines whether the connection to the host is connected.

Returns

{Boolean} true if host connection is established; false if not

METHODS

| | |
|-------------------------------------|---|
| <code>getPresentationSpace()</code> | <p>Provides access to the PresentationSpace instance for this session.</p> <p>Returns</p> <p>{PresentationSpace} instance associated with this session.</p> |
| <code>getDeviceName()</code> | <p>Returns the connected available device name, the configured device name, or null if no device name is configured.</p> <p>The connected device name is the name agreed upon during the connection negotiation process between the host and the terminal. It may be what is specified, or it could possibly be different, if for example a device name pool was specified.</p> <p>Returns</p> <p>{String} The connected device name, the configured device name, or null.</p> |
| <code>getType()</code> | <p>Returns the type of host session. See SessionType.</p> <p>Returns</p> <p>{String} The type of host session.</p> |
| <code>setDeviceName()</code> | <p>Provides a means to modify the device name on a session instance.</p> <p>Parameters</p> <p>{String} name Device name to use when connecting to a host.</p> <p>Throws</p> <p>{Error} If an attempt is made to set the device name while the session is connected.</p> |
| <code>getOIA()</code> | <p>Provides access to the Operator Information Area (OIA) instance for this session.</p> <p>Returns</p> <p>{OIA} Associated with this session</p> |

SessionType

Constants used to identify the type of host to which the connection is being made. See [Session object](#).

Table 7-18 SessionType

| Host Type | Description |
|-----------|---|
| IBM_3270 | Indicates an IBM 3270 terminal session. |
| IBM_5250 | Indicates an IBM 5250 terminal session. |
| VT | Indicates a VT session. |

StatusSet

You can use the StatusSet object to decode the OIA status. The StatusSet object returns values defined in the [OIAStatus](#) object and when used together, you can get status information from the OIA.

Table 7-19 StatusSet

| Method | |
|---|---|
| <code>contains(statusFlag)</code> | Determines if the set contains the specified status flag from OIAStatus constants. Parameters <code>{Number} statusFlag</code> status to check Returns <code>{Boolean}</code> True if the status flag is present in the set. |
| <code>isEmpty()</code> | Determines if the status set is empty. Returns <code>{Boolean}</code> True if the set is empty. |
| <code>size()</code> | Indicates the number of status flags in the set. Returns <code>{Number}</code> The status count |
| <code>toArray()</code> | Converts the internal status set to an array. Returns <code>{Object []}</code> Array of status flags in the set. |
| <code>toString()</code> | Converts the internal status set to a string. Returns <code>{String}</code> Space delimited names of status flags in the set. |
| <code>forEach(callback, thisArg)</code> | Function to iterate over each element in the status set. Parameters <code>{forEachCallback}</code> Callback to perform the specific operation. Called with the name of each status in the set. <code>{Object}</code> <code>thisArg</code> optional pointer to a context object. |
| <code>forEachCallback(string, thisArg)</code> | A user provided callback function where you provide the behavior, to be used as the callback parameter to <code>forEach</code> . Parameters <code>{String} string</code> The name of a status in the status set. <code>{Object} thisArg</code> Optional pointer to a context object |

User Interface

The user interface object provides functions for interacting with the user, prompting for and displaying basic information. The UI object is made automatically available in your macro as the “ui” variable”.

NOTE: Important! All UI functions require the ‘yield’ keyword in front of them. This allows the macro to block execution until the conditions of the UI function have been met.

[parameter] denotes an optional parameter.

Table 7-20 User Interaction

METHODS

`prompt (message, [defaultAnswer],
[mask])`

Prompt the user for information in the user interface,

Parameters

{String} message title to display to the user. Default: blank String.

{String} defaultAnswer to use if user leaves it blank. Default: blank String

{Boolean} mask indicates whether to hide the prompt (as with a password).

Returns

{Promise} Fulfilled when the user closes the dialog window. Returns the users input on “OK” or null on “Cancel”.

`message ([message])`

Display a message in the user interface.

Parameters

{String} message to display to the user. Default: blank String.

Returns

{Promise} Fulfilled when the user closes the message window.

Wait

Use the wait object to wait for a particular session or screen state. For example, you can wait until the cursor is found at a particular location or text is present at a certain location before continuing with the macro execution.

Wait functions are often used in conjunction with asynchronous functions such as `connect()` and `sendKeys()`.

