Contents

About Reflection ZFE 7

1 Release Notes 9
What’s New ............................................................... 9
Known Issues ........................................................... 9
Installing the Product .................................................. 9
Contacting Micro Focus ............................................. 10
Legal Notice ............................................................ 10

2 Installing Reflection ZFE 11
Before you install ...................................................... 11
System requirements .................................................. 12
  Supported web browsers ........................................... 12
  Session server operating systems ................................ 12
Preparing to install .................................................... 13
  Ports used by Reflection ZFE ....................................... 13
Upgrading from previous installations ............................ 13
Troubleshooting the installation .................................... 14
  Connecting using HTTP ............................................. 14
  Other known issues ................................................ 15

3 Setting Post Installation Options 17
  How to Adjust Session Timeout Values .......................... 17
  How to Set Up the Terminal ID Manager for Reflection ZFE ... 17
  How to Set Up Metering for Reflection ZFE .................... 18
  How to Start and Stop Services Automatically ................ 19
  How to Change Ports ............................................... 20
  How to set up Automated Single Sign-On for Mainframe .... 21

4 Making Secure Connections 23
  Securing the Web Browser to Reflection ZFE Session Server .. 24
  Securing Reflection ZFE Session Server and Management Component to MSS .... 25
  Securing Reflection ZFE Session Server to the Host ............ 26
    How to install unlimited strength policy jars ................. 26
    How to configure the keystore location on the Reflection ZFE Server .......... 26
    How to configure the keystore in MSS ........................ 26
    How to configure a Reflection ZFE terminal session .......... 27
  Configuring X.509 Authentication ................................ 28
  Configuring Single Sign-on through IIS ........................ 29
  Enabling FIPS Level Security .................................... 30
    To enable FIPS mode: ........................................... 30
  Enabling SSL ....................................................... 31
5 Configuring Connections
Connecting to the host .......................................................... 33
  Providing access to the session ........................................... 33
Common connection settings .................................................. 34
3270 and 5250 connection settings ....................................... 35
VT connection settings .......................................................... 38
UTS connection settings ....................................................... 39
How to test Terminal ID Manager criteria ............................... 40

6 Configuring Session Options
Configure Display Settings ...................................................... 41
  Customize host screen colors ............................................ 41
  Configure hotspots .......................................................... 42
  Configure screen dimensions for VT and UTS hosts ............... 43
  Set cursor options .......................................................... 43
  Set VT scrollbar buffer options ........................................ 44
  Set keyboard options ...................................................... 45
  Configure terminal settings .............................................. 45
  Set other display options ............................................... 46
Map keys .............................................................................. 47
  3270, 5250, VT, and UTS Keyboard Mapping ....................... 48
Configure macros for your users .......................................... 55
Transfer files ......................................................................... 56
  IND$FILE ........................................................................ 56
  FTP ............................................................................... 56
Specify copy and paste options ............................................ 58
Set User Preference Rules ..................................................... 59
Customize your session ......................................................... 59
  Use Plus to customize screens ........................................... 60
  Use server side events ...................................................... 61

7 Using Sessions
Managing your sessions .......................................................... 63
  Using Quick Keys ............................................................. 63
  Copying and Pasting ........................................................ 65
  Recording macros for the session ....................................... 65
  Logging out ..................................................................... 66

8 Printing
3270 Host Printing ............................................................... 67

9 Creating Macros
How do I? ............................................................................. 69
  Record a macro ............................................................... 70
  Edit a macro ................................................................... 70
  Run a macro .................................................................... 71
  Stop a macro ................................................................... 71
  Delete a macro ................................................................ 71
  View my macros ............................................................ 72
  Debug my macros .......................................................... 73
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration options</td>
<td>128</td>
</tr>
<tr>
<td>Troubleshooting the configuration</td>
<td>129</td>
</tr>
<tr>
<td>Known Issues</td>
<td>129</td>
</tr>
<tr>
<td>Browser issues</td>
<td>129</td>
</tr>
<tr>
<td>Security issues</td>
<td>130</td>
</tr>
<tr>
<td>Host specific issues</td>
<td>130</td>
</tr>
</tbody>
</table>
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 Release Notes.

- 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. After creating and assigning sessions for your users, you launch the session in the Reflection ZFE web client.

• 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. See Configuring Connections.

• Your users have access to their assigned sessions through a URL you provide (for example, https://<sessionserver>:7443/zfe). They can select which session to open from the list of available sessions you have configured for them. See Providing access to the session for more information.

• Depending on the User Preferences you set, your users can record their own macros, customize their color palette, and personalize their sessions in other ways.

### 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:<port>/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.
Release Notes

Reflection ZFE version 2.1.4 released September 2017. These release notes list the features and known issues in this release and information on how to obtain the product. Reflection ZFE provides terminal emulation for 3270, 5250, VT, and UTS host types, while requiring only an HTML 5-capable browser.

What’s New

- Additional terminal settings for VT host types.
- Anonymously authenticated users can access their previous user configurations, user macros, and other personalizations.
- Users can access particular sessions from URLs created using “name” and “singleSession” parameters. This means that users can have a direct link from their company portal to a single named Reflection ZFE session.
- Support for FTP file transfer protocol.

Known Issues

Micro Focus Technical Support is always available to help you with any issues you may encounter in Reflection ZFE.

Unresolved issues from previous releases are listed in Technical References under Known Issues.

NOTE: WebSocket Communication

Reflection ZFE terminal emulation uses WebSocket technology for communication between a web browser and the Reflection ZFE session server. WebSocket is a feature of the HTML5 specification, and is supported by current releases of all widely available web browser software. If your organization has modified router or network configurations to disable WebSocket communications on your network, you cannot use Reflection ZFE terminal sessions.

Some anti-virus software might prevent WebSocket connections, especially when ports 80 or 8080 are used. If you think your anti-virus software may be preventing WebSockets, first try a different port. For troubleshooting, see http://websocketstest.com.

Installing the Product

Read Installing Reflection ZFE for specific system and installation requirements and helpful tips.
Contacting Micro Focus

For specific product issues, contact Micro Focus Support (https://www.microfocus.com/support-and-services/).

Additional technical information or advice is available from several sources:

- Product documentation, Knowledge Base articles and videos - see Support for Reflection ZFE.
- The Micro Focus Community pages – see Micro Focus Communities.

Legal Notice

For information about legal notices, trademarks, disclaimers, warranties, export and other use restrictions, U.S. Government rights, patent policy, and FIPS compliance, see https://www.microfocus.com/about/legal/.

Copyright © 2017 Attachmate Corporation, a Micro Focus company. All rights reserved.

The only warranties for this product and any associated updates or services are those that may be described in express warranty statements accompanying the product or in an applicable license agreement you have entered into. Nothing in this document should be construed as creating any warranty for a product, updates, or services. The information contained in this document is subject to change without notice and is provided "AS IS" without any express or implied warranties or conditions. Micro Focus shall not be liable for any technical or other errors or omissions in this document. Please see the product’s applicable end user license agreement for details regarding the license terms and conditions, warranties, and limitations of liability.

Any links to third-party websites take you outside Micro Focus websites, and Micro Focus has no control over and is not responsible for information on third party sites.
Installing Reflection ZFE

You can install Reflection ZFE from the Micro Focus Download site.

In this section

- Before you install
- System requirements
- Preparing to install
- Upgrading from previous installations
- Troubleshooting the installation

Related Topics

Setting Post Installation Options
Making Secure Connections

Before you install

These prerequisites need to be met before you can successfully install Reflection ZFE.

- Host Access Management and Security Server

  Reflection ZFE requires Host Access Management and Security Server (MSS) for session management: you can use an existing MSS installation or a simple first-time install. The Windows install program will install MSS, the ZFE session server, and documentation to a single machine. Different components can reside on different machines.

  You will be asked for the user name and password for the Management and Security Server used by Reflection ZFE. It is a good idea to have those credentials in hand before you start installation.

  MSS uses activation files (activation.jaw) to enable product functionality. The Reflection ZFE install program contains the needed activation file. You will need to provide an activation file if you intend to use an already installed or remote MSS server that has not been activated for use with Reflection ZFE. UTS support and the Terminal ID Manager also require separate activation files. You can read all about MSS activation files in the Host Access Management and Security Server Installation Guide.

- Reflection ZFE and Java

  Reflection ZFE requires a Java JDK version 8 or higher and MSS requires a Java JRE version 8 or higher. This Java requirement is met during installation, except for these platform exceptions:

  - For systems, such as AIX or Linux on System Z that require an IBM JDK, you can use the "nojdk" installer media, which does not include a bundled JDK.

    To use the nojdk media option:
The installation must be able to locate a Java executable to start. If a Java executable cannot be found by the installer, then you can set the $INSTALL4J\_JAVA\_HOME$ environment variable to refer to a Java installation's bin directory.

When started, the installation program will automatically search for version-compatible JDKs on the system. If more than one JDK is found, a list is displayed from which you can choose. If only a JRE is found on the system, you can continue with the installation, but the Reflection ZFE server will not run correctly until you have updated the wrapper.java.command property located in sessionserver/container.conf to refer to a JDK installation.

- Both Reflection ZFE and MSS require that the Java installation support unlimited strength encryption. More information is available on the Java web site.
- If necessary, you can use the environment variables named above and $INSTALL4J\_JAVA\_HOME\_OVERRIDE$ to specify a specific Java installation.
- If you plan on using the IIS Reverse Proxy with Reflection ZFE, read Technical Note 2859 for prerequisites and configuration instructions.

System requirements

All requirements listed are the minimum required to successfully install Reflection ZFE.

Supported web browsers

The only thing needed to access Reflection ZFE terminal emulation is a supported web browser. The following web browsers are currently supported:

- Google Chrome 33+
- Mozilla Firefox 27+
- Microsoft Internet Explorer 11+
  See Technical Note 2827 for information on performance issues when using Internet Explorer.
- Microsoft Edge
- Apple iOS Safari 7+

MSS is platform independent and supports any web browser using JRE 7 or later that supports JavaScript and Cascading Style Sheets (CSS).

Session server operating systems

The Reflection ZFE session server supports the following 64-bit platforms:

- Windows 2008 Server
- Solaris 10 (SPARC)
- Red Hat Enterprise Linux (RHEL) 6.x
- SUSE Enterprise Linux 11.x
- AIX 6.x

z/Linux (SUSE E11.x and RHEL 6.x) installation

Follow the procedures described in the download site instructions.
Installing on UNIX platforms

- You must either install as “root” or use a user account with root privileges to complete successfully. When the installation has successfully completed, the installed application can be started and managed by “root” or someone running as ‘root’.
- Elevated privileges are needed to open any application ports lower than 1024. Reflection ZFE will not start using a lower port number unless you have system privileges to open low numbered ports.
- You can use the `chmod` command to assign application privileges to users other than root.

Preparing to install

Reflection ZFE supports TLS and SSH protocols to protect mission-critical data. To secure your passwords and other sensitive data, you should require browsers to use the HTTPS protocol.

To configure a Reflection ZFE session to use TLS, you must first establish a “trust” for the public certificate chain of the host to which you’re connecting. MSS centrally manages the trust store that Reflection ZFE uses. By default, the Reflection ZFE session server fetches this trust store every time it attempts a connection.

For a successful installation you must have a valid certificate signed by a trusted Certificate Authority (CA) and install it on the session server. To head off any installation issues, read Making Secure Connections. In a typical Reflection ZFE installation there are three main connection points that you need to consider in regard to security, the Making Secure Connections topic deals with all three; web browser to Reflection ZFE session server, Reflection ZFE session server to MSS, and Reflection ZFE session server to the host legacy system.

Ports used by Reflection ZFE

Configure your firewall to allow connections on the following TCP listening ports:

<table>
<thead>
<tr>
<th>Component</th>
<th>Default Port Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reflection ZFE session server</td>
<td>7070 - HTTP</td>
</tr>
<tr>
<td></td>
<td>7443 - HTTPS</td>
</tr>
<tr>
<td>MSS</td>
<td>80 - HTTP</td>
</tr>
<tr>
<td></td>
<td>443 - HTTPS</td>
</tr>
</tbody>
</table>

Both the Reflection ZFE and the MSS Administrative Server ports can be changed depending on your network needs. To modify the Reflection ZFE session server ports, see How to Change Ports.

Upgrading from previous installations

Upgrading is a simple and easy. It’s best to back up any previous work before you upgrade.

To upgrade from previous versions to the current version:

2. Uninstall the previous version of Reflection ZFE, but do not uninstall Management and Security Server.
3. Install the latest version of Reflection ZFE.

Troubleshooting the installation

To complete a successful installation, make sure that you have taken care of these common connection issues:

✔️ Is MSS configured for HTTPS?

Connect to the system where the Administrative Server is installed and log in to the Administrative Server. In the Administrative WebStation, open the Security Setup section and note the protocol selection.

✔️ Verify that both MSS and Reflection ZFE are using trusted certificates.

MSS imports certificates and private keys to C:\ProgramData\MicroFocus\MSS\MSSData\certificates.

If you are not using trusted certificates, have you configured Reflection ZFE to run using HTTP?

✔️ Are your connection properties configured properly?

In the unlikely event that you have to verify connection information, the container.properties file for both the management component and the Reflection ZFE session server contains the connection properties needed to make the Reflection ZFE to MSS connection as well as the browser to Reflection ZFE connection.

You can find the file in the Reflection ZFE installation at <install-dir>/sessionserver/conf/container.properties.

Connecting using HTTP

If you do not have a trusted certificate in place, you can configure Reflection ZFE to use HTTP. This configuration is not secure and should be used only when no other option is available.

<table>
<thead>
<tr>
<th>Connecting to...</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>An existing remote MSS Administrative Server</td>
<td>1. During the Reflection ZFE installation, after you accept the license agreement and choose a destination directory, select Use remotely hosted MSS. Click Next.&lt;br&gt;2. Enter either the host name, DNS name, or IP address.&lt;br&gt;3. Change the port from 443 to 80.&lt;br&gt;4. Select HTTP and complete the installation process.</td>
</tr>
</tbody>
</table>
Other known issues

This section documents miscellaneous known issues and work around tips for Reflection ZFE.

**HTTPS connections between mobile devices running Apple iOS 8 and the Reflection session server**

Due to an apparent bug in iOS 8, Reflection ZFE users cannot connect to a session server over HTTPS from their iPad when using a self-signed certificate. If feasible, the quickest solution is to use HTTP instead of HTTPS.

If HTTPS is needed, you have the following options:

- Obtain a valid certificate signed by a trusted CA and install it on the session server.
- Find an alternate browser that will accept the self-signed certificate. See System requirements for a list of supported browsers.
- Leverage a custom certificate authority:
  1. Create a custom CA, CA root certificate, and a server certificate signed by that CA’s root certificate.
  2. Install the server certificate on the session server.
  3. Install the custom CA root certificate on the iPad by means of a profile. The iPad should now accept the server certificate as it was signed by a “trusted CA”.

For a list of CAs trusted by Apple iOS, see Lists of available trusted root certificates in iOS (https://support.apple.com/en-us/HT204132).

