Additional Preference Settings

This section describes additional configuration settings for SOAtest and Virtualize available in the **Parasoft> Preferences** menu.

Browser Settings

The Browser panel lets you set options related to Web scenario recording. Available settings include:

- **Firefox Executable Path**: Specifies the path to the Firefox executable. On Windows machines, SOAtest and/or Virtualize will attempt to detect a Firefox installation automatically. Linux users will have to browse to the Firefox executable.
- **Chrome Executable Path**: Specifies the path to the Chrome executable. Paths set here will be used in web recording wizards and other applicable areas. On Linux, choose `google-chrome` (e.g. `/opt/google/chrome/google-chrome`)—not `chrome`.
- **Safari Executable Path**: Specifies the path to the Safari executable.
- **Proxy port**: Specifies the proxy port. See Proxy Configuration Details below for more information and tips.
- **Browser communication port**: Specifies the browser communication port.
- **Browser Timeout Settings**: Specifies the length of delay (in milliseconds) after which SOAtest and/or Virtualize should stop waiting for browser startup or a user action and consider it to be "timed out."
- **Wait Condition Default Timeout Settings**: Specifies the length of delay (in seconds) after which SOAtest and/or Virtualize should stop waiting for the activity specified in the wait condition to occur and consider it to be "timed out."
- **Debug Options> Print debugging information**: During recording of a web scenario, it is possible that an action taken is not recorded by SOAtest and/or Virtualize. Enabling this option will allow messages to be printed to the message console during recording, with information about what events SOAtest and/or Virtualize handled, any locators that may have been generated, and if applicable, any exceptions that took place during recording.
- **Error Reporting> Report website's scripting errors**: Configures SOAtest and/or Virtualize to report scripting errors that occur during scenario execution. In Internet Explorer, the Selenium Web-Driver framework will not report JavaScript errors on the page to SOAtest and/or Virtualize; this reporting is supported only for the legacy engine.
- **Allowable Binary Files in Traffic Viewer and Outputs**: Allows binary files with the specified extensions or MIME types to be used in the Traffic Viewer and output. By default, only text files will be allowed.
- **Browser Contents Viewer Rendering Engine**: Enables you to specify what browser is used for the Browsear Contents Viewer tool(described in Browser Contents Viewer) , which can be attached to the Browser Playback tool.
    - The default option **(Same browser used for playback)** is generally the recommended option because some web applications generate their pages differently based on the browser used. Using the same browser that was used during the playback can help ensure that pages display properly in the Browser Contents Viewer. Note that if the playback browser was Chrome, the Firefox rendering engine is used.
    - When **Internet Explorer** is selected, the version of IE that is used depends on what version of IE is installed on the machine running SOAtest and/or Virtualize. The Internet Explorer option is available only on Windows.
    - When **Firefox** is selected, the version of Firefox that is used depends on what Eclipse is being used to run SOAtest and/or Virtualize. It can range from Firefox 3.0.1 to Firefox 10, depending on what OS is being used.
    - If the particular web application being tested does not render properly in the Browser Contents Viewer, you could try configuring this option to use either Internet Explorer or Firefox specifically (rather than use the default **Same browser used for playback** option) to see if using a specific rendering engine will improve how the page is shown in the Browser Contents Viewer.
- **HTML Content Fetch Mode**: Enables you to determine whether the contents of hidden frames are displayed in the pre- and post-action HTML viewer. This can impact record and playback speed, as well as file size. It is possible to use one mode on some of your team's SOAtest and/or Virtualize machines (e.g., desktop installation), and a different mode on others (e.g, the Server machine running in command-line mode).

- **Fetch all HTML content** Choose this option if you want to see the contents of hidden frames in the pre- and post-action HTML viewer (in the Browser Playback tool, Browser Contents Viewer, Browser Data Bank, and Browser Validation tool). This is desirable if you want to create validations/extractions in hidden iframes. *This mode will significantly slow down recording and playback. Moreover, it will significantly increase .tst file size if your application uses hidden iframes.*
- **Fetch content for all content except hidden frames**: Choose this option if you do not need to see the contents of hidden frames in the pre- and post-action HTML viewer.(in the Browser Playback tool, Browser Contents Viewer, Browser Data Bank, and Browser Validation tool). In this mode, the browser will still retrieve all frames from the server and it can still perform validations and extractions on the hidden iframes. However, it will not display the contents of hidden frames in the pre- and post-action HTML viewer.