NOTE: All functions take timeouts as an optional parameter and have a default time out value of 10 seconds (10000ms).

Important: All wait functions require the ‘yield’ keyword in front of them. This allows the macro to block execution until the conditions of the wait function are met.

[parameter] denotes an optional parameter.

Table 7-21 *Waiting for the host*

METHODS

| | |
|---|---|
| <code>setDefaultTimeout(timeout)</code> | Sets the default timeout value for all functions. Parameters {Number} default timeout to use for all wait functions in milliseconds. Returns None Throws {RangeError} If the specified timeout is less than zero. |
| <code>forConnect([timeout])</code> | Waits for a connect request to complete. Parameters {Number} in milliseconds. Returns {Promise} Fulfilled if the session is already connected or when connection occurs. Rejected if the wait times out. |
| <code>forDisconnect([timeout])</code> | Waits for a disconnect request to complete. Parameters {Number} timeout in milliseconds. Returns {Promise} Fulfilled if the session is already disconnected or when it finally disconnects. Rejected if the wait times out. |
| <code>forFixedTime([timeout])</code> | Waits unconditionally for fixed time. Time is in milliseconds (ms) Parameters {Number} timeout in milliseconds. Returns {Promise} Fulfilled after time elapses |
| <code>forCursor(position, [timeout])</code> | Waits for the cursor to arrive at the specified position. Parameters {Position} The position specifying the row and column, {Number} timeout in milliseconds Returns {Promise} Fulfilled if the cursor is already located or when it is finally located. Rejected if the wait times out. |

METHODS

```
forText(string, position,  
[timeout])
```

Wait for text located at a specific position on the screen

Parameters

{String} text to expect

{Position} position specifying the row and column

{Number} timeout in milliseconds

Returns

{Promise} Fulfilled if the text is already at the specified position or whenever it is located. Rejected if the wait times out.

Throws

{rangeError} if the position is not valid.

```
forHostPrompt(string,  
column, [timeout])
```

Waits for a command prompt located at a particular column on the screen.

Parameters

{String} text prompt to expect

{Number} column where cursor is expected

{Number} timeout in milliseconds.

Returns

{Promise} Fulfilled if the conditions are already met or when the conditions are finally met. Rejected if the wait times out.

Throws

{rangeError} if the column is out of range.

Sample Macros

To help you create successful macros that take advantage of all the capabilities of the Macro Editor and Reflection ZFE, these samples are available as a starting point.

- ◆ [“Basic Host Interaction” on page 80](#)
- ◆ [“User Interaction” on page 81](#)
- ◆ [“Paging Through Data” on page 83](#)
- ◆ [“Invoking a Web Service” on page 84](#)
- ◆ [“Working with DataCells and Attributes” on page 86](#)
- ◆ [“Using Fields and Field Lists” on page 88](#)
- ◆ [“Automatic Sign-On Macro for Mainframes” on page 89](#)

Basic Host Interaction

This sample illustrates basic host interaction, including:

- ♦ Sending data to the host
- ♦ Waiting for screens to display
- ♦ Using the `yield` keyword to wait for asynchronous functions
- ♦ Reading text from the screen
- ♦ Displaying basic information to the user
- ♦ Handling error basics

All macros have the following objects available by default:

1. **session** - Main entry point for access to the host. Can connect, disconnect and provides access to the `PresentationSpace`.

The `PresentationSpace` object obtained from the session represents the screen and provides many common capabilities such as getting and setting the cursor location, sending data to the host and reading from the screen.