**SSL/TLS Error message Issues**

- **(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.
Setting Post 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>MicroFocus\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.
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 3-1 Metering options**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metering.enabled</td>
<td>Turns metering on or off, with a value of &quot;true&quot; or &quot;false&quot;. Any value other than &quot;true&quot; turns metering off.</td>
</tr>
<tr>
<td>metering.host.required</td>
<td>Determines whether the session can connect to the host even if the metering server cannot be contacted. &quot;True&quot; means that session connections will fail if the metering host is unavailable. &quot;False&quot; means that session connections will still work even if the metering host is unavailable.</td>
</tr>
<tr>
<td>metering.server.url</td>
<td>Specifies the name or address of the metering server, the port, the protocol, and the webapp context. The syntax is &quot;host:port</td>
</tr>
</tbody>
</table>

#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
    ;;
    *)
        echo "Invalid option "$1""
        echo "Usage: start|stop|status"
        echo "chkconfig: 235 19 08"
        echo "Description: Manage the Reflection ZFE session server"
        exit 1
esac
```
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.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Follow these steps</th>
</tr>
</thead>
</table>
| Linux    | 1. Copy the file to the /etc/init.d directory.  
          | 2. Set the file permission. Run chmod using the value 755. For example,  
          | chmod 755 zfe  
          | 3. Run chkconfig to add the initialization script. For example, /sbin/  
          | chkconfig --add zfe |
| Solaris  | 1. Copy the file to the /etc/init.d directory.  
          | 2. Set the file permission. Run chmod using the value 755. For example,  
          | chmod 755  
          | 3. Create a symbolic link in /etc/rc.d/rc3.d. For example, ln -s/etc/  
          | init.d/zfe/etc/rc3.d/S99zfe |
| AIX      | 1. Copy the file to /etc/rc.zfe. For example, cp zfe /etc/rc.zfe  
          | 2. Set the file permission. Run chmod. For example, chmod +x /etc/  
          | rc.zfe  
          | 3. Add these lines at the end of the /etc/rc.tcpip file: start/etc/rc.zfe  
          | "" "start"

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>
<thead>
<tr>
<th>Table 3-2 Reflection ZFE and MSS Default Ports</th>
</tr>
</thead>
</table>
| Session server  | HTTP - 7070  
          | HTTPS - 7443 |
| Management and Security Server | HTTP - 80  
          | HTTPS - 443 |

To change the default ports:
### 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](#)
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 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  keytool -list -keystore keystore.jks
Delete certificates  keytool -delete-alias mydomain -keystore keystore.jks
Export a certificate  keytool -export -alias mydomain -file mydomain.cer -keystore keystore.jks
Import a certificate  keytool -importcert -file <path to certificate> -alias <some-alias> -keystore servletcontainer.jks -storetype jceks -storepass not-secure
View stand alone certificate  keytool -printcert -v -file mydomain.crt

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, changeit 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
Making Secure Connections

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:


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\Micro Focus\mss\MssData. The file can be either a base64-encoded DER certificate or a binary Reflection Administrative Server certificate.
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

Depending on your host type, you can configure a terminal session using different security protocols.

To configure a terminal session using TLS/SSL

To connect to the new trusted host using TLS/SSL, 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.

To configure a VT terminal session 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
AAAAAB3NzaC1yc2EAAAADAQABA...........
```

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 user name 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. Install the signing authorities certificate into MSS and Reflection ZFE.
2. Restart the servers.

#### Step 1. 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:**
  - In \<RZFE_install_directory>\sessionserver\etc import the certificate: keytool -importcert -file <path to certificate> -alias <ca_friendly_name> -keystore servletcontainer.jks -storetype jceks -storepass not-secure

**Step 2. Restart all the servers**

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

**Step 3. 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 Connections
Configuring Connections

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.

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.

See Providing access to the session for information on the two ways available to make sessions available to your users.

Providing access to the session

Your users have access to their assigned sessions through a URL you provide (for example, https://<sessionserver>:7443/zfe). From this URL users select which session to open from the list of available sessions you have configured for them.
Your users can switch between sessions, open additional sessions and close sessions with which they are no longer working.

**NOTE:** A new session will not be launched if the specified session already exists when the user opens the link.

Alternatively, you can provide URLs to particular sessions that are launched using name and session parameters, (for example a direct link on a company portal page). To enable the launch of a single session use the query parameter `singleSession`. You can use this parameter on its own to just launch the Reflection ZFE web client in single session mode, for example, `http://<sessionserver>:7443/zfe/?exampleSessionName`, or it can be used in conjunction with a named session parameter to launch a particular named session in single session mode: `http://<sessionserver>:7443/zfe/?singleSession&name=HumanResources`. The order of the parameters does not matter.

When your users access a single session, they cannot switch between open sessions and cannot open new sessions.

---

**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.

- **Reconnect when host terminates connection**
When set to Yes, Reflection ZFE attempts to reconnect as soon as the host connection terminates.

**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 5-1  Protocol Descriptions**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TN3270</td>
<td>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.</td>
</tr>
<tr>
<td>TN3270E</td>
<td>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.</td>
</tr>
<tr>
<td>TN5250</td>
<td>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.</td>
</tr>
<tr>
<td>Secure Shell (VT)</td>
<td>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</td>
</tr>
<tr>
<td>Telnet (VT)</td>
<td>Telnet Secure Socket Layer (SSL) and Transport Layer Security (TLS) are available to provide secure connections.</td>
</tr>
<tr>
<td>INT1 (UTS)</td>
<td>Provides access to Unisys 1100/1200 hosts using the TCP/IP network protocol.</td>
</tr>
</tbody>
</table>

**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 at least one Pool of IDs in Terminal ID Manager before the ID request can succeed.

<table>
<thead>
<tr>
<th>Table 5-2  Terminal ID Manager Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Criterion</strong></td>
</tr>
<tr>
<td>Pool name</td>
</tr>
<tr>
<td>Client IP address</td>
</tr>
<tr>
<td>Host address</td>
</tr>
<tr>
<td>Host port</td>
</tr>
<tr>
<td>Session name</td>
</tr>
<tr>
<td>Session type</td>
</tr>
<tr>
<td>User name</td>
</tr>
<tr>
<td>To configure a session based on user names, a default placeholder user name is available: tidm-setup.</td>
</tr>
</tbody>
</table>

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)

If you choose **Custom Model**, you can set the number of columns and rows to customize the terminal model.

**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>
<thead>
<tr>
<th>Security options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS 1.2, 1.1, 1.0, and SSL 3.0</td>
<td>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.</td>
</tr>
<tr>
<td>TLS 1.2, 1.1, 1.0</td>
<td>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.</td>
</tr>
<tr>
<td>SSL 3.0</td>
<td>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 &quot;SSLv3&quot; from the jdk.tls.disabledAlgorithms property in the java.security file or by dynamically setting this Security property to &quot;true&quot; before JSSE is initialized. For example: You must remove or comment out the line, jdk.tls.disabledAlgorithms=SSLv3, in &lt;install_dir&gt;\MicroFocus\ReflectionZFE\jre\jre\lib\security\java.security.</td>
</tr>
</tbody>
</table>

**Table 5-3** **TLS/SSL Descriptions**

**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 5-4  VT session configuration options

<table>
<thead>
<tr>
<th>VT Settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal ID</td>
<td>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.</td>
</tr>
<tr>
<td>Local Echo</td>
<td>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.</td>
</tr>
<tr>
<td>Renegotiate Echo</td>
<td>No (default). When set to Yes, passwords are not displayed on the local screen, but all other typed text is visible. Reflection ZFE supports the Telnet Suppress Local Echo (SLE) option when connected to a host in half-duplex mode. This means that Reflection ZFE will suppress character echo to the host computer, and with SLE support Reflection ZFE can be instructed to suppress echo locally.</td>
</tr>
<tr>
<td>Set Host Window Size</td>
<td>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.</td>
</tr>
<tr>
<td>Request Binary</td>
<td>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.</td>
</tr>
<tr>
<td>Send LF after CR</td>
<td>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.</td>
</tr>
<tr>
<td>Ctrl-break sends</td>
<td>Choose what sequence Ctrl-break sends to the host when pressed. Options are: Telnet break sequence (the default), Interrupt process, or Nothing.</td>
</tr>
<tr>
<td>Host Character Set</td>
<td>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.</td>
</tr>
<tr>
<td>Auto Answerback</td>
<td>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.</td>
</tr>
</tbody>
</table>
In addition to the common connection settings, UTS hosts require these additional settings:

**Table 5-5  UTS INT1 session configuration options**

<table>
<thead>
<tr>
<th>UTS INT1 options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>The name of the host application or host operating mode to be accessed. 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.</td>
</tr>
<tr>
<td>TSAP</td>
<td>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.</td>
</tr>
<tr>
<td>Initial transaction</td>
<td>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.</td>
</tr>
<tr>
<td>Start transaction</td>
<td>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. 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. You can use this setting in combination with Send initial transaction.</td>
</tr>
</tbody>
</table>
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
Configuring Session Options

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

How do I?

- Configure Display Settings
- Map keys
- Configure macros for your users
- Transfer files
- Specify copy and paste options
- Set User Preference Rules
- Customize your session

Configure Display Settings

You can customize host foreground and background colors, set up hotspots and map keys to make it easier to navigate the host application, as well as specify other host-specific options. 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
- Set VT scrollback buffer options
- Set keyboard options
- Configure terminal settings
- Configure hotspots
- Map keys
- 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.
- **Inverse video** - This option reverses the foreground and background colors when the VT host sends an inverse video escape sequence. If this option is not enabled, the inverse video sequences sent from the host are ignored.

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 **Foreground** to choose a color to map the new color for the text or type the Hex code you want to use. Select **Background** to map the new color to the 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 6-1  Hotspots for 3270 Hosts

<table>
<thead>
<tr>
<th>Hotspot</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF1...PF24</td>
<td>Transmits a PF1...PF24 to the host</td>
</tr>
<tr>
<td>PA1, PA2, or PA3</td>
<td>Transmits a PA1, PA2, or PA3 to the host</td>
</tr>
<tr>
<td>enter</td>
<td>Transmits an Enter key to the host</td>
</tr>
<tr>
<td>more</td>
<td>Transmits a Clear key to the host</td>
</tr>
</tbody>
</table>

### Table 6-2  Hotspots for 5250 Hosts

<table>
<thead>
<tr>
<th>Hotspot</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enter</td>
<td>Transmits an Enter key to the host</td>
</tr>
<tr>
<td>more...</td>
<td>Transmits a Roll Up key to the host (scrolls down one page)</td>
</tr>
<tr>
<td>PF1 - PF24</td>
<td>Transmits a PF1...PF24 to the host</td>
</tr>
</tbody>
</table>

### Table 6-3  Hotspots for VT Hosts

<table>
<thead>
<tr>
<th>Hotspot</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1 - F20</td>
<td>Transmits a F1...F20 to the host</td>
</tr>
</tbody>
</table>

**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.
   - Select Clear on host change to clear the terminal window and move the contents to the scrollback buffer when the column size changes. (VT-only).
3. Click Save.

**Set cursor options**

Use the cursor options to configure the appearance and behavior of the cursor and ruler.
### Configuring Session Options

#### Set VT scrollback buffer options

The VT scrollback buffer contains the data that has scrolled off the display and is no longer accessible by the host computer. When a scrollback buffer exists, you can view it by using the vertical scroll bar.

The scrollback buffer is enabled by default. When enabled, the session maintains a buffer of lines that have scrolled off the terminal screen. This option is available to all users when they are granted permission to modify Terminal Display Settings by the administrator.

<table>
<thead>
<tr>
<th>This option</th>
<th>Does this...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cursor type</strong></td>
<td>- <strong>Underline</strong> displays the text cursor as an underline.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Vertical bar</strong> displays the cursor as a vertical line.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Block</strong> displays the text cursor as an inverse video block.</td>
</tr>
<tr>
<td><strong>Ruler type</strong></td>
<td>- <strong>Vertical</strong> displays a vertical ruler at the cursor position.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Horizontal</strong> displays a horizontal ruler at the cursor position.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Crosshair</strong> displays both a horizontal and vertical ruler at the cursor position.</td>
</tr>
<tr>
<td><strong>Cursor color</strong></td>
<td>Click the color field to open the color grid. From the color grid, select the color you want to use as the color of both the cursor and ruler. Alternatively, type the Hex color number for the color you want to use.</td>
</tr>
<tr>
<td><strong>Cursor blinks</strong></td>
<td>By default, the cursor (whether in block or underline mode) blinks. Clear this option to display a visible non-blinking cursor.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>This option</strong></th>
<th><strong>Does this...</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scrollback row limit</strong></td>
<td>Limits the number of rows held in the scrollback buffer. The default setting is 500 rows.</td>
</tr>
<tr>
<td><strong>Save display before clearing</strong></td>
<td>When selected (the default), the data on the terminal display moves into the scrollback buffer when you, or the host, clear the terminal display. If you prefer not to have the terminal display saved to the scrollback buffer, clear this option; when the terminal display is cleared, the data is discarded.</td>
</tr>
<tr>
<td><strong>Save from scrolling regions</strong></td>
<td>When top and bottom screen margins are set (for example, by a text editor such as EDT or TPU, or with the DECSTBM function) the area within the margins is called the scrolling region. When this option is cleared, scrolling text within this region isn't saved to the scrollback buffer. Select this option to save information within scrolling regions to the scrollback buffer. <strong>Note:</strong> This can cause display memory to fill quickly.</td>
</tr>
<tr>
<td><strong>Save before clearing from any row</strong></td>
<td>This setting specifies whether data that has been cleared from a portion of the terminal window is saved in display memory.</td>
</tr>
<tr>
<td><strong>Compress blank rows</strong></td>
<td>Select this option to save room in display memory by compressing multiple blank rows into a single blank row.</td>
</tr>
</tbody>
</table>
Set keyboard options

You can set the following keyboard options:

- **Backspace sends** (VT)
  
  Configures the function that the Backspace key sends. On the VT terminal keyboard the back arrow key (<x) is configurable: it can send either a delete (ASCII 127) or a backspace (ASCII 8) character.

- **Cursor keys**
  
  Controls the characters that the four arrow keys (on both the numeric and editing keypad) transmit. This setting is typically set by the host. In general, you should keep this set to **Normal**.
  
  If the arrow keys aren’t working properly, it may mean that this option remained incorrectly set to Application when a host program terminated abnormally. Changing this setting back to Normal should fix the problem with the arrow keys.

- **Keypad**
  
  Controls the characters that the numeric keypad keys transmit. This setting is typically set by the host. In general, you should keep this set to **Numeric**.
  
  If the number or PF keys aren’t working properly, it may mean that this option was incorrectly left set to Application when a host program terminated abnormally. Changing this setting back to Numeric should fix the numeric keypad.

- **Local echo** (VT)
  
  This option causes each character typed at the keyboard to be displayed on the screen. This option is cleared by default, because most hosts echo back received characters.

Related Topics

Map keys
3270, 5250, VT, and UTS Keyboard Mapping

Configure terminal settings

The following terminal settings are specific to VT host types.

- **Terminal type** (VT)
  
  Specifies which terminal should be emulated. These choices affect the codes generated by the numeric keypad, the interpretation of control functions, and the response to terminal identification requests.

- **Terminal ID** (VT)
  
  Specifies 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. This setting is independent of the terminal type setting. When set to the default value of Reflection, Reflection ZFE responds to a primary DA request with the set of features it supports. If your host requires a more specific terminal ID, select another value from the list.
新行 (VT)

选择此选项以在您按 Enter 时发送回车和换行。当 Reflection ZFE 收到换行、行结束或垂直制表符时，它将光标移动到下一行的第一个列。当此选项清除时（默认），Enter 键仅发送回车。从主机接收的换行、行结束或垂直制表符将光标移动到当前列的下一行。如果显示中的行被不断覆盖（即，主机未随回车发送换行），则选择此选项。如果新的行选项被选中但主机不期望随每个回车发送换行，则行将被双倍行距显示。

设置其他显示选项

有些显示选项是主机特定的，如下面所示。当主机类型未指定时，这些选项适用于所有支持的主机类型。

<table>
<thead>
<tr>
<th>选项</th>
<th>做到……</th>
</tr>
</thead>
<tbody>
<tr>
<td>输入字段下划线 (3270, 5250)</td>
<td>您可以确定如何处理主机输入字段的下划线：</td>
</tr>
<tr>
<td></td>
<td>- 主机控制输入字段下划线 (默认)</td>
</tr>
<tr>
<td></td>
<td>- 永远下划线输入字段</td>
</tr>
<tr>
<td></td>
<td>- 从不下划线输入字段</td>
</tr>
<tr>
<td>状态行 (VT)</td>
<td>通过底部显示开启状态行。</td>
</tr>
<tr>
<td></td>
<td>选择：</td>
</tr>
<tr>
<td></td>
<td>- 无以禁用状态行。 (默认)</td>
</tr>
<tr>
<td></td>
<td>- 指示器以显示页面、光标位置和打印机状态。</td>
</tr>
<tr>
<td></td>
<td>- 主机可写以允许主机应用程序在状态行显示信息。</td>
</tr>
<tr>
<td>保持方面比选</td>
<td>选择此选项以保持显示的方面比，无论浏览器窗口的大小。 aspect ratio 描述了一个图像的宽度与其高度的比。</td>
</tr>
<tr>
<td>显示 OIA (3270, 5250)</td>
<td>选择此选项以在终端窗口的底部显示操作和状态消息。 OIA display 默认是开启的。</td>
</tr>
<tr>
<td>忽略鼠标点击 (窗口激活)</td>
<td>当鼠标点击激活终端窗口时，此选项指定是否执行更新光标位置、清除选择或执行热区的操作。  默认情况下，这些操作未执行。</td>
</tr>
</tbody>
</table>
| 自动换行 (VT)                 | 当选择时，字符自动在右边缘结束并继续在下一行。 当清除时，到达右边缘的字符不换行，直到回车。新字符覆盖右边缘的字符，直到回车。
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.