Proxy Configuration Details

When you record or run web scenarios in a browser, the proxy settings in the browser are set to an internal proxy maintained by SOAtest and/or Virtualize. All communication to and from the browser during recording and playback goes through the internal proxy, which is an intermediary used to capture traffic and otherwise facilitate execution. During recording and playback, SOAtest and/or Virtualize temporarily creates this proxy on localhost using the port specified by the Browser Playback setting's **Proxy port** option.

The default host and port for the internal proxy is localhost:55555. Change the port number if this port is already in use using the controls Proxy port field. Do not change this from within the browser.

If your machine is configured to use your own proxy, you should configure SOAtest and/or Virtualize to point to that proxy. This enables SOAtest and/or Virtualize to configure its internal proxy to forward all traffic to the specified proxy configured in Proxy Settings.

Internet Explorer Notes

SOAtest and Virtualize modify the global registry settings prior to starting its instance of the browser. If an instance of Internet Explorer was running on the machine prior to launching SOAtest or Virtualize (not recommended), the global registry settings will not be set in the existing browser instance.

In these cases, check the Internet Options panel in the existing browser instance while a web scenario is running to verify that the settings point to SOAtest's or Virtualize's proxy and click **OK** in the Internet Options panel. If you click **OK**, the proxy settings are updated in the existing browser instance. If you click **Cancel**, or do not go to the Internet Options panel, then the existing browser instance never picks up the proxy settings and should continue to navigate fine.

Proxy settings may not be reset properly if the browser exited abnormally, if there is a hanging browser process, etc. Such issues can affect new browser instances (or other programs that connect to the internet). If this happens, you can resolve it by resetting your machine's proxy settings to the appropriate settings or killing any hanging browser processes.

Console Settings

The Console panel allows you to determine the amount of information that is reported to the Console view and whether it is automatically activated when it contains messages.

Continuous Testing Platform Settings

If you have Continuous Testing Platform (CTP) and a valid license, you can configure your connection to CTP:

- **Use DTP settings:** Enable this option to use the connections settings specified in Development Testing Platform (DTP). See Connecting to Parasoft Development Testing Platform.
- **Server Name**: Specifies a name for the server you are connecting to CTP. This is the name that will be used to identify this server within CTP.
- **CTP URL**: Specifies the location of the CTP to which you are connecting (e.g., http://emdemo:8080).
- **Notify CTP of virtual asset deployment**: Determines whether the server notifies Parasoft CTP when virtual assets are first deployed.
- **Username**: Specifies the username for logging into CTP.
- **Password**: Specifies the password for logging into CTP.

Global Data Source Settings

Global Data Sources can be reused and shared outside of a single SOAtest project and across Virtualize deployments. The Global Data Source panel lets you determine how information about global data sources is saved.

Technical Support Settings

If you are experiencing problems with Virtualize, the best way to remedy the problem is to create a zip archive containing the related files, then send that zip file to Parasoft's support team. To facilitate this process, you can have the product automatically create an archive when problems occur. On average, these archives are about half a megabyte, and are created in about one minute.

By default, an archive is not created when problems occur. You can either manually prepare and send a support archive when needed, or you can modify Parasoft archive creation options so that the product automatically prepares and sends an archive when problems occur.

To configure the product to automatically prepare and send archives when problems occur:

1. Open the Technical Support panel by choosing **Parasoft> Preferences**, then selecting the **Parasoft> Technical Support** category.
2. Check **Enable auto-creation of support archives**.
3. Customize additional options as needed. Note that **Enable auto-creation of support archives** and **Send archives by email** are not applicable to Virtualize.
4. Click **Apply**, then **OK**.

To manually create a support archive:

- Choose **Parasoft> Preferences**, select the **Technical Support** category, select the desired archive options, then click **Create Archive**.

To open the Technical Support Archive Manager, which allows you to review, e-mail, or delete recent support archives:

- Choose **Parasoft> Preferences**, select the **Technical Support** category, then click **Browse Recent Archives**.