2. **wait** - Provides a simple way to wait for various host states before continuing to send more data or read from the screen.
3. **UI** - Provides basic user interface capabilities. Display data to the user or prompt them for information.

```
// Create a new macro function
var macro = createMacro(function* () {
  'use strict';

  // All macros have the following objects available by default:
  // 1. session - Main entry point for access to the host. Can connect, disconnect
and provides access to the PresentationSpace.
  //   The PresentationSpace object obtained from the session represents the
screen and provides many common capabilities such as getting and setting the
  //   cursor location, sending data to the host and reading from the screen.
  // 2. wait - Provides a simple way to wait for various host states before
continuing to send more data or read from the screen.
  // 3. ui - Provides basic User Interaction capabilities. Display data to the user
or prompt them for information.

  // Declare a variable for reading and displaying some screen data.
  // As a best practice all variables should be declared near the top of a function.
  var numberOfAccounts = 0;

  // Start by obtaining the PresentationSpace object, which provides many common
screen operations.
  var ps = session.getPresentationSpace();

  try {
    // Can set and get the cursor location
    ps.setCursorPosition(new Position(24, 2));

    // Use the sendKeys function to send characters to the host
    ps.sendKeys('cics');

    // SendKeys is also used to send host keys such as PA and PF keys.
    // See "Control Keys" in the documentation for all available options
```



```

    ps.sendKeys(ControlKey.ENTER);

    // Wait for the cursor to be at the correct position.
    // The wait object provides various functions for waiting for certain states to
    occur
    // so that you can proceed to either send more keys or read data from the
    screen.
    yield wait.forCursor(new Position(24, 2));

    // You can mix characters and control keys in one sendKeys call.
    ps.sendKeys('data' + ControlKey.TAB + ControlKey.TAB + 'more data' +
    ControlKey.ENTER);

    // The "yield" keyword must be used in front of all "wait" and "ui" function
    calls.
    // It tells the browser to pause execution of the macro until the
    // (asynchronous) wait function returns. Consult the documentation for which
    functions
    // require the yield keyword.
    yield wait.forCursor(new Position(10, 26));
    ps.sendKeys('accounts' + ControlKey.ENTER);

    // Can also wait for text to appear at certain areas on the screen
    yield wait.forText('ACCOUNTS', new Position(3, 36));
    ps.sendKeys('1' + ControlKey.ENTER);

    // All wait functions will timeout if the criteria is not met within a time
    limit.
    // Can increase timeouts with an optional parameter in the wait functions (in
    milliseconds)
    // All timeouts are specified in milliseconds and the default value is 10
    seconds (10000ms).
    yield wait.forCursor(new Position(1, 1), 15000);
    ps.sendKeys('A' + ControlKey.ENTER);

    // PS provides the getText function for reading text from the screen
    numberOfAccounts = ps.getText(new Position(12, 3), 5);

    // Use the ui object to display some data from the screen
    ui.message('Number of active accounts: ' + numberOfAccounts);

    // The try / catch allows all errors to be caught and reported in a central
    location
    } catch (error) {
        // Again we use the ui object to display a message that an error occurred
        yield ui.message('Error: ' + error.message);
    }
    //End Generated Macro
});

// Run the macro and return the results to the Macro Runner
// The return statement is required as the ZFE application leverages
// this to know if the macro succeeded and when it is finished
return macro();

```

User Interaction

This sample illustrates how to use the provided API methods to prompt the user for input or alert them with a message.

```

var macro = createMacro(function*(){
    'use strict';

    // The "ui" object provides functions for prompting the user for information and
    displaying information

    // Declare variables for later use
    var username;
    var password;
    var flavor;
    var scoops;

    //Begin Generated Macro
    var ps = session.getPresentationSpace();

    try {
        // Prompt the user to enter their name and store it in a variable.
        // Note that 'yield' keyword is needed to block execution while waiting for the
user input.
        username = yield ui.prompt('Please enter your username');

        // Prompt the user to enter a value with a default provided to them.
        flavor = yield ui.prompt('What is your favorite flavor of ice cream?',
'Chocolate');

        // Prompt the user to enter private information by using the 'mask' option and
the input field will be masked as they type.
        // If a parameter is not used, 'null' can be used to specify that it isn't to
be used.
        // Here we illustrate that by specifying that we don't need to show a default
value .
        password = yield ui.prompt('Please enter your password', null, true);

        // The prompt function returns null if the user clicks the 'Cancel' button
instead of the 'OK' button.
        // One way to handle that case is to wrap the call in a try/catch block.
        scoops = yield ui.prompt('How many scoops would you like?');
        if (scoops === null) {
            // This will exit the macro.
            return;
            // Alternatively could throw an Error and have it be caught in the "catch"
below
        }
        // Use the collected values to order our ice cream
        ps.sendKeys(username + ControlKey.TAB + password + ControlKey.ENTER);
        yield wait.forCursor(new Position(5, 1));
        ps.sendKeys(flavor + ControlKey.TAB + scoops + ControlKey.ENTER);
    }
}

```

```

    // Display a message to the user. Using the 'yield' keyword in front of the
    call will block
    // further execution of the macro until the user clicks the 'OK' button.
    yield ui.message('Order successful. Enjoy your ' + scoops + ' scoops of ' +
    flavor + ' ice cream ' + username + '!');
    } catch (error) {
    // Here we use the ui object to display a message that an error occurred
    yield ui.message(error.message);
    }
    //End Generated Macro

});

return macro();