When you create a keyboard shortcut to perform actions, like Run Macro or Send Text, you can specify the necessary parameters in the Value field.

**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, Ctrl+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. From the toolbar, click Settings.
2. From the left navigation pane, open the Key Mappings panel.
3. Under Modified Mappings, click Add to enter a new key map entry.
4. Press the key or key combination you want to trigger the assigned action in the key field.
5. From the Action drop down list, select the action you want to associate with the key selection. If you select Send text, enter the string you want sent to the host in the Value field. Likewise, if you select Run Macro, choose the macro you want triggered by the keyboard shortcut.

   ![Key Mapping Panel](image)

   The Send text action supports mapping characters with codes less than or equal to 0xFFFF via Unicode escape sequences. The escape sequence begins with \\u followed by exactly four hexadecimal digits. You can embed Unicode escape sequences in any string. For example, *this embedded \u0045* will be interpreted as *this embedded E*, since 45 is the hexadecimal code for the character E.

   To pass Unicode escape sequences to the host, escape the sequence with a leading backslash. For example, to send the string literal \u001C to the host, map a key to \\u001C. Reflection ZFE will convert this to the string \u001C when that key is pressed and send the 6 characters of the resulting string to the host.

6. Click the blue check mark to accept the mapping and add the key map to the session.
7. Click Save to close the Key Mapping panel and resume configuring your host connection.
After you complete mapping keys for the session, the updated Current Mappings table displays the custom key mappings in boldface type. 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, Ctrl+ Shift + A initiates a Select All action. See 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 6-4 IBM 3270 Keyboard Mapping

<table>
<thead>
<tr>
<th>Key</th>
<th>Maps to</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + F1</td>
<td>Attention</td>
<td>Sends the ATTENTION key to the host</td>
</tr>
<tr>
<td>Shift + Tab</td>
<td>Backtab</td>
<td>Moves the cursor to the previous unprotected field</td>
</tr>
<tr>
<td>Ctrl + F2</td>
<td>Clear</td>
<td>Clears the screen and sends the CLEAR key to the host</td>
</tr>
<tr>
<td>Alt + ArrowLeft</td>
<td>Cursor left double</td>
<td>Moves the cursor two positions to the left</td>
</tr>
<tr>
<td>Alt + ArrowRight</td>
<td>Cursor right double</td>
<td>Moves the cursor two positions to the right</td>
</tr>
<tr>
<td>Ctrl + F3</td>
<td>Cursor select</td>
<td>Simulates a lightpen select in the current field</td>
</tr>
<tr>
<td>Alt + Delete</td>
<td>Delete word</td>
<td>Deletes three characters from the current field</td>
</tr>
<tr>
<td>Ctrl + 5</td>
<td>Duplicate</td>
<td>Inserts the DUP character at the cursor location</td>
</tr>
<tr>
<td>Enter</td>
<td>Enter</td>
<td>Sends the ENTER key to the host</td>
</tr>
<tr>
<td>End</td>
<td>Erase end of field</td>
<td>Erases all data from the cursor location to the end of the current field</td>
</tr>
<tr>
<td>Alt + F5</td>
<td>Erase input</td>
<td>Erases all data in all unprotected fields of the current screen.</td>
</tr>
<tr>
<td>Ctrl + Alt + F</td>
<td>Field delimiter</td>
<td>Toggles whether field delimiters are displayed on screen</td>
</tr>
<tr>
<td>Ctrl + 6</td>
<td>Field mark</td>
<td>Inserts the Field Mark character at the cursor location</td>
</tr>
<tr>
<td>Home</td>
<td>Home</td>
<td>Moves the cursor to the first unprotected field on the screen</td>
</tr>
<tr>
<td>Insert</td>
<td>Insert</td>
<td>Toggles Insert mode</td>
</tr>
</tbody>
</table>
### Table 6-5 IBM 5250 Keyboard Mapping

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

**Table 6-5 IBM 5250 Keyboard Mapping**
<table>
<thead>
<tr>
<th>Key</th>
<th>Maps to</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + F3</td>
<td>Cursor select</td>
<td>Simulates a lightpen select in the current field</td>
</tr>
<tr>
<td>Ctrl + Backspace</td>
<td>Destructive backspace</td>
<td>Moves the cursor one position to the left</td>
</tr>
<tr>
<td>Ctrl + 5</td>
<td>Duplicate</td>
<td>Inserts the DUP character at the cursor location</td>
</tr>
<tr>
<td>Ctrl + End</td>
<td>End of field</td>
<td>Moves the cursor to the end of the field</td>
</tr>
<tr>
<td>End</td>
<td>Erase end of field</td>
<td>Erases all data from the cursor location to the end of the current field</td>
</tr>
<tr>
<td>Alt + End</td>
<td>Erase input</td>
<td>Erases all data in the all unprotected fields of the current screen</td>
</tr>
<tr>
<td>Alt + F5</td>
<td>Erase input</td>
<td>Erases all data in all unprotected fields of the current screen.</td>
</tr>
<tr>
<td>Ctrl + Enter</td>
<td>Field exit</td>
<td>Moves the cursor out of an input field</td>
</tr>
<tr>
<td>KP + Subtract</td>
<td>Field exit minus</td>
<td>Moves the cursor out of a signed-numeric or numeric-only field</td>
</tr>
<tr>
<td>Ctrl + Subtract</td>
<td>Field exit minus</td>
<td>Moves the cursor out of a signed-numeric or numeric-only field</td>
</tr>
<tr>
<td>KP + Add</td>
<td>Field exit plus</td>
<td>Moves the cursor out of a signed-numeric or numeric-only field</td>
</tr>
<tr>
<td>Ctrl + Add</td>
<td>Field exit plus</td>
<td>Moves the cursor out of a signed-numeric or numeric-only field</td>
</tr>
<tr>
<td>Ctrl + 6</td>
<td>Field mark</td>
<td>Inserts the field mark character at the cursor location</td>
</tr>
<tr>
<td>Ctrl + H</td>
<td>Help</td>
<td>Sends the Help key to the host</td>
</tr>
<tr>
<td>Ctrl + X</td>
<td>Hex mode</td>
<td>Places the terminal in Hex mode</td>
</tr>
<tr>
<td>Home</td>
<td>Home</td>
<td>Moves the cursor to the first unprotected field on the screen</td>
</tr>
<tr>
<td>Insert</td>
<td>Insert</td>
<td>Toggles Insert mode</td>
</tr>
<tr>
<td>Shift + Enter</td>
<td>New line</td>
<td>Moves to the next unprotected field</td>
</tr>
<tr>
<td>Ctrl + 1</td>
<td>PA1</td>
<td>Sends the PA1 key to the host</td>
</tr>
<tr>
<td>Ctrl + 2</td>
<td>PA2</td>
<td>Sends the PA2 key to the host</td>
</tr>
<tr>
<td>Ctrl + 3</td>
<td>PA3</td>
<td>Sends the PA3 key to the host</td>
</tr>
<tr>
<td>F1 - F11</td>
<td>PF1 - PF11</td>
<td>Sends the PF1, PF2,…PF11 key to the host</td>
</tr>
<tr>
<td>Alt + 1</td>
<td>PF11</td>
<td>Sends the PF11 key to the host</td>
</tr>
<tr>
<td>Alt + 2</td>
<td>PF12</td>
<td>Sends the PF12 key to the host</td>
</tr>
<tr>
<td>F12</td>
<td>PF12</td>
<td>Sends the PF12 key to the host</td>
</tr>
<tr>
<td>Shift + 1</td>
<td>PF13</td>
<td>Sends the PF13 key to the host</td>
</tr>
<tr>
<td>Shift + F2</td>
<td>PF14</td>
<td>Sends the PF14 key to the host</td>
</tr>
<tr>
<td>Shift + F3</td>
<td>PF15</td>
<td>Sends the PF15 key to the host</td>
</tr>
<tr>
<td>Shift + F4</td>
<td>PF16</td>
<td>Sends the PF16 key to the host</td>
</tr>
</tbody>
</table>
### Table 6-6  VT Keyboard Mapping

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

<table>
<thead>
<tr>
<th>Key</th>
<th>Maps to</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + Cancel</td>
<td>Break</td>
<td>Sends the Break key to the host</td>
</tr>
<tr>
<td>Ctrl + Enter</td>
<td>Enter</td>
<td>Send the Enter key to the host</td>
</tr>
<tr>
<td>Alt + F1</td>
<td>F1</td>
<td>Sends the F1 key to the host</td>
</tr>
<tr>
<td>Ctrl + F1</td>
<td>F11</td>
<td>Sends the F11 key to the host</td>
</tr>
<tr>
<td>Ctrl + F2</td>
<td>F12</td>
<td>Sends the F12 key to the host</td>
</tr>
<tr>
<td>Ctrl + F3</td>
<td>F13</td>
<td>Sends the F13 key to the host</td>
</tr>
<tr>
<td>Ctrl + F4</td>
<td>F14</td>
<td>Sends the F14 key to the host</td>
</tr>
<tr>
<td>Ctrl + F5</td>
<td>F15</td>
<td>Sends the F15 key to the host</td>
</tr>
<tr>
<td>Ctrl + F6</td>
<td>F16</td>
<td>Sends the F16 key to the host</td>
</tr>
<tr>
<td>Ctrl + F7</td>
<td>F17</td>
<td>Sends the F17 key to the host</td>
</tr>
<tr>
<td>Ctrl + F8</td>
<td>F18</td>
<td>Sends the F18 key to the host</td>
</tr>
<tr>
<td>Ctrl + F9</td>
<td>F19</td>
<td>Sends the F19 key to the host</td>
</tr>
<tr>
<td>Ctrl + F10</td>
<td>F20</td>
<td>Sends the F20 key to the host</td>
</tr>
<tr>
<td>Home</td>
<td>Find</td>
<td>Sends the Find key to the host</td>
</tr>
<tr>
<td>Key</td>
<td>Maps to</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>F1</td>
<td>Hold</td>
<td>Sends the Hold Screen to the host</td>
</tr>
<tr>
<td>Pause</td>
<td>Hold</td>
<td>Sends the Hold Screen to the host</td>
</tr>
<tr>
<td>Insert</td>
<td>Insert</td>
<td>Sends the Insert key to the host</td>
</tr>
<tr>
<td>Ctrl + Insert</td>
<td>Keypad 0</td>
<td>Sends the numeric keypad 0 key to the host</td>
</tr>
<tr>
<td>Ctrl + End</td>
<td>Keypad 1</td>
<td>Sends the numeric keypad 1 key to the host</td>
</tr>
<tr>
<td>Ctrl + ArrowDown</td>
<td>Keypad 2</td>
<td>Sends the numeric keypad 2 key to the host</td>
</tr>
<tr>
<td>Ctrl + Pagedown</td>
<td>Keypad 3</td>
<td>Sends the numeric keypad 3 key to the host</td>
</tr>
<tr>
<td>Ctrl + ArrowLeft</td>
<td>Keypad 4</td>
<td>Sends the numeric keypad 4 key to the host</td>
</tr>
<tr>
<td>Ctrl + Clear</td>
<td>Keypad 5</td>
<td>Sends the numeric keypad 5 key to the host</td>
</tr>
<tr>
<td>Ctrl + ArrowRight</td>
<td>Keypad 6</td>
<td>Sends the numeric keypad 6 key to the host</td>
</tr>
<tr>
<td>Ctrl + Home</td>
<td>Keypad 7</td>
<td>Sends the numeric keypad 7 key to the host</td>
</tr>
<tr>
<td>Ctrl + ArrowUp</td>
<td>Keypad 8</td>
<td>Sends the numeric keypad 8 key to the host</td>
</tr>
<tr>
<td>Ctrl + Pageup</td>
<td>Keypad 9</td>
<td>Sends the numeric keypad 9 key to the host</td>
</tr>
<tr>
<td>Ctrl + Alt-add</td>
<td>Keypad comma</td>
<td>Sends the numeric keypad Comma key to the host</td>
</tr>
<tr>
<td>Ctrl + add</td>
<td>Keypad minus</td>
<td>Sends the numeric keypad Minus key to the host</td>
</tr>
<tr>
<td>Ctrl + decimal</td>
<td>Keypad period</td>
<td>Sends the numeric keypad Period key to the host</td>
</tr>
<tr>
<td>Ctrl + Delete</td>
<td>Keypad period</td>
<td>Sends the numeric keypad Period key to the host</td>
</tr>
<tr>
<td>Ctrl + Alt + ArrowUp</td>
<td>Row up</td>
<td>In the scrollback buffer moves up a row</td>
</tr>
<tr>
<td>Ctrl + Alt + ArrowDown</td>
<td>Row down</td>
<td>In the scrollback buffer moves down a row</td>
</tr>
<tr>
<td>Pagedown</td>
<td>Next</td>
<td>Sends the Next Screen key to the host</td>
</tr>
<tr>
<td>Ctrl + Pause</td>
<td>PF1</td>
<td>Sends the PF1 key to the host</td>
</tr>
<tr>
<td>Ctrl + Divide</td>
<td>PF2</td>
<td>Sends the PF2 key to the host</td>
</tr>
<tr>
<td>Ctrl + Multiply</td>
<td>PF3</td>
<td>Sends the PF3 key to the host</td>
</tr>
<tr>
<td>Ctrl + Subtract</td>
<td>PF4</td>
<td>Sends the PF4 key to the host</td>
</tr>
<tr>
<td>Pageup</td>
<td>Previous</td>
<td>Sends the Prev Screen key to the host</td>
</tr>
<tr>
<td>Delete</td>
<td>Remove</td>
<td>Sends the Remove key to the host</td>
</tr>
<tr>
<td>End</td>
<td>Select</td>
<td>Sends the Select key to the host</td>
</tr>
<tr>
<td>Shift + F6</td>
<td>UDK6</td>
<td>Sends the User Defined Key 6 to the host</td>
</tr>
<tr>
<td>Shift + F7</td>
<td>UDK7</td>
<td>Sends the User Defined Key 7 to the host</td>
</tr>
<tr>
<td>Shift + F8</td>
<td>UDK8</td>
<td>Sends the User Defined Key 8 to the host</td>
</tr>
<tr>
<td>Shift + F9</td>
<td>UDK9</td>
<td>Sends the User Defined Key 9 to the host</td>
</tr>
<tr>
<td>Shift + F10</td>
<td>UDK10</td>
<td>Sends the User Defined Key 10 to the host</td>
</tr>
<tr>
<td>Shift + Ctrl + F1</td>
<td>UDK11</td>
<td>Sends the User Defined Key 11 to the host</td>
</tr>
<tr>
<td>Key</td>
<td>Maps to</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>Shift + Ctrl + F2</td>
<td>UDK12</td>
<td>Sends the User Defined Key 12 to the host</td>
</tr>
<tr>
<td>Shift + Ctrl + F3</td>
<td>UDK13</td>
<td>Sends the User Defined Key 13 to the host</td>
</tr>
<tr>
<td>Shift + Ctrl + F4</td>
<td>UDK14</td>
<td>Sends the User Defined Key 14 to the host</td>
</tr>
<tr>
<td>Shift + Ctrl + F5</td>
<td>UDK15</td>
<td>Sends the User Defined Key 15 to the host</td>
</tr>
<tr>
<td>Shift + Ctrl + F6</td>
<td>UDK16</td>
<td>Sends the User Defined Key 16 to the host</td>
</tr>
<tr>
<td>Shift + Ctrl + F7</td>
<td>UDK17</td>
<td>Sends the User Defined Key 17 to the host</td>
</tr>
<tr>
<td>Shift + Ctrl + F8</td>
<td>UDK18</td>
<td>Sends the User Defined Key 18 to the host</td>
</tr>
<tr>
<td>Shift + Ctrl + F9</td>
<td>UDK19</td>
<td>Sends the User Defined Key 19 to the host</td>
</tr>
<tr>
<td>Shift + Ctrl + F10</td>
<td>UDK20</td>
<td>Sends the User Defined Key 20 to the host</td>
</tr>
</tbody>
</table>