When creating a support archive it is best to ensure that it contains all the info which is relevant to the problem and does not contain any unrelated info. To ensure that detailed logs are sent to the standard output console, add the following argument to `virtualizecli`:

```
-J-Dcom.parasoft.xtest.logging.config.jar.file=/com/parasoft/xtest/logging/log4j/config/eclipse.on.xml
```

Dictionary Settings

The Dictionary panel allows you to customize the dictionary that the Spell tool uses to identify misspelled words.

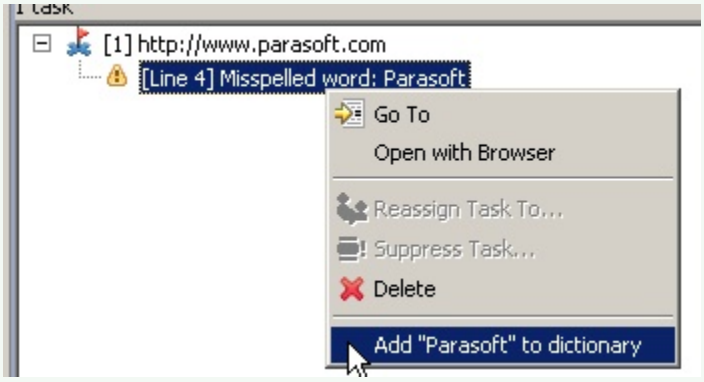
Adding Words

To add words to the dictionary:

- To add a new word, click **Add**, then enter it in the dialog that opens.
- To import a set of words from a text file, click **Impor**t, then specify the file that contains the set of words you want to import.
- To remove a word, select it in the list of words, then click **Delete**. You can select more than one word, then delete all selected words with one click.
- To export a list of words into a text file (for example, to export your User Added Words list so that your team members can import them) click **Expo rt**, then indicate what file you want to contain the exported words.

Adding Words from the Quality Tasks view

You can also add reported misspelled words to the dictionary from the Quality Tasks view. Just right-click the reported misspelled word, then choose **Add to Dictionary**.



Adding Dictionaries

You can expand SOAtest's built-in dictionary by extending it with additional sets of ispell-format dictionaries (such as dictionaries for language other than English, dictionaries of industry-specific terms, etc.). Each dictionary set has a name and one or more dictionaries.

To add an additional dictionary set:

1. Save the dictionaries in the SOAtest installation directory.
2. Click the **Add** button, then use the file chooser to select the set of dictionaries you want to add.

Adding Non-Text Characters or Words Containing Non-Text to the Dictionary

By default, SOAtest treats non-text characters as white space and does not allow you to add dictionary words that contain non-text characters. If you want SOAtest to consider a designated non-text character as a valid character within a word (rather than as one unit of white space), you need to add that character to the list of allowable non-text characters. This allows you to identify spelling errors in words that contain allowed non-text characters and to add dictionary words that contain allowed non-text characters.

To add non-text characters to the list of allowable non-text characters:

- Enter them in the **Allowable non-text characters** field. If you want to allow multiple non-text characters, list them one after the other—do not separate them with a space character, comma, or other delimiter.

MIME Type Settings

The MIME Types panel lets you add and remove MIME types. In addition, it lets you specify the location of your preferred text and XML editors and lets you specify what editor you want to use to edit all files that have a particular MIME type.

To add, edit, or remove a MIME type:

- To add a MIME type, click **Add MIME Type**, enter the new MIME type in the dialog box that opens, then enter the file extensions that you want to assign to this MIME type, and (optionally) indicate the implied MIME type by checking the appropriate check boxes. If you enter multiple extensions for a MIME type, separate the extensions with one space character.
- To edit a MIME type's settings, select the MIME type whose settings you want to edit, then modify the settings as needed.
- To remove a MIME type, select the MIME type that you want to remove, then click **Delete MIME Type**.

Miscellaneous Settings

The Misc panel allows you to set the following miscellaneous settings:

- **Show tool descriptions:** Enables/disables showing tool descriptions in applicable wizards.
- **Auto Beautify:** Tells SOAtest to automatically beautify XML messages in the selected tool or tools (Traffic Viewer, Diff, Editor) if the message is under the specified size (10 KB is the default setting).
- **Character Encoding:** You can enable **System default** to configure SOAtest and/or Virtualize to use the default character set for the particular system being used. Enable **Custom** to encode characters from the list of encodings available on the JVM being used.
- **Save settings:** Specifies what file format to use for saving project files (e.g., .pva, .pvn, .tst, .changetemplate).