```

Paging Through Data

This sample illustrates how to page through a variable number of screens and process the data on each screen.

```

// Create a new macro function.
var macro = createMacro(function*(){
    'use strict';

    // Create variable(s) for later use
    var password;
    var accountNumber;
    var transactionCount = 0;
    var row = 0;

    // Obtain a reference to the PresentationSpace object.
    var ps = session.getPresentationSpace();

    try {
    // Enter Username and Password to log on to the application.
    yield wait.forCursor(new Position(19, 48));
    ps.sendKeys('bjones' + ControlKey.TAB);

    yield wait.forCursor(new Position(20, 48));
    password = yield ui.prompt('Password:', null, true);
    ps.sendKeys(password);
    ps.sendKeys(ControlKey.ENTER);

    // Enter an application command.
    yield wait.forCursor(new Position(20, 38));
    ps.sendKeys('4');
    ps.sendKeys(ControlKey.ENTER);

    // Going to list transactions for an account.
    yield wait.forCursor(new Position(13, 25));
    ps.sendKeys('2');
    // Input an account number. Hard coded here for simplicity.
    yield wait.forCursor(new Position(15, 25));
    accountNumber = yield ui.prompt('Account Number:', '167439459');
    ps.sendKeys(accountNumber);
    ps.sendKeys(ControlKey.ENTER);

    // Wait until on account profile screen
    yield wait.forText('ACCOUNT PROFILE', new Position(3, 33));

    // Search for text that indicates the last page of record has been reached
    while (ps.getText(new Position(22, 12), 9) !== 'LAST PAGE') {

    // While the last page of record has not been reached, go to the next page of records.
    ps.sendKeys(ControlKey.PF2);
    yield wait.forCursor(new Position(1, 1));

    // If the cursor position does not change between record screens, and there is no text

```

```

// on the screen you can check to confirm a screen is updated, you may wait for a
// fixed time period after an aid key is sent for the screen to settle.
// For example:
// yield wait.forFixedTime(1000);

// For each of the rows, increment the count variable if it contains data.
for (row = 5; row <= 21; row++) {

    // There are 2 columns on the screen. Check data on column 1.
    // In this example we know that if there is a space at a particular
    // position then there is a transaction.
    if (ps.getText(new Position(row, 8), 1) !== ' ') {
        transactionCount++;
    }
    // Check data on column 2.
    if (ps.getText(new Position(row, 49), 1) !== ' ') {
        transactionCount++;
    }
}
}
// Check data on column 2.
if (ps.getText(new Position(row, 49), 1) !== ' ') {
    transactionCount++;
}
}

// After going through all record pages, display the number of records in a message box.
yield ui.message('There are ' + transactionCount + ' records found for account ' +
accountNumber + '.');

// Log out of the application
ps.sendKeys(ControlKey.PF13);
ps.sendKeys(ControlKey.PF12);

// The try / catch allows all errors to be caught and reported in a central location
} catch (error) {
    // Here we use the ui object to display a message that an error occurred
    yield ui.message(error.message);
}
});

// Here we run the macro and return the results to the Macro Runner
// The return statement is required as the ZFE application leverages
// this to know if the macro succeeded
return macro();

```

Invoking a Web Service

This sample illustrates how to make an AJAX / REST call directly from a macro to a web service. You can integrate data from your host application into the web service call or from the web service into your host application.

In this example, we are calling the Verastream Host Integrator (VHI) CICSAcctsDemo REST service. However, you can easily adapt the code to call any web service. You are not limited to VHI.

In the example the call goes through a proxy configured in the session server (shown below) to avoid a "Same Origin Policy" complication. If you are using a web service that supports [Cross-origin Resource Sharing \(CORS\)](#) and are using a modern browser, the proxy is unnecessary.

Since the jQuery library is available in macros, so you may use the \$.post() function directly to invoke REST services.