Table 6-7  UTS Keyboard Mapping

<table>
<thead>
<tr>
<th>Key</th>
<th>Maps to</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F4</td>
<td>Clear Change Bit</td>
<td>Sends the CLEARCHANGEBIT key to the host.</td>
</tr>
<tr>
<td>Keypad+Enter</td>
<td>Carriage Return</td>
<td>Sends a carriage return to the host.</td>
</tr>
<tr>
<td>Ctrl+PageDown</td>
<td>Clear End of Display</td>
<td>Clears text from the cursor location to the end of the display.</td>
</tr>
<tr>
<td>Ctrl+PageUp</td>
<td>Clear End of Display FCC</td>
<td>Clears all data (including FCC information) from the cursor to the end of the display.</td>
</tr>
<tr>
<td>Ctrl+End</td>
<td>Clear End of Field</td>
<td>Clears text from the cursor location to the end of the field.</td>
</tr>
<tr>
<td>Ctrl+Shift+end</td>
<td>Clear End of Line</td>
<td>Clears text from the cursor location to the end of the row.</td>
</tr>
<tr>
<td>F7</td>
<td>Clear FCC</td>
<td>Clears the field control character</td>
</tr>
<tr>
<td>Ctrl+Home</td>
<td>Clear Home</td>
<td>Sends the CLEAR_HOME key to the host.</td>
</tr>
<tr>
<td>Ctrl+H</td>
<td>Column Separator Right</td>
<td>Sends the COLUMN_SEP_RIGHT key to the host.</td>
</tr>
<tr>
<td>Ctrl+F1</td>
<td>Control Page</td>
<td>Sends the CONTROL_PAGE key to the host.</td>
</tr>
<tr>
<td>Keypad+2</td>
<td>Cursor Down</td>
<td>Moves the cursor one row down.</td>
</tr>
<tr>
<td>Keypad+4</td>
<td>Cursor Left</td>
<td>Moves the cursor one column to the left.</td>
</tr>
<tr>
<td>Keypad+6</td>
<td>Cursor Right</td>
<td>Moves the cursor one column to the right.</td>
</tr>
<tr>
<td>Keypad+8</td>
<td>Cursor Up</td>
<td>Moves the cursor one row up.</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete in Line</td>
<td>Sends the DELETE_IN_LINE key to the host.</td>
</tr>
<tr>
<td>Ctrl+Delete</td>
<td>Delete in Page</td>
<td>Sends the DELETE_IN_PAGE key to the host.</td>
</tr>
<tr>
<td>Ctrl+Shift+Delete</td>
<td>Delete Line</td>
<td>Deletes the row at the cursor location.</td>
</tr>
<tr>
<td>Ctrl+ArrowDown</td>
<td>Duplicate Line</td>
<td>Duplicates the row at the cursor location.</td>
</tr>
<tr>
<td>F8</td>
<td>Enable FCC</td>
<td>Enables the field control character.</td>
</tr>
<tr>
<td>Key</td>
<td>Maps to</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Keypad+-</td>
<td>End of Display and Transmit</td>
<td>Sends the EOD_AND_TRANSMIT key to the host.</td>
</tr>
<tr>
<td>Shift+End</td>
<td>End of Field</td>
<td>Moves the cursor to the end of the field.</td>
</tr>
<tr>
<td>End</td>
<td>End of Line</td>
<td>Moves the cursor to the end of the row.</td>
</tr>
<tr>
<td>Ctrl+ArrowRight</td>
<td>End of Page</td>
<td>Moves the cursor to the end of the page.</td>
</tr>
<tr>
<td>Shift+Space</td>
<td>Erase Character</td>
<td>Erases the character at the cursor location.</td>
</tr>
<tr>
<td>Ctrl+Shift+E</td>
<td>Euro Character</td>
<td>Sends the Euro character to the host.</td>
</tr>
<tr>
<td>Ctrl+1 - Ctrl+9</td>
<td>F1 - F9</td>
<td>Sends the F1 - F9 key to the host.</td>
</tr>
<tr>
<td>Ctrl+0</td>
<td>F10</td>
<td>Sends the F10 key to the host.</td>
</tr>
<tr>
<td>Ctrl+-</td>
<td>F11</td>
<td>Sends the F11 key to the host.</td>
</tr>
<tr>
<td>Ctrl+=</td>
<td>F12</td>
<td>Sends the F12 key to the host.</td>
</tr>
<tr>
<td>Ctrl+Q</td>
<td>F13</td>
<td>Sends the F13 key to the host.</td>
</tr>
<tr>
<td>Ctrl+W</td>
<td>F14</td>
<td>Sends the F14 key to the host.</td>
</tr>
<tr>
<td>Ctrl+E</td>
<td>F15</td>
<td>Sends the F15 key to the host.</td>
</tr>
<tr>
<td>Ctrl+R</td>
<td>F16</td>
<td>Sends the F16 key to the host.</td>
</tr>
<tr>
<td>Ctrl+T</td>
<td>F17</td>
<td>Sends the F17 key to the host.</td>
</tr>
<tr>
<td>Ctrl+Y</td>
<td>F18</td>
<td>Sends the F18 key to the host.</td>
</tr>
<tr>
<td>Ctrl+U</td>
<td>F19</td>
<td>Sends the F19 key to the host.</td>
</tr>
<tr>
<td>Ctrl+I</td>
<td>F20</td>
<td>Sends the F20 key to the host.</td>
</tr>
<tr>
<td>Ctrl+O</td>
<td>F21</td>
<td>Sends the F21 key to the host.</td>
</tr>
<tr>
<td>Ctrl+P</td>
<td>F22</td>
<td>Sends the F22 key to the host.</td>
</tr>
<tr>
<td>Shift+F3</td>
<td>FF</td>
<td>Sends a formfeed to the host.</td>
</tr>
<tr>
<td>F9</td>
<td>Generate FCC</td>
<td>Generates a field control character.</td>
</tr>
<tr>
<td>Home</td>
<td>Home</td>
<td>Moves the cursor to the first field in the display.</td>
</tr>
<tr>
<td>Ctrl+Shift+Space</td>
<td>Insert in Line</td>
<td>Sends the INSERT_IN_LINE key to the host.</td>
</tr>
<tr>
<td>Ctrl+Space</td>
<td>Insert in Page</td>
<td>Sends the INSERT_IN_PAGE key to the host.</td>
</tr>
<tr>
<td>Ctrl+Shift+Insert</td>
<td>Insert Line</td>
<td>Inserts a new row into display memory.</td>
</tr>
<tr>
<td>Insert</td>
<td>Insert Mode</td>
<td>Toggles insert character mode.</td>
</tr>
<tr>
<td>F5</td>
<td>Locate FCC</td>
<td>Disables the field control characters and moves to the first character of the next field to the right of the cursor.</td>
</tr>
<tr>
<td>F3</td>
<td>Message Wait</td>
<td>Sends the MESSAGE_WAIT key to the host.</td>
</tr>
<tr>
<td>Shift+F2</td>
<td>New Line</td>
<td>Moves the cursor to a new row.</td>
</tr>
<tr>
<td>Keypad+Shift+2</td>
<td>Next Field</td>
<td>Moves the cursor to the next field.</td>
</tr>
<tr>
<td>Keypad+Shift+4</td>
<td>Next Field</td>
<td>Moves the cursor to the next field.</td>
</tr>
<tr>
<td>PageDown</td>
<td>Page Down</td>
<td>Sends the Page Down key to the host.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Key</th>
<th>Maps to</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PageUp</td>
<td>Page Up</td>
<td>Sends the Page Up key to the host.</td>
</tr>
<tr>
<td>Keypad+Shift+6</td>
<td>Previous Field</td>
<td>Moves the cursor to the previous field.</td>
</tr>
<tr>
<td>Keypad+Shift+8</td>
<td>Previous Field</td>
<td>Moves the cursor to the previous field.</td>
</tr>
<tr>
<td>Clear</td>
<td>SOE Character</td>
<td>Sends the SOE character to the host.</td>
</tr>
<tr>
<td>F12</td>
<td>SOE Character</td>
<td>Sends the SOE character to the host.</td>
</tr>
<tr>
<td>Ctrl+Clear</td>
<td>Set Tab</td>
<td>Sends the SET_TAB key to the host.</td>
</tr>
<tr>
<td>Ctrl+Tab</td>
<td>Set Tab</td>
<td>Sends the SET_TAB key to the host.</td>
</tr>
<tr>
<td>Shift+Home</td>
<td>Start of Field</td>
<td>Moves the cursor to the start of the field.</td>
</tr>
<tr>
<td>Ctrl+ArrowLeft</td>
<td>Start of Line</td>
<td>Moves the cursor to the start of the row</td>
</tr>
<tr>
<td>Ctrl+[</td>
<td>System Mode</td>
<td>Sends the SYSTEM_MODE key to the host.</td>
</tr>
<tr>
<td>Ctrl+J</td>
<td>Toggle Column Separator</td>
<td>Toggles the column separator.</td>
</tr>
<tr>
<td>Ctrl+F12</td>
<td>Toggle Message Wait Beep</td>
<td>Sends the TOGGLEMSGWAITBEEP key to the host.</td>
</tr>
<tr>
<td>Ctrl+L</td>
<td>Toggle Strike Thru</td>
<td>Toggles strike thru mode.</td>
</tr>
<tr>
<td>Ctrl+K</td>
<td>Toggle Underline</td>
<td>Toggles underline mode.</td>
</tr>
<tr>
<td>Ctrl+Enter</td>
<td>Transmit</td>
<td>Transmits the contents of the display to the host.</td>
</tr>
<tr>
<td>ScrollLock</td>
<td>Transmit</td>
<td>Transmits the contents of the display to the host.</td>
</tr>
<tr>
<td>Keypad++</td>
<td>Transmit</td>
<td>Transmits the contents of the display to the host.</td>
</tr>
<tr>
<td>Keypad+Ctrl+</td>
<td>Transmit</td>
<td>Transmits the contents of the display to the host.</td>
</tr>
<tr>
<td>Escape</td>
<td>Unlock</td>
<td>Sends the UNLOCK key to the host.</td>
</tr>
<tr>
<td>Ctrl+]</td>
<td>Workstation Mode</td>
<td>Sends the WORKSTATION_MODE key to the host.</td>
</tr>
</tbody>
</table>

**Related Topics**

Using the Macro API  
Sample Macros  
Creating Macros
Transfer files

Reflection ZFE provides two different methods for transferring files between computers. Using User Preferences, administrators make this functionality available to their users and to specific sessions. Depending on the transfer method you want to use, you will see different configuration options. Once configured, the file transfer dialog box is available from the toolbar.

- IND$FILE
- FTP

Related Topics
File and Data Transfer

IND$FILE

IND$FILE is a file transfer program from IBM which you can use to 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 CICS.

You cannot transfer files unless you are connected and logged into the host. After establishing a connection, from the toolbar, select IND$FILE to open the File Transfer dialog box.

For instructions on how to transfer files, see File and Data Transfer.

Related Topics
File and Data Transfer

FTP

With Reflection ZFE your local computer can act as an FTP client. Using the FTP client, you can connect to an FTP server running on another machine. Once connected, you can view files on the server and use FTP) to transfer files between your local computer (or any networked drive) and the FTP server. Using FTP, a client can upload, download, delete, rename, move and copy files on a server.

Select FTP to open the File Transfer dialog box.

To configure FTP

Select Enable FTP and proceed with the configuration:

- Protocol

Use FTP to start a standard FTP session. Use SFTP to start an SFTP session.
You can set up an FTP client to use the SFTP protocol and perform all operations over an encrypted secure shell transport. Reflection ZFE uses user name and password to authenticate.

- **Host**
  Specify the host name or IP address of the FTP server to which you want to connect.

- **Port**
  The port of the FTP server specified.

- **If remote files exists when uploading file**
  Specify how you want to handle the transfer if a file with the same name already exists. You can select:

<table>
<thead>
<tr>
<th>This option</th>
<th>Does this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append</td>
<td>Append the file being sent to the existing file</td>
</tr>
<tr>
<td>Ask user (default)</td>
<td>Prompt for a decision on how to handle the duplicate file name</td>
</tr>
<tr>
<td>Cancel</td>
<td>Cancel the file transfer</td>
</tr>
<tr>
<td>Fail</td>
<td>Cancel the file transfer and receive a notification of failure</td>
</tr>
<tr>
<td>Overwrite</td>
<td>Overwrite the existing file on the remote machine</td>
</tr>
<tr>
<td>Skip</td>
<td>When multiple files are in a request, skip the file matching an existing file name, but proceed with the transfer for other files.</td>
</tr>
<tr>
<td>Unique</td>
<td>Create a new file with a unique file name</td>
</tr>
</tbody>
</table>

- **Initial remote directory**
  Specify the path to a home or default directory for the FTP site. When a connection to the FTP site is opened, the server working directory is set automatically to the specified home path. The files and folders in the server home directory appear in the FTP session window. If the initial remote directory is not found, a warning is reported and the connection continues.

- **Session timeout (seconds)**
  This value tells the FTP client the maximum number of seconds to wait for data packets being transferred to or from the host. If nothing is received within the period specified, a timeout error displays and the transfer terminates; in this case, try the operation again. If you receive repeated timeout errors, increase the timeout value. Entering 0 (zero) in this box prevents the FTP client from ever timing out when waiting for a response. For SFTP sessions, the default is 0 (zero).

- **Keep Alive time (seconds)**
  Select this option and enter a time in seconds if you want to continue your connection to the server beyond the server’s automatic timeout value for inactivity. Most servers have an idle time value that specifies how long a user’s FTP session can last when no activity is detected. When the user exceeds the time limit, the server connection is closed.

  This setting allows you to direct the FTP client to send a NOOP command to the server at timed intervals to prevent the server from closing the connection due to inactivity. Be aware that by continuing your session you may prevent another user from making a connection to the FTP server.

- **Anonymous user**
  Select this option to log onto the specified FTP server as a guest, with the user name “Anonymous”.
• **Host encoding**

Specifications the character set used by the host to display the names of files that are transferred. By default, Reflection ZFE uses UTF-8 (Unicode). If you transfer files with the default setting and the file names are unrecognizable, change the Host encoding option to the character set used by the host. (This option does not affect the encoding for the contents of the files that are transferred; it applies to the file names only.)

---

**Related Topics**

File and Data Transfer

---

**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.

<table>
<thead>
<tr>
<th>Key Combination</th>
<th>Host type</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + A</td>
<td>UTS, 3270, 5250</td>
<td>Select all</td>
</tr>
<tr>
<td>Shift + Arrow key</td>
<td>UTS, 3270, 5250, VT</td>
<td>Change the extent of the current selection</td>
</tr>
<tr>
<td>Ctrl + C</td>
<td>UTS, 3270, 5250</td>
<td>Copy</td>
</tr>
<tr>
<td>Ctrl + V</td>
<td>UTS, 3270, 5250</td>
<td>Paste</td>
</tr>
<tr>
<td>Ctrl + Shift + A</td>
<td>VT</td>
<td>Select all</td>
</tr>
<tr>
<td>Ctrl + Shift + C</td>
<td>VT</td>
<td>Copy</td>
</tr>
<tr>
<td>Ctrl + Shift + V</td>
<td>VT</td>
<td>Paste</td>
</tr>
</tbody>
</table>
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 own 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
Configure Display Settings
Customize host screen colors
Configure hotspots
Map keys
Configure macros for your users
Transfer files
Use Plus to customize screens
3270 Host Printing

Customize your session

There are two features you can use to customize sessions for your end users:

- **Plus** - Enable custom controls to provide a more efficient work flow and a more modern and friendly interface. See Use Plus to customize screens.