  See the following sections for additional information about each type, including performance and forward compatibility:

  - Understanding the Available Project File Formats in SOAtest
  - Understanding the Available Project File Formats in Virtualize

- **Default timeout (milliseconds):** Allows you to enter the length of delay (in milliseconds) after which SOAtest should consider your FTP, telnet, or HTTP requests to be "timed out." The default is 30000 milliseconds.
- **Report each duplicate error that occurs on the same line:** Tells SOAtest to show only the first instance of duplicate errors that occur for the same line of code.
- **Reset Cookies:** Allows you to clear the current global cookies so that next HTTP invocations start a new session.
- Enable the **Automatically backup project files** option and specify a file size threshold for .tst and/or .pva files in the **Warn on file size large than (MB)** field to be notified when the size of the file exceeds the threshold. You can then reduce the size (and prevent performance problems) by dividing it into smaller files.

Proxy Settings

The Proxy panel controls how SOAtest and/or Virtualize works with proxy servers. It does not control the separate intermediary proxy used for web scenarios (for details on this other proxy, see Proxy Configuration Details).

- If Windows and IE (which use the same settings) are configured to properly use the proxy to access the relevant websites, select **Use system proxy configuration**.
  Otherwise, select **Enable proxy** and manually enter the correct settings. These settings should be equivalent to what you would use in the browser outside of SOAtest/Virtualize.
  - To use an automatic configuration script, select **Use automatic configuration script**, then enter the proxy address in the **Address** field.
  - If you want to use the same proxy server for all protocols, check the **Same proxy server for all protocols** check box, then enter the address and port of the proxy server you want to use in the **Proxy Address** and **Proxy Port** fields.
  - If you want to use different proxy servers for different protocols, clear the **Same proxy server for all protocols** check box, then enter the address and port of each proxy server you want to use in the **Proxy Address** and **Proxy Port** fields.

- If your proxy server requires authentication, check the **Enable proxy authentication** check box, then enter a valid user name and password in the **Username** and **Password** fields.
- If you want to allow Web traffic from designated IP addresses to pass through directly (avoiding the proxy), enter those IP addresses in the **Proxy Exceptions** text field. If you enter multiple addresses, use a semicolon (;) to separate the entries.
- The **Proxy Address** value should be a URL to the script: either an HTTP(S) URL or a file URL. File URLs should be formatted as "file:///" followed by the file system path where the proxy autoconfiguration script lives. For example, on Windows this could be `file:///c:/Users/user/scripts/proxy.pac`. On Linux, it might be `file:///home/machine/scripts/proxy.pac`.

HTTP proxies that do not require authentication can be used while managing remote SOAtest and Virtualize servers. HTTP proxies that require authentication will not be applied when adding a remote SOAtest or Virtualize server to the server tree.

Scanning Settings

This Scanning panel specifies settings related to how SOAtest scans Web applications. Available options include:

- **Agent name**: Determines the user agent that SOAtest uses to identify itself.
- **FTP Log**: Determines whether (and how) to create a log of FTP connections made to scan Web resources from SOAtest.
  Script options:
  - **Script extensions**: Determines what files SOAtest considers "scripts".
  - **Limit the number of script items per page to**: Determines the maximum number of script items per page that SOAtest will consider. If a page has more script items than the number that you allow, SOAtest will place a "red flag" icon next to the related Project tree node.
  - **Load JavaScript**: Determines whether or not SOAtest loads JavaScript.
  - **Simulate JavaScript events**: Determines how SOAtest simulates JavaScript events (such as opening and closing additional windows, running timers, and so on). If you choose **single time**, SOAtest triggers each handler once, with default arguments. If you choose **multiple times**, SOAtest tries to create multiple kinds of events while loading a site (in order to find new links).

Scripting Settings

The Scripting panel allows you to specify properties used for custom scripts.

- **Java**: For Java, you can specify the Java home directory and the path to the `javac` compiler. You need to specify these parameters if you want to compile Java methods in the editor.

  > **Note**
  >
  > The `javac` compiler is not included.