This example also demonstrates how to wrap a jQuery REST call in a new Promise. The promise returned from the custom function below allows "yield" to be used in the main macro code. This allows the main macro execution to wait until the service call is complete before continuing.

```

var macro = createMacro(function*() {
  'use strict';

  // Create a few variables for later user
  var username;
  var password;
  var accountNumber;
  var accountDetails;

  // Create a function that will make an AJAX / REST call to a VHI Web Service.
  // Could be adjusted to call any web service, not just VHI.
  // If not using CORS, the request will likely need to pass through a
  // proxy on the session server. See sample notes for more information.
  /**
   * Hand-coded helper function to encapsulate AJAX / REST parameters, invoke the
   * REST service and return the results inside a Promise.
   * @param {Number} acctNum to send to the REST query.
   * @param {String} username to access the REST service.
   * @param {String} password to access the REST service.
   * @return {Promise} containing $.post() results that are compatible with yield.
   */
  var getAccountDetails = function (acctNum, username, password) {
    var url = "proxy1/model/CICSAcctsDemo/GetAccountDetail";
    var args = {"filters": {"AcctNum": acctNum}, "envVars": {"Username": username,
"Password": password}};

    // Wrap a jQuery AJAX / HTTP POST call in a new Promise.
    // The promise being returned here allows the macro to yield / wait
    // for its completion.
    return Promise.resolve($.post(url, JSON.stringify(args)))
      .catch(function (error) {
        // Map errors that happen in the jQuery call to our Promise.
        throw new Error('REST API Error: ' + error.statusText);
      });
  };

  // Begin Generated Macro
  var ps = session.getPresentationSpace();
  try {
    // Could interact with the host here, log into a host app, etc...
    // Gather username and password
    username = yield ui.prompt('Username:');
    password = yield ui.prompt('Password:', null, true);
    accountNumber = yield ui.prompt('Account Number:');
    if (!username || !password || !accountNumber) {
      throw new Error('Username or password not specified');
    }

    // Invoke external REST service, and yields / waits for the call to complete.
    accountDetails = yield getAccountDetails(accountNumber, username, password);

    // We now have the data from our external service.
    // Can integrate the data into our local host app or simply display it to the user.
    // For this sample we simply display the resulting account details.
    if (accountDetails.result && accountDetails.result.length > 0) {
      yield ui.message(accountDetails.result[0].FirstName + ' $' +
accountDetails.result[0].AcctBalance);
    } else {
      yield ui.message('No records found for account: ' + accountNumber);
    }
  } catch (error) {
    // If an error occurred during the AJAX / REST call
    // or username / password gathering we will end up here.
    yield ui.message(error.message);
  }
});

// Run our macro
return macro();

```

Cross Origin Scripting Proxy Support

If you have web services that do not support CORS, AJAX/REST calls will fail if they attempt to access a different server than the one where the ZFE application originated. This is a browser security feature.

The Reflection ZFE server provides a way explicitly to proxy to trusted remote servers.

- ◆ Open `..\ReflectionZFE\sessionserver\webapps\zfe\WEB-INF\web.xml` in your ZFE deployment.
- ◆ Modify the file as shown:

```
<!--
Example of how to proxy 3rd party services that do not support CORS.
The following configuration would allow you to invoke remote VHI REST
services via local URLs (e.g. an HTTP POST to
"proxyl/model/CICSAcctsDemo/GetAccountDetail" would be routed to
"http://remote-vhi-server:9680/vhi-rs/model/CICSAcctsDemo/GetAccountDetail").
-->
<servlet>
  <servlet-name>vhi-rs-proxyl</servlet-name>
  <servlet-class>org.eclipse.jetty.proxy.ProxyServlet$Transparent</servlet-class>
  <init-param>
    <param-name>proxyTo</param-name>
    <param-value>http://remote-vhi-server:9680/vhi-rs/</param-value>
  </init-param>
  <init-param>
    <param-name>prefix</param-name>
    <param-value>/proxyl</param-value>
  </init-param>
  <async-supported>true</async-supported>
</servlet>
<servlet-mapping>
  <servlet-name>vhi-rs-proxyl</servlet-name>
  <url-pattern>/proxyl/*</url-pattern>
</servlet-mapping>
```

- ◆ Uncomment the servlet and servlet-mappings sections.
- ◆ Change `http://remote-vhi-server:9680/vhi-rs` to the actual URL of your target REST server. You can also rename the url-pattern.
- ◆ Keep in mind that even when a REST server supports CORS headers, some older browsers may not, so this example may still be relevant.

TIP: Your customized `web.xml` file may be replaced whenever you redeploy Reflection ZFE. Always back up your files.