  Using this option, you can add tool tips to fields, replace old-style numbered lists with more modern drop-down lists, add buttons to the host interface and program them to start macros or perform other actions, and replace manual date entry with a graphical calendar date-picker.

- **Server-side Events** - Supply procedural Java code that extends and improves the presentation of host data.

  Using server side events, you can define specific events and suspend the host application, replacing or interrupting it with code that you have supplied to the session, as well as extend error handling options. For example, you can add an event that recognizes when an error occurs and then implements the code to intercept the error, take control, and correct the error. See Use server side events.
Both of these options are configured on the Customization panel.

1. Click Settings on the toolbar to open the left navigation panel.
2. Click Customization.

---

Related Topics

Use Plus to customize screens
Use server side events

Use Plus to customize screens

**NOTE:** The Plus feature requires customization files (.rdar) produced by Micro Focus Screen Designer version 9.5 or higher. The Screen Designer is available in Micro Focus Rumba Desktop 9.5. Reflection Desktop 16.1 includes a limited version of the Screen Designer and Micro Focus Plus. To get access to more controls and full use of Plus and the Screen Designer, you can purchase and install the Micro Focus Reflection Desktop Plus add-on.

1. On the Customization panel, click Enable Plus.
2. Select the customization file you want to use from the drop down list or upload a file from a different location. Customization files are identified by a rdar file extension.
   
   Customization (.rdar) files are archive files and the output of a Screen Designer project and are used to provide the custom control criteria. For more information on the Micro Focus Screen Designer there are instructional videos available here: Screen Designer videos.
   
   If you are updating the customization (.rdar) file associated with your Plus enabled session, you must first delete the folder containing the old .rdar file from the session server. After you delete the folder, you can open your Plus enabled session and the new rdar file will be downloaded to the session server.
3. Verify the number of milliseconds for the host settle delay time is accurate. This is the time that the server waits for a synchronous connection before deciding that the host has finished sending data.
4. When you return to your session, Plus is available. Click on the toolbar to turn off the custom controls.

When you enable Plus for a session, all end users of that session see the Plus icon on the toolbar and any controls made available through the Screen Designer customization file.

---

Related Topics

Customize your session
Screen Designer videos
Use server side events

Using server side events, you can supply procedural Java code that can extend and improve the presentation of host data.

1. Make the Reflection ZFE SDK available to your development environment. The SDK is available at `install-dir\Micro Focus\ReflectionZFE\sessionserver\sdk`.

2. Write the Java code necessary to accomplish the task and compile the code into a Java class within a JAR (Java Archive) file.

3. Copy the JAR file to `install-dir\Micro Focus\ReflectionZFE\sessionserver\webapps\zfe\WEB-INF\lib` and restart the session server.
   If you have more than one session server on which you want the event to run, you must copy the JAR file to this location on each server.

4. Add the session you want to associate with the event in the Administrative Console.

5. As you configure the session in the Reflection ZFE web client, open the Customization panel.

6. Under Server Side Events type the fully qualified class name to the event.

7. Launch the session and test the event.

To help you create server side events, Reflection ZFE has an SDK and samples that are available to provide you with a starting point.

Access API documentation and event samples

Related Topics

- Customize your session
- Connectors and APIs
- Using the Reflection ZFE SDK
- Reflection ZFE SDK
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 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 7-1  3270 Quick Keys

<table>
<thead>
<tr>
<th>Keys</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Attention Keys</td>
<td>Select to send a program attention key to the host.</td>
</tr>
<tr>
<td>PA1-PA3</td>
<td></td>
</tr>
<tr>
<td>Program Function Keys</td>
<td>Select to send a program function key to the host.</td>
</tr>
<tr>
<td>PF1-PF24</td>
<td></td>
</tr>
<tr>
<td>Attention</td>
<td>Select to send an Attention key to the host.</td>
</tr>
<tr>
<td>Enter</td>
<td>Select to send an Enter key to the host.</td>
</tr>
<tr>
<td>Erase</td>
<td>Select to erase all characters, from the cursor to the end of the entry.</td>
</tr>
<tr>
<td>Clear</td>
<td>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.</td>
</tr>
<tr>
<td>Reset</td>
<td>Clear the Input Inhibited indicator and reset the Insert mode.</td>
</tr>
<tr>
<td>SYSRQ</td>
<td>Sends the System Request key to the host.</td>
</tr>
</tbody>
</table>

### Table 7-2  5250 Quick Keys

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Function Keys</td>
<td>Select to send a program function key to the host.</td>
</tr>
<tr>
<td>PF1-PF24</td>
<td></td>
</tr>
<tr>
<td>Attention</td>
<td>Select to send an Attention key to the host.</td>
</tr>
<tr>
<td>Enter</td>
<td>Select to send an Enter key to the host.</td>
</tr>
<tr>
<td>Erase</td>
<td>Select to erase all characters, from the cursor to the end of the entry.</td>
</tr>
<tr>
<td>Clear</td>
<td>Signal the host to erase all user-entered text from the current screen.</td>
</tr>
<tr>
<td>Reset</td>
<td>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.</td>
</tr>
<tr>
<td>Roll Up</td>
<td>Select to scroll down one page in the current host screen. This option is equivalent to Page Down.</td>
</tr>
<tr>
<td>Roll Down</td>
<td>Select to scroll up one page in the current host screen. This option is equivalent to Page Up.</td>
</tr>
</tbody>
</table>
### 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
Printing

You can capture the contents of a screen and send it to a printer.

The settings available to you regarding page setup and orientation are dependent on your browser options.

To print the screen

The print screen option prints the contents of the terminal screen. It does not print the toolbar or other display information.

1. Navigate to the screen you want to print.
2. Click Print Screen on the toolbar.
3. Use your browser’s print dialog to select the printer and page setup options.

Related Topics

Customize your session
3270 Host Printing
IND$FILE file transfer

3270 Host Printing

This feature is available to 3270 host sessions. When you enable host printing, a separate 3287 session is launched and associated with the previously launched 3270 session.

The 3287 session builds a PDF file that contains the file to print and sends it to the Reflection ZFE web client. After receiving the file, the Reflection ZFE web client displays an alert which contains a link to download the PDF file.

The downloaded file is available in your browser's download folder.

NOTE: You can provide end users with the ability to print by setting the print User Preferences option.

To enable 3270 host printing

1. From a 3270 session, click Settings on the tool bar to open the left navigation panel.
2. In the left panel, click Print.
3. Select the protocol to use. The options are:
   - TN3270E - TN3270E of Telnet Extended, is for users of TCP/IP software who connect to their IBM mainframe through a Telnet gateway that implements RFC 1647.
   - TN3287 - TN3287 is for users of TCP/IP software who connect to their IBM mainframe through a Telnet gateway that implements RFC 1646.
Specify whether you want to use an Host LU Name or, if you select TN3270E, a TN Association, to link the terminal session with the print session. Select either:

- **Specify Host LU Name** - Specify the name of the host LU (logical unit) to use when the session connects to the host. The host LU is also known as the device name or the pool.

- **Use TN Association** - (TN3270E) If you choose to use a TN association, Reflection ZFE uses the device name specified in the connection settings to link the 3270 and 3287 session together.

5 Click **Save** to return to your session.

6 Send the print command to the host. To verify the host LU name, if needed, click on the tool bar.

   The 3287 session (that was created in the background) gets the command to print the file and builds the PDF file containing the file to print. This file is sent to the Reflection ZFE web client and is made available for download.

7 Click **OK** to download the file to your browser’s download location.

Use the **Print Job** button to flush your current print job. When you flush a print job, whatever has been accumulated so far is printed and the next task begins printing.

---

**Related Topics**

**Set User Preference Rules**
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 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 ➤.

You can also map keys that will automatically trigger an already recorded macro. In the Key Map settings dialog box, choose Run Macro from the Action drop down list. Choose a macro to associate with the key mapping from the Value list.

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.

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.
Creating 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>
<thead>
<tr>
<th>Browser</th>
<th>Open debugger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla Firefox 40.0.3</td>
<td>• From the toolbar, open the Menu, and choose Developer.</td>
</tr>
<tr>
<td></td>
<td>• From the Web Developer menu, choose Debugger. The debugger opens in a lower panel.</td>
</tr>
<tr>
<td>Google Chrome 45.0</td>
<td>• From the toolbar, open the Menu, and choose More tools.</td>
</tr>
<tr>
<td></td>
<td>• Choose Developer Tools to open the Debugger.</td>
</tr>
<tr>
<td>Microsoft Internet Explorer 11</td>
<td>• From the toolbar, open Settings, and choose F12 Developer Tools.</td>
</tr>
<tr>
<td></td>
<td>• Open the Debugger tab.</td>
</tr>
</tbody>
</table>

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 9-1  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 9-2  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 9-3  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
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
• FileTransferFactory
• FileTransfer
• FileTransferOptions
• HostFile
• HostFileType
• OIA
• OIAStatus
• Position
• SessionType
• StatusSet

**Attribute**

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

**Table 10-1 Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROTECTED</td>
<td>Indicates a protected data cell.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MODIFIED</td>
<td>Indicates a modified data cell.</td>
</tr>
<tr>
<td>NUMERIC_ONLY</td>
<td>Indicates the beginning of a numeric only data cell.</td>
</tr>
<tr>
<td>ALPHA_NUMERIC</td>
<td>Indicates an alpha numeric data cell.</td>
</tr>
<tr>
<td>HIGH_INTENSITY</td>
<td>Indicates whether the data cell contains high intensity text.</td>
</tr>
<tr>
<td>HIDDEN</td>
<td>Indicates whether the data cell contains hidden text</td>
</tr>
<tr>
<td>PEN_DETECTABLE</td>
<td>Indicates whether the data cell is pen detectable</td>
</tr>
<tr>
<td>ALPHA_ONLY</td>
<td>Indicates an alpha only data cell.</td>
</tr>
<tr>
<td>NUMERIC_SHIFT</td>
<td>Indicates the beginning of a numeric shift. field</td>
</tr>
<tr>
<td>NUMERIC_SPECIAL</td>
<td>Indicates the data cell marks the beginning of a numeric special field</td>
</tr>
<tr>
<td>KATAKANA_SHIFT</td>
<td>Indicates a section of Katakana text.</td>
</tr>
<tr>
<td>MAGNETIC_STRIP</td>
<td>Indicates the data cell marks the beginning of a magnetic strip field.</td>
</tr>
<tr>
<td>SIGNED_NUMERIC_ONLY</td>
<td>Indicates the data cell is a signed numeric field.</td>
</tr>
<tr>
<td>TRANSMIT_ONLY</td>
<td>Indicates the data cell is a transmit only field</td>
</tr>
<tr>
<td>FIELD_END_MARKER</td>
<td>Indicates the data cell marks the end of a modified field.</td>
</tr>
<tr>
<td>FIELD_START_MARKER</td>
<td>Indicates the data cell marks the start of a modified field.</td>
</tr>
<tr>
<td>SPECIAL_EMPHASIS_PROTECTED</td>
<td>Indicates a special emphasis protected field</td>
</tr>
<tr>
<td>TAB_STOP</td>
<td>Indicates that the data cell contains a tab stop.</td>
</tr>
<tr>
<td>REVERSE</td>
<td>Indicates the data cell displays in reverse video mode.</td>
</tr>
<tr>
<td>BLINKING</td>
<td>Indicates the data cell contains blinking text</td>
</tr>
<tr>
<td>RIGHT_JUSTIFIED</td>
<td>Indicates the data cell marks the beginning of a right justified field.</td>
</tr>
<tr>
<td>LEFT_JUSTIFIED</td>
<td>Indicates the data cell marks the beginning of a left justified field.</td>
</tr>
<tr>
<td>LOW_INTENSITY</td>
<td>Indicates the data cell contains low intensity text</td>
</tr>
<tr>
<td>UNDERLINE</td>
<td>Indicates the data cell contains underlined text</td>
</tr>
<tr>
<td>DOUBLE_BYTE</td>
<td>Indicates the data cell contains double byte text.</td>
</tr>
<tr>
<td>COLUMN_SEPARATOR</td>
<td>Indicates the data cell contains a column separator.</td>
</tr>
<tr>
<td>BOLD</td>
<td>Indicates the data cell contains bold text.</td>
</tr>
<tr>
<td>DOUBLE_WIDTH</td>
<td>Indicates the data cell marks a double width field.</td>
</tr>
<tr>
<td>DOUBLE_HEIGHT_TOP</td>
<td>Indicates a double height top data cell.</td>
</tr>
<tr>
<td>DOUBLE_HEIGHT_BOTTOM</td>
<td>Indicates a double height bottom data cell.</td>
</tr>
<tr>
<td>CONTROL_PAGE_DATA</td>
<td>Indicates the data cell contains control page data.</td>
</tr>
<tr>
<td>RIGHT_COLUMN_SEPARATOR</td>
<td>Indicates the data cell contains a right column separator.</td>
</tr>
<tr>
<td>LEFT_COLUMN_SEPARATOR</td>
<td>Indicates a data cell containing a left column separator.</td>
</tr>
<tr>
<td>UPPERSCORE</td>
<td>Indicates the data cell contains an upperscore.</td>
</tr>
</tbody>
</table>
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 10-2 AttributeSet**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRIKE_THROUGH</td>
<td>Indicates the data cell contains strike through text.</td>
</tr>
</tbody>
</table>

**METHODS**

- **contains(attribute)**
  - Determines if the set contains the specified Attribute.
  - **Parameters**
    - `{Number}` attribute to check
  - **Returns**
    - `{Boolean}` True if the attribute is in the set.

- **isEMPTY()**
  - Determines if the attribute set is empty.
  - **Returns**
    - `{Boolean}` True if the set is empty.

- **size()**
  - Indicates the number of attributes in a set.
  - **Returns**
    - `{Number}` The attribute count.

- **toArray()**
  - Converts the internal attribute set to an array.
  - **Returns**
    - `{Number[]}` Array of values of attributes in the set.

- **toString()**
  - Converts the internal attribute set to a string.
  - **Returns**
    - `{String}` Space-delimited names of attributes in the set.

- **forEach(callback, thisArg)**
  - 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}` thisArg optional pointer to a context object.
METHODS

forEachCallback(string, object)  A user provided callback function where you provide the behavior, to be used as the callback parameter to forEach.