  - **Java home**: Specifies the Java installation directory.
  - **Javac classpath**: Specifies the Java classpath.
- **JavaScript**: If you create scripts using Jython or JavaScript, you can specify a script template in the **Script Template** field.
  - **Script Template**: Whatever code is specified in this field will be used as the default code for inlined scripting in the language with which the field is associated. This is prmarily useful for setting default inputs and common global variables. Script templates apply to scripts used in Extension tools; they do not apply to JavaScript that runs in a browser context.
- **Jython**: For Jython, you can specify the jython.home and jython.path variables. Both variables are used to locate Jython modules. Jython code that does not import any Jython modules can use the Jython scripting support without setting either variable. If you set the jython.home and jython.path variables, you need to restart SOAtest or Virtualize before the changes will take effect.
- **Jython Home**: Specifies the Jython installation directory. This must be a single directory. Use a forward slash (/) or backslash (\) to escape special characters.
- **Jython Path**: Used to add to your path modules that are not in your jython.home/Lib directory. Multiple paths can be listed. Use a forward slash (/) or backslash (\) to escape special characters.
- **Script Template**:Jy thon code that does not import any Jython modules can use the Jython scripting support without setting either the jython.home or jython.path.
- **Timeout Settings**: Specify how many minutes SOAtest or Virtualize should wait before attempting to stop an unresponsive script and log an error message.

Security Settings

The Security panel allows you to set the following security settings:

- **Global HTTP Authentication Properties**: Allows you to specify global HTTP authentication properties that can be used when configuring HTTP protocols within an applicable tool. Select the **Perform Authentication** check box and enter the **Username** and **Password** to authenticate the request, and select either **Basic**, **NTLM**, or **Digest**, from the **Authentication Type** drop-down list. For Kerberos, enter the **Service Principal** to authenticate the request. If the correct username and password, or the correct service principal, are not used, the request will not be authenticated.
- **Kerberos realm**: Specify the Kerberos realm associated with your network. By convention, this is typically your domain name in all caps (e.g. PARASOFT.COM).
- **KDC server**: Specify the hostname of your Key Distribution Center (e.g. kdc.parasoft.com).

- **Check Ticket**: This will execute a simple test to locate a cached Kerberos TGT (Ticket Granting Ticket) to grant access to the service. SOAtest and/or Virtualize will not be able to communicate with the service if it cannot first locate a valid TGT. For more information about Kerberos, see Co nfiguring Kerberos Authentication in SOAtest and/or Configuring Kerberos Authentication in Virtualize.
- **Trust all certificates**: Enable this option to accept any certificate. This is useful if you want to load pages whose certificates are not "trusted."
- **Use default Java cacerts**: Enable this option to accept only certificates from the standard list of Java-trusted certificate vendors.
- **Use client keystore**: Enables you to specify settings for both the server side and client side certificates for SSL through the Client Keystore options.

> **Important**
>
> In order to perform operations that use the XML Signature Verifier, XML Signer, or XML Encryption tools, or if using Key Stores, you will need to download and install the Unlimited Strength Java Cryptography Extension. For details, see JCE Prerequisite.

Keystores are specified at the test or responder suite level. If this option is selected, the following options are available in the **Certificate** and **Private Key** tabs:

Certificate tab options:
- **Use same key store for private key**: Select if the Key Store contains private keys for the certificate.
- **Key store file**: Specify the key store file by clicking the **Browse** button and using the file chooser that opens. If you want the path saved as a relative path (for example, to facilitate project sharing), check the **Persist as Relative Path** option.
- **Key store password**: Specify the Key Store password.
- **Key store type**: Specify the type of Key Store being used (e.g. JKS, PKCS12, BKS, PEM, UBER)
- **Load**: Click to populate the aliases with the available certificates/keys if the path, type, and key store password are valid.
- **Certificate alias**: Specify the certificate alias.

Private Key tab options:
- **Key store file**: (Only available if the **Key Store Contains Keys** option is unselected in the Certificate tab) Specify the key store file by clicking the **Browse** button and using the file chooser that opens. If you want the path saved as a relative path (for example, to facilitate project sharing), check the **Persist as Relative Path** option.
- **Key store password**: (Only available if the **Key Store Contains Keys** option is unselected in the Certificate tab) Specify the Key Store password.
- **Key store type**: (Only available if **the Use same key store for private key** option is unselected in the Certificate tab) Specify the type of Key Store being used (e.g. JKS, PKCS12, BKS, PEM, UBER)
- **Load**: Click to populate the aliases with the available certificates/keys if the path, type, and key store password are valid.
- **Private key alias**: Specify the private key alias.
- **Private key password**: Specify the private key password.