Working with DataCells and Attributes

This macro illustrates how to use `DataCells` and `AttributeSet` to inspect a given row/column on the screen for text and attributes. In this sample you can see:

- ◆ How to get a collection of `DataCells` for a given position and length.
- ◆ How to iterate through `DataCells` to build up a text string
- ◆ How, for comparison, you can also do a similar thing using `getText()`.
- ◆ And finally, how to work with attributes, get a string listing, or determine whether specific ones are set at a given screen location.

```

var macro = createMacro(function*() {
    'use strict';

    // Obtain the PresentationSpace for interacting with the host
    var ps = session.getPresentationSpace();

    // Declare variables for later use
    var cells;
    var text;
    var attrs;

    // Set the default timeout for "wait" functions
    wait.setDefaultTimeout(10000);

    // Sample macro for working with DataCells and Attributes
    try {
        yield wait.forCursor(new Position(24, 2));

        // Get DataCells from the presentation space.
        // Row 19, col 3 is the prompt, 35 characters long
        // "Choose from the following commands:"
        cells = ps.getDataCells({row:19, col:3}, 35);
        text = '';

        // You can display text using getText
        yield ui.message("Screen text: " + ps.getText({row:19, col:3}, 35));

        // Or you can assemble the text from the DataCells at each position
        for(var index = 0; index < cells.length; index++) {
            text = text.concat(cells[index].getChar());
        }
        // And display the text
        yield ui.message("Cells text: " + text);

        // Get the attributes for the first DataCell (cell[0])
        attrs = cells[0].getAttributes();

        // Display whether we have any attributes on the data cell
        yield ui.message("Attribute set is empty: " + attrs.isEmpty());

        // Display how many attributes are set
        yield ui.message("Number of attributes: " + attrs.size());

        // Display which attributes are set
        yield ui.message("Attributes: " + attrs.toString());

        // Now display whether the high intensity attribute is set
        yield ui.message("Is high intensity: " +
            attrs.contains(Attribute.HIGH_INTENSITY));

        // Now display whether the underline attribute is set
        yield ui.message("Is underline: " +
            attrs.contains(Attribute.UNDERLINE));

        // Now display whether alphanumeric, intensified and pen-detectable attributes are
set
        yield ui.message("Is alphanumeric, intensified and pen-detectable: " +
            attrs.containsAll([Attribute.ALPHA_NUMERIC, Attribute.HIGH_INTENSITY,
Attribute.PEN_DETECTABLE]));

        // Now display whether underline, intensified and pen-detectable attributes are set
        yield ui.message("Is underline, intensified and pen-detectable: " +
            attrs.containsAll([Attribute.UNDERLINE, Attribute.HIGH_INTENSITY,
Attribute.PEN_DETECTABLE]));
    } catch (error) {
        yield ui.message(error);
    }
} //End Generated Macro
});

// Run the macro
return macro();

```

Using Fields and Field Lists

This macro sample illustrates how to use common functions to interact with the fields in the Macro API. For example, how to get field text, view field information, and how to use `field.setText` as an alternative to `sendKeys` to interact with the host.

NOTE: Due to browser considerations `ui.message` collapses strings of spaces down to a single space. The spaces are preserved in the actual JavaScript.

```
var macro = createMacro(function*() {
  'use strict';

  // Obtain the PresentationSpace for interacting with the host
  var ps = session.getPresentationSpace();

  // Declare variables for later use
  var fields;
  var field;
  var searchString = 'z/vM';

  // Set the default timeout for "wait" functions
  wait.setDefaultTimeout(10000);

  // Sample macro for working with FieldList and Fields
  try {
    yield wait.forCursor(new Position(24, 2));

    // Get the field list.
    fields = ps.getFields();

    // Run through the entire list of fields and display the field info.
    for(var index = 0; index < fields.size(); index++) {
      field = fields.get(index);

      yield ui.message("Field " + index + " info: " + field.toString());
    }

    yield ui.message("Now, find a field containing the text '" + searchString + "'");
    field = fields.findField(new Position(1, 1), searchString);

    if(field !== null) {
      yield ui.message("Found field info: " + field.toString());
      yield ui.message("Found field foreground is green? " + (Color.GREEN ===
field.getForegroundColor()));
      yield ui.message("Found field background is default? " + (Color.BLANK_UNSPECIFIED
=== field.getBackgroundColor()));
    }

    // Now, find command field and modify it.
    field = fields.findField(new Position(23, 80));
    if(field !== null) {
      field.setText("cics");
    }

    yield ui.message("Click to send 'cics' to host.");
    ps.sendKeys(ControlKey.ENTER);

    // Wait for new screen; get new fields.
    yield wait.forCursor(new Position(10, 26));
    fields = ps.getFields();

    // Find user field and set it.
    field = fields.findField(new Position(10, 24));
    if(field !== null) {
      field.setText("bvtst01");
    }

    // Find password field and set it.
    field = fields.findField(new Position(11, 24));
```



```

        if(field !== null) {
            field.setText("milk");
        }

        yield ui.message("Click to send login to host.");
        ps.sendKeys(ControlKey.ENTER);

        // Wait for new screen; get new fields.
        yield wait.forCursor(new Position(1, 1));
        fields = ps.getFields();

        // Find command field and set logoff command.
        field = fields.findField(new Position(24, 45));
        if(field !== null) {
            field.setText("cesf logoff");
        }

        yield ui.message("Click to send logoff to host.");
        ps.sendKeys(ControlKey.ENTER);

    } catch (error) {
        yield ui.message(error);
    }
    //End Generated Macro
});

// Run the macro
return macro();