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 10-3  Color constants

<table>
<thead>
<tr>
<th>Color</th>
<th>Description</th>
<th>Numeric Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLANK_UNSPECIFIED</td>
<td>No color specified</td>
<td>0</td>
</tr>
<tr>
<td>BLUE</td>
<td>Blue</td>
<td>1</td>
</tr>
<tr>
<td>GREEN</td>
<td>Green</td>
<td>2</td>
</tr>
<tr>
<td>CYAN</td>
<td>Cyan</td>
<td>3</td>
</tr>
<tr>
<td>RED</td>
<td>Red</td>
<td>4</td>
</tr>
<tr>
<td>MAGENTA</td>
<td>Magenta</td>
<td>5</td>
</tr>
<tr>
<td>YELLOW</td>
<td>Yellow</td>
<td>6</td>
</tr>
<tr>
<td>WHITE_NORMAL_INTENSITY</td>
<td>Normal intensity white</td>
<td>7</td>
</tr>
<tr>
<td>GRAY</td>
<td>Gray</td>
<td>8</td>
</tr>
<tr>
<td>LIGHT_BLUE</td>
<td>Light blue</td>
<td>9</td>
</tr>
<tr>
<td>LIGHT_GREEN</td>
<td>Light green</td>
<td>10</td>
</tr>
<tr>
<td>LIGHT_CYAN</td>
<td>Light cyan</td>
<td>11</td>
</tr>
<tr>
<td>LIGHT_RED</td>
<td>Light red</td>
<td>12</td>
</tr>
<tr>
<td>LIGHT_MAGENTA</td>
<td>Light magenta</td>
<td>13</td>
</tr>
<tr>
<td>BLACK</td>
<td>Black</td>
<td>14</td>
</tr>
<tr>
<td>WHITE_HIGH_INTENSITY</td>
<td>High intensity white</td>
<td>15</td>
</tr>
<tr>
<td>BROWN</td>
<td>Brown</td>
<td>16</td>
</tr>
<tr>
<td>PINK</td>
<td>Pink</td>
<td>17</td>
</tr>
<tr>
<td>TURQUOISE</td>
<td>Turquoise</td>
<td>18</td>
</tr>
</tbody>
</table>
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 10-4  IBM 3270

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

**IBM 5250**

Table 10-5  IBM 5250

<table>
<thead>
<tr>
<th>Key word</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTVIEW</td>
<td>Alternate view</td>
</tr>
<tr>
<td>ATTN</td>
<td>Attention</td>
</tr>
<tr>
<td>AU1 - AU16</td>
<td>AU1 - AU16</td>
</tr>
<tr>
<td>BACKSPACE</td>
<td>Back space</td>
</tr>
<tr>
<td>BACKTAB</td>
<td>Back tab</td>
</tr>
<tr>
<td>BEGIN_FIELD</td>
<td>Begin field</td>
</tr>
<tr>
<td>CLEAR</td>
<td>Clear</td>
</tr>
<tr>
<td>DELETE_CHAR</td>
<td>Delete, delete character</td>
</tr>
<tr>
<td>DEST_BACK</td>
<td>Destructive backspace</td>
</tr>
<tr>
<td>DOWN</td>
<td>cursor down</td>
</tr>
<tr>
<td>DSPSOSI</td>
<td>Display SO/SI</td>
</tr>
<tr>
<td>DUP</td>
<td>Duplicate field</td>
</tr>
<tr>
<td>END_FILE</td>
<td>End of field</td>
</tr>
<tr>
<td>ENTER</td>
<td>Enter</td>
</tr>
<tr>
<td>ERASE_EOF</td>
<td>Erase end of field</td>
</tr>
<tr>
<td>ERASE_FIELD</td>
<td>Erase field</td>
</tr>
<tr>
<td>ERASE_INPUT</td>
<td>Erase input</td>
</tr>
<tr>
<td>FIELD_EXT</td>
<td>Field exit</td>
</tr>
<tr>
<td>Key word</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>FIELD_MINUS</td>
<td>Field minus</td>
</tr>
<tr>
<td>FIELD_PLUS</td>
<td>Field plus</td>
</tr>
<tr>
<td>FIELD_MARK</td>
<td>Field mark</td>
</tr>
<tr>
<td>HELP</td>
<td>Help request</td>
</tr>
<tr>
<td>HEXMODE</td>
<td>Hex mode</td>
</tr>
<tr>
<td>HOME</td>
<td>Cursor home</td>
</tr>
<tr>
<td>INSERT</td>
<td>Insert</td>
</tr>
<tr>
<td>LEFT_ARROW</td>
<td>Cursor left</td>
</tr>
<tr>
<td>NEW_LINE</td>
<td>New line</td>
</tr>
<tr>
<td>PA1 - PA3</td>
<td>PA1 - PA3</td>
</tr>
<tr>
<td>[PF1 - PF24</td>
<td>PF1 - PF24</td>
</tr>
<tr>
<td>[print]</td>
<td>Print</td>
</tr>
<tr>
<td>RESET</td>
<td>Reset, reset terminal</td>
</tr>
<tr>
<td>RIGHT_ARROW</td>
<td>Cursor right, right</td>
</tr>
<tr>
<td>PAGE_UP</td>
<td>Page up</td>
</tr>
<tr>
<td>PAGE_DOWN</td>
<td>Page down</td>
</tr>
<tr>
<td>SYSTEM_REQUEST</td>
<td>System request</td>
</tr>
<tr>
<td>TAB</td>
<td>Tab</td>
</tr>
<tr>
<td>UP</td>
<td>Cursor up</td>
</tr>
</tbody>
</table>

**VT**

Table 10-6 VT

<table>
<thead>
<tr>
<th>Keywords</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKSPACE</td>
<td>Back space</td>
</tr>
<tr>
<td>BREAK</td>
<td>Break</td>
</tr>
<tr>
<td>CLEAR</td>
<td>Clear or clear display</td>
</tr>
<tr>
<td>CURSOR_SELECT</td>
<td>Cursor select</td>
</tr>
<tr>
<td>DELETE_CHAR</td>
<td>Delete, delete character</td>
</tr>
<tr>
<td>DOWN</td>
<td>Cursor down</td>
</tr>
<tr>
<td>EK_FIND</td>
<td>Edit keypad find</td>
</tr>
<tr>
<td>EK_INSERT</td>
<td>Edit keypad insert</td>
</tr>
<tr>
<td>EK_NEXT</td>
<td>Edit keypad next</td>
</tr>
<tr>
<td>EK_PREV</td>
<td>Edit keypad previous</td>
</tr>
</tbody>
</table>
Keywords

EK_REMOVE
EK_SELECT
ENTER
END_FILE
F1 - F24
HOLD
HOME
INSERT
KEYPAD_COMMA
KEYPAD_DOT
KEYPAD_MINUS
KEYPAD_ENTER
KEYPAD0 - KEYPAD9
LEFT_ARROW:
PF1 - PF20
PAGE_DOWN
PAGE_UP
RESET
RETURN
RIGHT_ARROW
TAB
UDK16 - UDK20
UP

Description

Edit keypad remove
Edit keypad select
Enter
End of field
F1 - F24
Hold
Home
Insert
Keypad comma
Keypad decimal
Keypad minus
Keypad enter
Keypad 0 - Keypad 9
Cursor left
PF1 - PF20
Page down
Page up
Reset, reset terminal
Return, carriage return
Cursor right, right
Tab key
User defined key 6 - User defined key 20
Cursor up

UTS

Table 10-7  UTS

Key word

BACKSPACE
BACKTAB
CHAR_ERASE
CLEAR_DISPLAY
CLEAR_EOD
CLEAR_EOF

Description

Uses the Macro API

Moves the cursor to the previous tab position on the screen.
Back tab <Shift> <Tab>
Erases character at the cursor and advances the cursor.
Clear display
Clear to end of display
Clear to end of field
<table>
<thead>
<tr>
<th>Key word</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEAR_EOL</td>
<td>Clear to end of line</td>
</tr>
<tr>
<td>CLEAR_FCC</td>
<td>Clear Field Control Character</td>
</tr>
<tr>
<td>CLEAR_HOME</td>
<td>Clear display and cursor home</td>
</tr>
<tr>
<td>CONTROL_PAGE</td>
<td>Toggles the control page</td>
</tr>
<tr>
<td>DELETE_LINE</td>
<td>Deletes the line containing the cursor and shifts remaining lines up one row</td>
</tr>
<tr>
<td>DOWN</td>
<td>Moves the cursor down one line. Wraps at bottom.</td>
</tr>
<tr>
<td>DELIN_LINE</td>
<td>Deletes character under cursor and shifts remaining characters on line to the left.</td>
</tr>
<tr>
<td>DELIN_PAGE</td>
<td>Deletes character under cursor and shifts remaining characters on page to the left.</td>
</tr>
<tr>
<td>DUP_LINE</td>
<td>Creates a copy of the current line and overwrites the next line with the duplicate.</td>
</tr>
<tr>
<td>EURO</td>
<td>Inserts the Euro character</td>
</tr>
<tr>
<td>END_FIELD</td>
<td>Moves the cursor to the end of the current field.</td>
</tr>
<tr>
<td>END_PAGE</td>
<td>Moves the cursor to the end of the current page.</td>
</tr>
<tr>
<td>F1 - F22</td>
<td>Function keys F1-F22</td>
</tr>
<tr>
<td>HOME</td>
<td>Moves the cursor to beginning of current page (row 1, col 1)</td>
</tr>
<tr>
<td>INSERT</td>
<td>Toggles insert/overwrite mode.</td>
</tr>
<tr>
<td>INSERT_IN_LINE</td>
<td>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.</td>
</tr>
<tr>
<td>INSERT_IN_PAGE</td>
<td>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.</td>
</tr>
<tr>
<td>INSERT_LINE</td>
<td>Inserts a new line at the cursor row and shifts the remaining lines down. The last row on the page is discarded.</td>
</tr>
<tr>
<td>LEFT_ARROW</td>
<td>Moves the cursor one position to the left wrapping if necessary.</td>
</tr>
<tr>
<td>LOCATE_FCC</td>
<td>Finds the next field control character on the screen.</td>
</tr>
<tr>
<td>MSG_WAIT</td>
<td>Retrieves messages queued to the terminal.</td>
</tr>
<tr>
<td>RETURN</td>
<td>Carriage return</td>
</tr>
<tr>
<td>RIGHT_ARROW</td>
<td>Moves the cursor one position to the right, wrapping if necessary.</td>
</tr>
<tr>
<td>SOE</td>
<td>Inserts the Start of Entry character</td>
</tr>
<tr>
<td>START_OF_FIELD</td>
<td>Moves the cursor to the beginning of the field.</td>
</tr>
<tr>
<td>START_OF_LINE</td>
<td>Moves the cursor to column 1 of current line.</td>
</tr>
<tr>
<td>TAB</td>
<td>Moves the cursor to the next tab position of the screen.</td>
</tr>
<tr>
<td>TOGGLE_COL_SEP</td>
<td>Toggles the column separator attribute.</td>
</tr>
</tbody>
</table>
Key word  | Description  
--- | ---  
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 10-8 DataCell*

**METHODS**

**getPosition()**

Returns the position of this data cell on the screen.

Returns

{Position} the position of the data cell on the screen.

**getChar()**

Obtains the character associated with the cell.

Returns

{String} The character associated with the cell.

**getAttributes()**

Returns the set of attributes specified for this data cell instance. See AttributeSet.

Returns

{AttributeSet} Of attributes for this data cell instance.

**getForegroundColor()**

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.

**getBackgroundColor()**

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.

**toString**

Converts the internal data cell to a string.

Returns

{String} The string representation of a data cell.
**METHODS**

`isFieldDelimiter()` 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 10-9 Dimension**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Dimension(rows,cols)</code></td>
<td>Creates a new Dimension instance.</td>
</tr>
<tr>
<td>Parameters</td>
<td>{Number} rows screen rows dimension</td>
</tr>
<tr>
<td></td>
<td>{Number} cols screen columns dimension</td>
</tr>
</tbody>
</table>

---

**Field**

Use the Field object, along with FieldList, to obtain the information present in a field on the screen.

**Table 10-10 Field**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getAttributes()</code></td>
<td>Returns the set of attributes specified for this field instance. See AttributeSet.</td>
</tr>
<tr>
<td>Returns</td>
<td>{AttributeSet} The set of attributes for this field</td>
</tr>
<tr>
<td><code>getForegroundColor()</code></td>
<td>Returns the foreground color of the field.</td>
</tr>
<tr>
<td>Returns</td>
<td>{Number} the foreground color for this field. These values are defined in the Color object.</td>
</tr>
<tr>
<td><code>getBackgroundColor()</code></td>
<td>Returns the background color of the field.</td>
</tr>
<tr>
<td>Returns</td>
<td>{Number} the background color for this field. These values are defined in the Color object.</td>
</tr>
</tbody>
</table>
**Method**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getStart()</code></td>
<td>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.</td>
</tr>
<tr>
<td>Returns</td>
<td><code>{Position}</code> Starting position of the field.</td>
</tr>
<tr>
<td>Throws</td>
<td><code>{RangeError}</code> For zero length fields.</td>
</tr>
<tr>
<td><code>getEnd()</code></td>
<td>Returns the ending position of the field. The ending position is the position in the presentation space containing the last character of the field.</td>
</tr>
<tr>
<td>Returns</td>
<td><code>{Position}</code> Ending position of the field.</td>
</tr>
<tr>
<td>Throws</td>
<td><code>{RangeError}</code> For zero length fields.</td>
</tr>
<tr>
<td><code>getLength()</code></td>
<td>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.</td>
</tr>
<tr>
<td>Returns</td>
<td><code>{Number}</code> Length of the field.</td>
</tr>
<tr>
<td><code>getDataCells()</code></td>
<td>Obtains the data cells that comprise this field. See DataCell.</td>
</tr>
<tr>
<td>Returns</td>
<td><code>{DataCell[]}</code> Data cells that comprise this field.</td>
</tr>
<tr>
<td><code>getText()</code></td>
<td>Obtains the text from the field.</td>
</tr>
<tr>
<td>Returns</td>
<td><code>{String}</code> field text.</td>
</tr>
<tr>
<td><code>setText()</code></td>
<td>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.</td>
</tr>
<tr>
<td>Parameters</td>
<td><code>{String}</code> Text to set on the field.</td>
</tr>
<tr>
<td>Throws</td>
<td><code>{Error}</code> If the field is protected.</td>
</tr>
</tbody>
</table>
FieldList

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clearField()</td>
<td>Clears the current field in an emulation-specific manner.</td>
</tr>
<tr>
<td></td>
<td><strong>Throws</strong></td>
</tr>
<tr>
<td></td>
<td>{Error} If the field is protected or clear is not supported.</td>
</tr>
<tr>
<td>getPresentationSpace()</td>
<td>Obtains the presentation space which created this field.</td>
</tr>
<tr>
<td></td>
<td><strong>Returns</strong></td>
</tr>
<tr>
<td></td>
<td>{PresentationSpace} Parent of this field instance.</td>
</tr>
<tr>
<td>toString()</td>
<td>Creates a user-friendly description of the field.</td>
</tr>
<tr>
<td></td>
<td><strong>Returns</strong></td>
</tr>
<tr>
<td></td>
<td>{String} A user readable rendition of the field.</td>
</tr>
</tbody>
</table>

FieldList

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getPresentationSpace()</td>
<td>Obtains the presentation space which created this field list.</td>
</tr>
<tr>
<td></td>
<td><strong>Returns</strong></td>
</tr>
<tr>
<td></td>
<td>{PresentationSpace} Parent of this field list instance.</td>
</tr>
</tbody>
</table>
Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>findField(position, text, direction)</td>
<td>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.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Position from which to start the search. See Position object.</td>
</tr>
<tr>
<td></td>
<td>The text to search for (optional). If not provided, returns the next field to the right of or below the specified position.</td>
</tr>
<tr>
<td></td>
<td>Direction of the search (optional). Use PresentationSpace.SearchDirection constants for this parameter. For example, PresentationSpace.SearchDirection.FORWARD or PresentationSpace.SearchDirection.BACKWARD. If not provided, searches forward.</td>
</tr>
<tr>
<td>Returns</td>
<td>Field containing the string or null if a field meeting the given criteria is not found.</td>
</tr>
<tr>
<td>Throws</td>
<td>RangeError if the position is out of range.</td>
</tr>
<tr>
<td>get(index)</td>
<td>Obtains the field at the given index.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Index into the field list.</td>
</tr>
<tr>
<td>Returns</td>
<td>Field located at the specified index.</td>
</tr>
<tr>
<td>Throws</td>
<td>RangeError if the index is out of range.</td>
</tr>
<tr>
<td>isEmpty()</td>
<td>Determines if the field list is empty.</td>
</tr>
<tr>
<td>Returns</td>
<td>True if the list is empty.</td>
</tr>
<tr>
<td>size()</td>
<td>Indicates the number of fields in the list.</td>
</tr>
<tr>
<td>Returns</td>
<td>The field count</td>
</tr>
<tr>
<td>toString()</td>
<td>Creates a user-friendly description of the field list.</td>
</tr>
<tr>
<td>Returns</td>
<td>User readable rendition of the field list.</td>
</tr>
</tbody>
</table>
FileTransferFactory

A fileTransferFactory object is available to all macros. If file transfers are configured for the session, you can use it to get a reference to a FileTransfer object.

Table 10-12  fileTransferFactory

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getIND$File()</td>
<td>Returns a FileTransfer object for interacting with the configured Ind$File type for the session.</td>
</tr>
<tr>
<td></td>
<td>Returns {FileTransfer}</td>
</tr>
<tr>
<td></td>
<td>Throws {Error} if the session hasn't been configured to allow IND$File transfers.</td>
</tr>
</tbody>
</table>

FileTransfer

Use the FileTransfer object to list and transfer files between the host system and the client.

The Reflection ZFE file transfer API abstracts the file path conventions used by different host file implementations. Follow URL or Linux file system path formats when formatting file paths used by the API. For example, /root/directory/file. It is important to observe any rules specific to host systems, such as allowable characters or name lengths.

NOTE: Browsers place significant security restrictions around the ability of Javascript to interact with client file systems.

Table 10-13  FileTransfer

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getHostFileListing(remotePath)</td>
<td>Request a listing of host files. If remotePath is omitted, a file listing for the current remote working directory is shown.</td>
</tr>
<tr>
<td></td>
<td>Parameters</td>
</tr>
<tr>
<td></td>
<td>(String) (optional) If specified will get file listing for specified remote path. If not specified, will get file listing for current remote working directory.</td>
</tr>
<tr>
<td></td>
<td>Returns</td>
</tr>
<tr>
<td></td>
<td>(Promise) Resolves to an array of HostFile objects contained at remoteName. Rejected if the remote path can not be read.</td>
</tr>
</tbody>
</table>
### Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sendFile(localFile, remoteName)</code></td>
<td>Sends specified file to the host.</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td></td>
</tr>
<tr>
<td><code>{File}</code> Javascript file object pointing to local file to send.</td>
<td></td>
</tr>
<tr>
<td><code>{String}</code> Fully-qualified remote file name as allowed by remote system (Unix, Windows, MVS, VAX).</td>
<td></td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td></td>
</tr>
<tr>
<td><code>{Promise}</code> fulfilled with a HostFile object representing the sent file on success. Rejected if an error occurred sending the file.</td>
<td></td>
</tr>
<tr>
<td><code>getDownloadURL(remoteName)</code></td>
<td>Constructs a link to download a file from a host system.</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td></td>
</tr>
<tr>
<td><code>{String}</code> Fully-qualified remote file name as allowed by remote system (Unix, Windows, MVS, VAX).</td>
<td></td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td></td>
</tr>
<tr>
<td><code>{URL}</code> that can be used to retrieve the file from the Reflection ZFE session server.</td>
<td></td>
</tr>
<tr>
<td><code>setTransferOptions(options)</code></td>
<td>Set transfer options for current FileTransfer session. The transfer options are applied to all future transfers until the session is either exited or overridden by another call to <code>setTransferOptions</code>.</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td></td>
</tr>
<tr>
<td><code>{JSON}</code> see FileTransferOptions for allowed names and values.</td>
<td></td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td></td>
</tr>
<tr>
<td><code>{Promise}</code> fulfilled when the call completes. Rejected if an error occurred setting the options.</td>
<td></td>
</tr>
<tr>
<td><code>cancel()</code></td>
<td>Cancels the current transfer in progress.</td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td></td>
</tr>
<tr>
<td><code>{Promise}</code> fulfilled when the call completes. Rejected if an error occurred canceling the transfer.</td>
<td></td>
</tr>
</tbody>
</table>

### HostFile

A HostFile object represents a file on the host file system.
Table 10-14  HostFile

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getName()</td>
<td>Gets the file name</td>
</tr>
<tr>
<td></td>
<td>Returns {String} the file name.</td>
</tr>
<tr>
<td>getParent()</td>
<td>Gets the parent of this host file</td>
</tr>
<tr>
<td></td>
<td>Returns {String} the parent of this host file.</td>
</tr>
<tr>
<td></td>
<td>This means different things on different host types. For example on TSO this is the name of the catalog in which the file resides.</td>
</tr>
<tr>
<td>getSize()</td>
<td>The byte size of the file</td>
</tr>
<tr>
<td></td>
<td>Returns {Number} the size of the file in bytes.</td>
</tr>
<tr>
<td>getType()</td>
<td>The type of file represented</td>
</tr>
<tr>
<td></td>
<td>Returns</td>
</tr>
</tbody>
</table>

Host File Type

The HostFileType object defines constants for determining the type of a HostFile object.

Table 10-15  HostFileType

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>Represents a file on the host system.</td>
</tr>
<tr>
<td>DIR</td>
<td>Represents a directory on the host system.</td>
</tr>
<tr>
<td>UNKNOWN</td>
<td>Represents a host file of unknown origin.</td>
</tr>
</tbody>
</table>

File Transfer Options

File transfer option object specification.

Example: fileTransfer.setTransferOptions({ transferMethod : 'ascii' });

Table 10-16  FileTransferOptions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transferMethod</td>
<td>(String) Allowed values:</td>
</tr>
<tr>
<td></td>
<td>• 'ascii'</td>
</tr>
<tr>
<td></td>
<td>• 'binary'</td>
</tr>
</tbody>
</table>
**OIA**

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

**Table 10-17  OIA**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getStatus()</td>
<td>Returns the set of enabled status flags. See StatusSet.</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td>{StatusSet} Containing the status information.</td>
</tr>
<tr>
<td>getCommErrorCode()</td>
<td>Returns the current communication error code.</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td>{Number} the current communication error code. If one doesn’t exist, it will be 0.</td>
</tr>
<tr>
<td>getProgErrorCode()</td>
<td>Returns the current program error code</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td>{Number} the current program error code. If one doesn’t exist, it will be 0.</td>
</tr>
</tbody>
</table>

**OIAStatus**

**Table 10-18  OIAStatus**

<table>
<thead>
<tr>
<th>OIAStatus</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTROLLER_READY</td>
<td>Controller ready</td>
</tr>
<tr>
<td>A_LINE</td>
<td>Online with a non-SNA connection</td>
</tr>
<tr>
<td>MY_JOB</td>
<td>Connected to a host application</td>
</tr>
<tr>
<td>OP_SYS</td>
<td>Connected to a SSCP (SNA)</td>
</tr>
<tr>
<td>UNOWNED</td>
<td>Not connected</td>
</tr>
<tr>
<td>TIME</td>
<td>Keyboard inhibited</td>
</tr>
<tr>
<td>SYS_LOCK</td>
<td>System lock following AID key</td>
</tr>
<tr>
<td>COMM_CHECK</td>
<td>Communication check</td>
</tr>
<tr>
<td>PROG_CHECK</td>
<td>Program check</td>
</tr>
<tr>
<td>ELSEWHERE</td>
<td>Keystroke invalid at cursor location</td>
</tr>
<tr>
<td>FN_MINUS</td>
<td>Function not available</td>
</tr>
<tr>
<td>WHAT_KEY</td>
<td>Keystroke invalid</td>
</tr>
</tbody>
</table>
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 10-19  AutoSignon**

**Method**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getPassTicket()</code></td>
<td>Obtains a pass ticket to be used for signing onto a mainframe application. Multiple pass tickets may be requested using different application IDs.</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td></td>
</tr>
<tr>
<td><code>{String}</code> application ID</td>
<td>tells the host which application the sign on is for</td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td></td>
</tr>
<tr>
<td><code>{Promise}</code> 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.</td>
<td></td>
</tr>
<tr>
<td><code>sendUserName()</code></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td></td>
</tr>
<tr>
<td><code>{String}</code> passTicketKey</td>
<td>obtained from <code>getPassTicket</code></td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td></td>
</tr>
<tr>
<td><code>{Promise}</code> fulfilled if the user name is successfully sent. Rejected if the operation fails.</td>
<td></td>
</tr>
</tbody>
</table>
Position

Represents a row and column on the screen.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sendPassword()</td>
<td>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.</td>
</tr>
<tr>
<td>Parameters</td>
<td>[String] passTicketKey obtained from getPassTicket</td>
</tr>
<tr>
<td>Returns</td>
<td>[Promise] fulfilled if the password is successfully sent. Rejected if the operation fails.</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position(row,col)</td>
<td>Creates a new Position instance.</td>
</tr>
<tr>
<td>Parameters</td>
<td>{Number} row screen row coordinate</td>
</tr>
<tr>
<td></td>
<td>{Number} col screen column coordinate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>METHOD</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getCursorPosition()</td>
<td>Returns a Position instance representing the current cursor position. An unconnected session has a cursor position of 0,0.</td>
</tr>
<tr>
<td>Returns</td>
<td>{Position} current cursor location</td>
</tr>
</tbody>
</table>
METHODS

`setCursorPosition(position)` 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**

- `{Position} Position new cursor position.

**Returns**

- None

**Throws**

- `{RangeError} If the position is not valid on the current screen.

`isCursorVisible()` Tests that the cursor is currently visible in the presentation space. The cursor is considered not visible if the session is not connected.

**Returns**

- `{Boolean} True if the cursor is visible. False if the cursor is not visible.

`sendsKeys(keys)` 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 `setCursorPosition` function first.

The text string can contain any number of characters and `ControlKeys`

For example: "myname" + ControlKey.TAB + "mypass" + ControlKey.ENTER will transmit a user ID, tab to the next field, transmit a password, and then transmit the Enter key.

If you need to transmit a square bracket, double the brackets (|[ or ]|).

**Parameters**

- `{String} keys text and/or control keys to transmit

`getText(position, length)` Returns a string representing a linear area of the presentation space. No new line characters are inserted if row boundaries are encountered.

**Parameters**

- `{Position} start position from which to retrieve text

- `{Number} 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.

**Returns**

- `{String} representing a linear area of the presentation space which may be empty if the session is not connected.

**Throws**

- `{RangeError} If the position or length are not valid on the current screen.
**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>
<thead>
<tr>
<th>Table</th>
<th>Session object functions</th>
</tr>
</thead>
</table>

**METHODS**

**connect()**

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

**Returns**

None
METHODS

**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

**getPresentationSpace()**

Provides access to the PresentationSpace instance for this session.

**Returns**

{PresentationSpace} instance associated with this session.

**getDeviceName()**

Returns the connected available device name, the configured device name, or null if no device name is configured.

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.

**Returns**

{String} The connected device name, the configured device name, or null.

**getType()**

Returns the type of host session. See SessionType.

**Returns**

{String} The type of host session.

**setDeviceName()**

Provides a means to modify the device name on a session instance.

**Parameters**

{String} name Device name to use when connecting to a host.

**Throws**

{Error} If an attempt is made to set the device name while the session is connected.

**getOIA()**

Provides access to the Operator Information Area (OIA) instance for this session.

**Returns**

{OIA} Associated with this session

---

**SessionType**

Constants used to identify the type of host to which the connection is being made. See Session object.
### Table 10-23  SessionType

<table>
<thead>
<tr>
<th>Host Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM_3270</td>
<td>Indicates an IBM 3270 terminal session.</td>
</tr>
<tr>
<td>IBM_5250</td>
<td>Indicates an IBM 5250 terminal session.</td>
</tr>
<tr>
<td>VT</td>
<td>Indicates a VT session.</td>
</tr>
</tbody>
</table>

### 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 10-24  StatusSet

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contains(statusFlag)</td>
<td>Determines if the set contains the specified status flag from OIAStatus constants.</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{Number} statusFlag status to check</td>
</tr>
<tr>
<td>Returns</td>
<td>{Boolean} True if the status flag is present in the set.</td>
</tr>
<tr>
<td>isEmpty()</td>
<td>Determines if the status set is empty.</td>
</tr>
<tr>
<td>Returns</td>
<td>{Boolean} True if the set is empty.</td>
</tr>
<tr>
<td>size()</td>
<td>Indicates the number of status flags in the set.</td>
</tr>
<tr>
<td>Returns</td>
<td>{Number} The status count</td>
</tr>
<tr>
<td>toArray()</td>
<td>Converts the internal status set to an array.</td>
</tr>
<tr>
<td>Returns</td>
<td>{Object []} Array of status flags in the set.</td>
</tr>
<tr>
<td>toString()</td>
<td>Converts the internal status set to a string.</td>
</tr>
<tr>
<td>Returns</td>
<td>{String} Space delimited names of status flags in the set.</td>
</tr>
<tr>
<td>forEach(callback, thisArg)</td>
<td>Function to iterate over each element in the status set.</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{forEachCallback} Callback to perform the specific operation.</td>
</tr>
<tr>
<td></td>
<td>Called with the name of each status in the set.</td>
</tr>
<tr>
<td></td>
<td>{Object} thisArg optional pointer to a context object.</td>
</tr>
</tbody>
</table>
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 10-25  User Interaction

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>forEachCallback(string, thisArg)</td>
<td>A user provided callback function where you provide the behavior, to be used as the callback parameter to forEach.</td>
</tr>
</tbody>
</table>

**Parameters**

- **String** String The name of a status in the status set.
- **Object** thisArg Optional pointer to a context object

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prompt(message,[defaultAnswer],[mask])</td>
<td>Prompt the user for information in the user interface,</td>
</tr>
</tbody>
</table>

**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”.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message([message])</td>
<td>Display a message in the user interface.</td>
</tr>
</tbody>
</table>

**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>
<thead>
<tr>
<th>Table 10-26  Waiting for the host</th>
</tr>
</thead>
</table>

**METHODS**

- **setDefaultTimeout(timeout)**
  
  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.

- **forConnect([timeout])**
  
  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.

- **forDisconnect([timeout])**
  
  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.
<table>
<thead>
<tr>
<th>METHOD</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>forFixedTime([timeout])</td>
<td>Waits unconditionally for fixed time. Time is in milliseconds (ms)</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>{Number} timeout in milliseconds.</td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td>{Promise} Fulfilled after time elapses</td>
</tr>
<tr>
<td>forCursor(position, [timeout])</td>
<td>Waits for the cursor to arrive at the specified position.</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>{Position} The position specifying the row and column,</td>
<td></td>
</tr>
<tr>
<td>{Number} timeout in milliseconds</td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td>{Promise} Fulfilled if the cursor is already located or when it is finally located. Rejected if the wait times out.</td>
</tr>
<tr>
<td>forText(string, position, [timeout])</td>
<td>Wait for text located at a specific position on the screen</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>{String} text to expect</td>
<td></td>
</tr>
<tr>
<td>{Position} position specifying the row and column</td>
<td></td>
</tr>
<tr>
<td>{Number} timeout in milliseconds</td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td>{Promise} Fulfilled if the text is already at the specified position or whenever it is located. Rejected if the wait times out.</td>
</tr>
<tr>
<td>Throws</td>
<td>{rangeError} if the position is not valid.</td>
</tr>
<tr>
<td>forHostPrompt(string, column, [timeout])</td>
<td>Waits for a command prompt located at a particular column on the screen.</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>{String} text prompt to expect</td>
<td></td>
</tr>
<tr>
<td>{Number} column where cursor is expected</td>
<td></td>
</tr>
<tr>
<td>{Number} timeout in milliseconds.</td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td>{Promise} Fulfilled if the conditions are already met or when the conditions are finally met. Rejected if the wait times out.</td>
</tr>
<tr>
<td>Throws</td>
<td>{rangeError} if the column is out of range.</td>
</tr>
</tbody>
</table>
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
- User Interaction
- Paging Through Data
- Invoking a Web Service
- Working with DataCells and Attributes
- Using Fields and Field Lists
- Automatic Sign-On Macro for Mainframes
- Using File Transfer (IND$File)

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.

```javascript
// 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.
        }
    } 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
});
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.
Using the Macro API

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.

```javascript
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) {
      // Display account details here...
    }
  }
```
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 "proxy1/model/CICSAcctsDemo/GetAccountDetail" would be routed to "http://remote-vhi-server:9680/vhi-rs/model/CICSAcctsDemo/GetAccountDetail").
-->
<servlet>
  <servlet-name>vhi-rs-proxy1</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>/proxy1</param-value>
  </init-param>
  <async-supported>true</async-supported>
</servlet>

- 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.

```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 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.UNDERLINE]));
  }
});
```
Using the Macro API

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.

```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");
  }

  // 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).

```javascript
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);
    }
});

// Run the macro
return macro();
```
Using File Transfer (IND$File)

This series of sample macros demonstrate how to use the File Transfer API to retrieve a list of files, download a file, and upload a file to a 3270 host.

NOTE: You must be logged in and at a ready prompt before running these macros.