```

Automatic Sign-On Macro for Mainframes

In this example the `Autosignon` object is used to create a macro that uses the credentials associated with a user to obtain a pass ticket from the Digital Certificate Access Server (DCAS).

```

var macro = createMacro(function*() {
    'use strict';

    // Obtain the PresentationSpace for interacting with the host
    var ps = session.getPresentationSpace();

    // Variable for login pass ticket
    var passTicket;

    // Login application ID
    var appId = 'CICSV41A';

    // Set the default timeout for "wait" functions
    wait.setDefaultTimeout(10000);

    // Begin Generated Macro
    try {
        yield wait.forCursor(new Position(24, 2));

        // Obtain a pass ticket from DCAS.
        passTicket = yield autoSignon.getPassTicket(appId);

        ps.sendKeys('cics');
        ps.sendKeys(ControlKey.ENTER);

        yield wait.forCursor(new Position(10, 26));

        // Replace generated username with sendUserName(passTicket) ...
        yield autoSignon.sendUserName(passTicket);

        // ps.sendKeys('bvtst01' + ControlKey.TAB + ControlKey.TAB);
        ps.sendKeys(ControlKey.TAB + ControlKey.TAB);

        yield wait.forCursor(new Position(11, 26));

        // Replace generated password with sendPassword(passTicket) ...
        yield autoSignon.sendPassword(passTicket);
    }
});

```

```
// var userInput3 = yield ui.prompt('Password:', '', true);
// if (userInput3 === null) {
//     // throw new Error('Password not provided');
// }
// ps.sendKeys(userInput3);
ps.sendKeys(ControlKey.ENTER);

yield wait.forCursor(new Position(1, 1));
yield ui.message('Logged in. Log me off.');
```

```
ps.sendKeys('cesf logoff');
ps.sendKeys(ControlKey.ENTER);
} catch (error) {
    yield ui.message(error);
}
//End Generated Macro
});

// Run the macro
return macro();
```

8 File and Data Transfer

Mainframe file transfer copies files between your computer and a 3270 mainframe using the IND\$FILE host program. As an administrator, you can choose to transfer files using either TSO or CMS host filing systems.

Mainframe file transfer


Before you can transfer files, you (as the administrator) must enable the transfer option for the current session. This is done on the Connection panel.

From the **Host File System** drop down list, select which IBM 3270 operating environment the host is running. ZFE supports TSO (Time Sharing Option) and CMS (Conversational Monitor System). The default selection is None.

There is support for ASCII or binary transfers.

Transferring files

You must be connected to the host to transfer files for the current 3270 session.

- 1 Verify that the host is in a 'ready' state to accept the IND\$FILE command.
- 2 From the left panel, click the **IND\$FILE** icon .
- 3 The File Transfer dialog box displays, containing a list of host files and directories that are available to transfer. Directories and files are indicated by an icon when you select the file.
- 4 Select the transfer method. The options are:
 - ♦ Binary
Use for program files and other types of files that should not be translated, such as files that have already been formatted for a particular type of printer or files with application-specific formatting. Binary files contain non-printable characters; using this method, a file is not converted or translated during the transfer.
 - ♦ ASCII
Use to transfer text files with no special formatting. ASCII files on the PC are translated to the EBCDIC character set on the host and host text files are converted from EBCDIC to ASCII when they are downloaded.