List files

This macro demonstrates how to use the File Transfer API to retrieve a list of files on a 3270 host using IND$File transfer. The IND$File transfer object is retrieved from the file transfer factory and then used to obtain an array of HostFile objects from either TSO or CMS.
var macro = createMacro(function*() {
  'use strict';

  try {
    var fileTransfer = fileTransferFactory.getInd$File();
    var hostFiles = yield fileTransfer.getHostFileListing();
    yield ui.message('Found ' + hostFiles.length + ' files');
    if (hostFiles.length > 0) {
      var firstFile = hostFiles[0];
      var msg1 = 'The catalog name is ' + firstFile.getParent() + '.  
      var msg2 = 'The first file is ' + firstFile.getName();
      yield ui.message(msg1 + msg2);
    } catch (error) {
      yield ui.message(error);
    }
  }

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

Download file
This macro shows how to use the File Transfer API to download a file from a 3270 host using
IND$File transfer. The IND$File transfer object is retrieved from the file transfer factory. In this
example the transfer method is set to ASCII to demonstrate use of the setTransferOptions function.
The sample macro downloads the first file returned from a call to getHostFileListing by creating a
download URI with a call to the getDownloadUrl function.

The macro can be used in either a CMS or TSO environment but the choice must be specified on the
first line or the code modified slightly for the intended system.

var hostEnvironment = 'CMS';  // 'TSO'
var prefix = fileNode.getParent() ? fileNode.getParent() + '/': '';
return prefix + fileNode.getName();

Download file
This macro shows how to use the File Transfer API to download a file from a 3270 host using
IND$File transfer. The IND$File transfer object is retrieved from the file transfer factory. In this
example the transfer method is set to ASCII to demonstrate use of the setTransferOptions function.
The sample macro downloads the first file returned from a call to getHostFileListing by creating a
download URI with a call to the getDownloadUrl function.

The macro can be used in either a CMS or TSO environment but the choice must be specified on the
first line or the code modified slightly for the intended system.

var hostEnvironment = 'CMS';  // 'TSO'
var prefix = fileNode.getParent() ? fileNode.getParent() + '/': '';
return prefix + fileNode.getName();

// Construct file path, ie catalog/file.name or catalog/partition/file
function getPath (fileNode) {
  var prefix = fileNode.getParent() ? fileNode.getParent() + '/': '';
  return prefix + fileNode.getName();
}

Download file
This macro shows how to use the File Transfer API to download a file from a 3270 host using
IND$File transfer. The IND$File transfer object is retrieved from the file transfer factory. In this
example the transfer method is set to ASCII to demonstrate use of the setTransferOptions function.
The sample macro downloads the first file returned from a call to getHostFileListing by creating a
download URI with a call to the getDownloadUrl function.

The macro can be used in either a CMS or TSO environment but the choice must be specified on the
first line or the code modified slightly for the intended system.

var hostEnvironment = 'CMS';  // 'TSO'
var prefix = fileNode.getParent() ? fileNode.getParent() + '/': '';
return prefix + fileNode.getName();

Download file
This macro shows how to use the File Transfer API to download a file from a 3270 host using
IND$File transfer. The IND$File transfer object is retrieved from the file transfer factory. In this
example the transfer method is set to ASCII to demonstrate use of the setTransferOptions function.
The sample macro downloads the first file returned from a call to getHostFileListing by creating a
download URI with a call to the getDownloadUrl function.

The macro can be used in either a CMS or TSO environment but the choice must be specified on the
first line or the code modified slightly for the intended system.

var hostEnvironment = 'CMS';  // 'TSO'
var prefix = fileNode.getParent() ? fileNode.getParent() + '/': '';
return prefix + fileNode.getName();
different browsers
    window.location = downloadUrl;

    // If you want to read the file contents into a variable instead of downloading
    // it, you can use jQuery
    // var fileContents = yield $.get(downloadUrl);

} catch (error) {
    yield ui.message(error);
}
});

// Run the macro
return macro();

Upload file
This macro illustrates how to use the File Transfer API to upload a file to a 3270 host using INDSFile
transfer. The sample macro prompts the user to choose a file from the local file system by triggering
the browser’s file selection dialog. It then retrieves the current catalog on TSO or drive identifier on
CMS by calling getHostFileListing. Finally, the sendFile function is called to deliver the selected local
file to the host.

The macro can be used in either a CMS or TSO environment but the choice should be specified on
the first line. In this example, the transfer method is set to ascii; you may want to change this to
binary.

var hostEnvironment = 'CMS'; // 'TSO'
// Open the browser's file chooser dialog programmatically
function promptForFileToUpload () {
    return new Promise(function (resolve, reject) {
        // We are not notified if the user cancels the file chooser dialog so reject after 30
        seconds
        var timerId = setTimeout(reject.bind(null, 'Timed out waiting for file selection'),
        30000);
        var fileSelector = document.createElement('input');
        fileSelector.setAttribute('type', 'file');
        fileSelector.onchange = function (evt) {
            var file = evt.target.files[0];
            clearTimeout(timerId);
            resolve(file);
        }
        fileSelector.click();
    });
}

var macro = createMacro(function* () {
    'use strict';
    try {
        var fileTransfer = fileTransferFactory.getIND$File();

        // The transferMethod options are 'binary' and 'ascii'
        fileTransfer.setTransferOptions({transferMethod: 'ascii'});

        var localFile = yield promptForFileToUpload();

        // Retrieve the current catalog name and append the selected file name to it
        var hostFiles = yield fileTransfer.getHostFileListing();
        var destination = hostFiles[0].getParent() + '/' + localFile.name;
if (hostEnvironment === 'CMS') {
    yield wait.forText('Ready', new Position(1,1), 5000);
}

var result = yield fileTransfer.sendFile(localFile, destination);

} catch (error) {
    yield ui.message(error);
}

// Run the macro
return macro();
File and Data Transfer

Reflection ZFE supports two different file transfer protocols; IND$FILE for 3270 host transfers and File Transfer Protocol (FTP) which allows a local computer to act as an FTP client. Once connected, you can view files on the server and use the File Transfer Protocol (FTP) to transfer files between your local computer (or any networked drive) and the FTP server.

Before you can transfer files, you (as the administrator) must enable the transfer option for the current session and make the necessary configurations. This is done on the File Transfer settings panel.

- IND$FILE file transfer
- FTP file transfer

IND$FILE file 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, CMS, or CICS host filing systems.

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

**NOTE:** When using CICS as the host system you must enter the names of the files you are transferring manually. A list of files to choose from is not available.

There is support for ASCII or binary transfers.

Transferring files

You must be connected and logged into 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 toolbar, 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. For CICS hosts, type in the names of the files you want to transfer.

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 left 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.

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 may be caused by browser security settings.

If a transfer completes but the file doesn’t contain the data expected, verify that the transfer method is properly set to either Binary or ASCII.

For host-specific errors, see IBM File Transfer Error Messages.
FTP file transfer

After the administrator configures a session to include FTP functionality, click to open the FTP File Transfer window containing a list of host files that are available to transfer. Directories and files are indicated by an icon when you select the file.

- Transferring files
- Downloading files
- Uploading files

Transferring files

1. 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.

2. You can rename, delete, or download a file from the list of files.

3. Refresh the file list at any time by clicking the Refresh icon in the upper left corner of the File Transfer dialog box.

Downloading files

1. From the list, select the file to initiate the transfer.
2. If necessary, you can cancel the transfer from the transfer progress panel.

Uploading files

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.
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.

Click *New Directory* to create a new directory on the remote server. You are prompted to enter the new directory name.
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).

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.
Connectors and APIs

The Reflection ZFE has a collection of APIs and libraries that help you develop efficient client/server and Web applications that integrate host data into various development environments.

Related Topics
Using the Reflection ZFE SDK
Customize your session

Using the Reflection ZFE SDK

Use server side events to supply procedural Java code that can extend and improve the presentation of host data. To help you create server side events, Reflection ZFE has an SDK and samples that are available to provide you with a starting point.

Examples and documentation

You can view the Reflection ZFE SDK.

To access the SDK for direct viewing and to import to your IDE:

1. Navigate to `<install-dir>`\Micro Focus\ReflectionZFE\sessionserver\sdk\java.
2. In the SDK directory, access:
   - `\javadoc`: This directory contains JavaDoc files for direct viewing.
   - `\samples`: This directory contains Java sources for direct viewing.
   - `\zfe-sdk.jar`: The JAR file contains the Java classes to import into your IDE.
   - `\zfe-sdk-javadoc.jar`: The JAR file contains JavaDoc files to import into your IDE.
In this section you can find information on specific issues that you may encounter. In the Micro Focus Technical Support Handbook you will find information about how to get technical support for your product, access to our online resources, and how to contact and work with our worldwide technical support organization.

- Copying Sessions between Management and Security Servers
- Macro Replication across Servers
- Configuring User Names when Using Anonymous Access Control
- Known Issues

Copying Sessions between Management and Security Servers

You can copy and convert Reflection for the Web sessions and make them available to another Management and Security Server (MSS) and Reflection ZFE.

**NOTE:** In the following procedure the Management and Security Server you are copying sessions from is the source, and the Management and Security Server you are copying to is the destination.

To copy sessions from the source server to the destination server follow these steps:

1. Stop the destination MSS server, if necessary.
2. On both source and destination MSS servers, open `SessionDS.xml`, located:
   - On Windows: `C:\ProgramData\Micro Focus\MSS\MSSData`
   - On Linux: `/var/opt/microfocus/mss/mssdata`
3. In the source XML file, locate the OBJECT_ARRAY element.
4. Still in the source XML file, under OBJECT_ARRAY, locate and copy the Reflection for the Web child `Session` elements.
5. Open the destination XML file and paste them under the destination file’s OBJECT_ARRAY element.
6. Still in the destination file, locate the OBJECT_ARRAY size attribute that corresponds to the number of sessions. Increase that value by the number of session elements you added. For example, if you pasted six `Session` elements in the destination file and the existing OBJECT_ARRAY size attribute value is 4; increase the value by six. The size attribute should now be ten. And you should now have 10 `Session` elements listed under the OBJECT_ARRAY element.
7. Session names must be unique. Check the destination file for duplicate session names. You can find session names in the `Session` child element, `SessionName`.
8 Copy the configuration files for every session added to SessionDS.xml from the source to the destination server. The names of the configuration files are located under the Session element in the child element, configuration. The files themselves are located:

- On Windows: C:\ProgramData\Micro Focus\MSS\MSSData\deploy\dyncfgs
- On Linux: /var/opt/microfocus/mss/mssdata/deploy/dyncfgs

9 If you stopped the destination MSS server, restart it. Open the Administrative Console. You should see all your copied sessions.

---

**Macro Replication across Servers**

Macros that are created by the end-user are not replicated between Management and Security Servers (MSS). Although Management and Security Servers can be deployed in a clustered environment where data is replicated across all servers in the deployment; macros created by the end-user for a session cannot be replicated.

To ensure that every Reflection ZFE session server has access to all end-user macros, it is important that every session server in a multi-server deployment environment points to a single Management and Security Server. To do this, specify the same Management and Security Server during the installation of the separate Reflection ZFE session servers.

You can back up the Reflection ZFE user macros by copying the <install-location>/Micro Focus/ReflectionZFE/managementserver/conf/prefs directory to another location.

---

**Configuring User Names when Using Anonymous Access Control**

Users need access to their macros, user configurations, and other personalized settings whether they are authenticated anonymously through Management and Security Server or not. Reflection ZFE uses user names to store user-specific information; but what happens when users are anonymously authenticated through Management and Security Server’s access control interface?

In a default environment, Reflection ZFE uses the session id of the HTTP session as the value for the user name. While this user name is unique for each browser session, it changes over time and when MSS is configured in anonymous mode, in order to consistently retrieve user settings, all users of that Reflection ZFE session necessarily share the same settings.

However, Reflection ZFE supports a number of ways that, as an administrator, you can configure a unique identifier for each user so their customized settings can be stored and retrieved.

---

**NOTE:** These configuration modifications do not alter the security considerations of using Management and Security Server in anonymous mode.

---

**Configuration options**

There are four different configuration options you can choose from when configuring user name identifiers. You must restart the Reflection ZFE session server before any changes take affect.

- **To use an HTTP request cookie value as the user name**

  Add the following lines to <session-server>/conf/container.properties:
zfe.principal.name.provider=com.microfocus.rzfe.webclient.security.rsg.CookieKeyNameAnonymousPrincipalNameProvider
zfe.principal.name.identifier=<the-cookie-key-to-be-used>

- **To use an HTTP request header value as the user name**
  Add the following lines to: `<session-server>/conf/container.properties`:

    zfe.principal.name.provider=com.microfocus.rzfe.webclient.security.rsg.HeaderKeyNameAnonymousPrincipalNameProvider
    zfe.principal.name.identifier=<the-header-key-to-be-used>

- **To use an HTTP request URL parameter as the user name**
  Add the following lines to: `<session-server>/conf/container.properties`:

    zfe.principal.name.provider=com.microfocus.rzfe.webclient.security.rsg.UrlParameterAnonymousPrincipalNameProvider
    zfe.principal.name.identifier=<the-url-parameter-key-to-be-used>

- **To use the client IP address as the user name**
  Add the following line to: `<session-server>/conf/container.properties`:

    zfe.principal.name.provider=com.microfocus.rzfe.webclient.security.rsg.RemoteIpAddressAnonymousPrincipalNameProvider

**Troubleshooting the configuration**

If any of your users experience problems when connecting to a Reflection ZFE web application after you have made the configuration changes, check the following:

- Users experience a **503 Service Unavailable** message when connecting to a Reflection ZFE web application. First check the log file (`<session-server>/logs/server.log`), then:

  -- If the log file contains this message: "Unable to create AnonymousPrincipalNameProvider instance for class...", then the `zfe.principal.name.provider` property is probably misspelled. Check the spelling and letter case to remedy this issue.

  -- If the log file contains this message: "zfe.principal.name.identifier is not defined", then the property is missing. Ensure the property is defined to remedy this issue.

- Users are unable to properly authenticate.
  Users should receive an error message indicating the initial HTTP request to the Reflection ZFE web application did not contain the required information.

**Known Issues**

These issues have been identified in previous releases and are known issues.

- **Browser issues**
- **Security issues**
- **Host specific issues**

**Browser issues**

The following notes are specific to different web browsers.
Recommended browsers

It is highly recommended that you use Google Chrome or Mozilla Firefox. While Reflection ZFE supports Microsoft Internet Explorer (IE) 10+, there are known performance issues with Internet Explorer’s JavaScript engine that may negatively affect the end user experience with Reflection ZFE.

- Internet Explorer Unable to Play Recorded Reflection ZFE Macros
- Internet Explorer Displays Blank Screens

There are remedies to both these issues, however the easiest solution is to use a different browser.

Key mapping issues with different browsers

Certain keys on a numeric keypad and some browser-specific keys cannot be mapped. For example, in Chrome, Ctrl+n and Ctrl+w cannot be mapped.

Security issues

These are noted security considerations.

SSL 3.0 is disabled in Reflection ZFE

For security reasons, enabling SSL 3.0 is not recommended. However, for hosts that absolutely require SSL 3.0, 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`.

Host specific issues

The following are issues that are specific to different host types.

Issues encountered with VT hosts

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance issues</td>
<td>- Heavy text output, such as form &quot;Is-IR&quot; may cause slow performance</td>
</tr>
<tr>
<td></td>
<td>- Scrolling regions may appear slow or choppy</td>
</tr>
<tr>
<td></td>
<td>- Cursor movement may be slow or choppy</td>
</tr>
<tr>
<td></td>
<td>- Internet Explorer is particularly slow, and performance degrades</td>
</tr>
<tr>
<td></td>
<td>further when used for rows and columns.</td>
</tr>
<tr>
<td>Character sets</td>
<td>- Graphical characters and some character sets are not supported.</td>
</tr>
<tr>
<td></td>
<td>- Some non-English characters may cause the terminal display to freeze.</td>
</tr>
<tr>
<td>Other VT issues</td>
<td>- Insert/delete column (DECIC, DECDC) may fail.</td>
</tr>
<tr>
<td></td>
<td>- VT400 will not recognize DECSCL.</td>
</tr>
</tbody>
</table>
Field outlines in 3270 sessions

The 3270 attributes for field outlines are not fully supported. Reflection ZFE currently supports underline and overline; however, left vertical line, right vertical line, and combinations of the four line types are not yet supported.