You can refresh the file list at any time by clicking the **Refresh** icon in the upper right corner of the File Transfer dialog box.

Downloading files

- 1 From the list, select the file or directory to initiate the transfer. You can choose to save or open the files in the format you selected in step 3.
- 2 If necessary, you can cancel the transfer from the transfer progress panel.

Uploading files

NOTE: IBM mainframe computer systems impose certain naming conventions for files. For detailed information on naming requirements, see the [IBM documentation](#).

There are two methods for uploading files:

- 1 From the File Transfer dialog box, click **Upload**.
- 2 Choose the file you want to upload from the Browse window. The uploaded file displays in the list in alphabetical order.

Or:

- 1 Drag the file you want to upload from its location to the File Transfer dialog box.
- 2 Click **Refresh** to verify the file was successfully uploaded.

If you cancel the upload process before a file has been completely transferred, a partial file will be left behind on the host.

Troubleshooting your file transfers

Occasionally you might encounter errors when attempting a file transfer. These errors may be mainframe issues or, because you are transferring files using a browser; browser settings can be a source of unplanned errors.

For example, a browser may prompt you to action, such as a Save As prompt, despite the fact that the file transfer failed. This issue can be resolved by simply changing the browser setting.

For host-specific errors, see [IBM File Transfer Error Messages](#).

9 Logging

Reflection ZFE uses Log4J 1.2 to implement logging. Log4J has its own configuration file and documentation. The configuration file, located in `ReflectionZFE/sessionserver/conf/log4j.xml`, has a number of logging levels configured for output, and contains comments about the type of information that you can gather by changing logging levels.

For more information, see the [Log4J documentation \(http://logging.apache.org/log4j/1.2\)](http://logging.apache.org/log4j/1.2).

The default logging (log4j) configurations are:

- ◆ Log file output is saved to `logs/server.log`
- ◆ In addition to logging to the `server.log` file, all console output is captured by the Reflection ZFE session server and stored in a file on disk.
- ◆ The configuration for how the console output is stored on file in `ReflectionZFE/sessionserver/conf/container.conf`.

The file storage configuration properties include, but are not limited to the following (there are comments in `container.conf` that provide more information):

- ◆ `wrapper.logfile` - the location of the captured log file (default is `.../logs/server.log`)
- ◆ `wrapper.logfile.rollmode` - the mechanism in which the existing log file is stored as a backup and a new file is created (default is rolling over when the log file reaches a certain size and storing the rolled over log file with a roll number modifier)
- ◆ `wrapper.logfile.maxsize` - the maximum size the log file can reach before it is rolled over (default is 10MB)
- ◆ `wrapper.logfile.maxfiles` - the maximum number of rolled log files to keep on disk (default is 10)
- ◆ There are various types of logging levels you can use to produce different types of information. Log4j supports the following levels (these definitions are taken from the Log4j documentation where you can find more detailed information):
 - ◆ Trace - this level designates finer-grained informational events than Debug
 - ◆ Debug - this level designates fine-grained informational events that are most useful to debug an application.
 - ◆ Info - This level designates informational messages that highlight the progress of the application at coarse-grained level.
 - ◆ Warn - This level designates potentially harmful situations.
 - ◆ Error - This level designates error events that might still allow the application to continue running.
 - ◆ Fatal - This level designates very severe error events that will presumably lead the application to terminate.

10 Troubleshooting

This section documents miscellaneous known issues and work around tips for Reflection ZFE. For a more detailed list of known issues in Reflection ZFE, see [Technical Note 2853](#).

Installation Connection Issues

Certificate authentication or other connection issues are well documented in [Technical Note 2838](#).

SSL/TLS Error Message

- ◆ **(ECL1011) Error connecting to host: Connection to host failed.**

This error can display in a number of situations that are not simply due to a connection failure.

- ◆ You may see this error if an SSL/TLS connection failed due to the lack of a trusted certificate in the MSS trust store.
- ◆ This error displays when a TLS handshake failure occurs when you use SSL/TLS to connect to or from a plain text host.

Displaying the Euro character

- ◆ If the EURO character does not display correctly on the terminal screen, talk to your system administrator to make sure the host character set for the session is setup correctly. By default, Reflection ZFE uses a character set which does not support the Euro character (€). To display the Euro character, change the character set to one that supports the Euro character.