About Kerberos Authentication

Kerberos authentication is known as a trusted third-party authentication mechanism. A client requests access to a service not directly, but from another service: the Key Distribution Center, which manages network-wide authorization. This mechanism facilitates Single Sign-On (SSO) so that a client need only provide authorization credentials once in a given time period (usually 8-10 hours). The authorization is granted in the form of a ticket which can then be cached and reused throughout the granted time period without re-authenticating.

Entities in a Kerberos-protected network, such as clients and servers, are known as principals. The network-space that Kerberos protects is known as a realm. Microsoft's IIS (Internet Information Services) Server provides HTTP-based services with Kerberos through the Negotiation protocol. Other server vendors provide their own implementations of Microsoft's Negotiate protocol.

The ticket that is received upon initial authentication is known as a Ticket Granting Ticket, or TGT. For example, in a Windows environment, the TGT is generated when first logging on to the workstation in the morning. SOAtest and/or Virtualize authorizes itself to use a Kerberos-protected service by retrieving a user's TGT from the system cache.

> **Avoiding Common Kerberos Errors**
>
> For tips on common Kerberos errors and how to solve them, see http://docs.oracle.com/javase/8/docs/technotes/guides/security/jgss/tutorials/Troubleshooting.html.

Configuring Kerberos Authentication

1. Configure the following options in the Security preferences panel:
   - **Kerberos realm**: Specify the Kerberos realm associated with your network. By convention, this is typically your domain name in all caps (e.g. PARASOFT.COM).
   - **KDC server**: Specify the hostname of your Key Distribution Center (e.g. kdc.parasoft.com).
   - **Check Ticket**: Click to execute a simple test to locate a cached Kerberos TGT (Ticket Granting Ticket) to grant access to the service. SOAtest and/or Virtualize will not be able to communicate with the service if it cannot first locate a valid TGT.
2. Select the tool for which you intend to use Kerberos authentication.
3. Select the **Transport** tab and select **Security** from the left pane of the **Transport** tab.

Configure the following options from the security panel of the **Transport** tab:

- **Perform Authentication**: Select this option to activate authentication.

- **Use Global Preferences**: Select this option if you have authentication properties setup in Security Preferences.
- **Type**: Select **Kerberos** to perform Kerberos Authentication.
- **Service Principal**: Specify the name of the service/server as defined in the Kerberos database (e.g. HTTP/soatest.parasoft.com).

2. Now when you invoke your tool, the required Negotiate token will automatically be generated and send as an HTTP header.

> **Note**
>
> Kerberos provides a mechanism to prevent so-called "replay" attacks where a user tries to provide captured duplicate credentials for a service in order to gain access to them.When performing a load test, where multiple virtual users provide the same user credentials, the KDC will respond as if a replay attack is occurring and errors will be thrown. This is expected behavior and it is uncertain at this point whether there is a work-around.

JCE Prerequisite

In order to perform security operations that use XML Signature Verifier, XML Signer, or XML Encryption tools—or to use Key Stores— you will need to download and install the Unlimited Strength Java Cryptography Extension. You can do this as follows:

1. Go to http://www.oracle.com/technetwork/java/javase/downloads/index.html.
2. Download the JCE Unlimited Strength Jurisdiction Policy Files.
3. Install the downloaded files in into the following directory on your machine:
   `[Parasoft Test install dir]\[Parasoft Test version number]\plugins\com.parasoft.xtest.jdk.eclipse.core.[platform]_[jre version]\jdk\jre\lib\security`
   For example:
   `C:\Program Files\Parasoft\Test\9.10\plugins\com.parasoft.xtest.jdk.eclipse.core.win32.x86_64_1.8.0.102\jdk\jre\lib\security`
   Be sure to replace the existing `local_policy.jar` and `US_export_policy.jar` files with the new ones that you downloaded.
4. Restart SOAtest and/or Virtualize.

> **Where to Install JCE Policy Files**
>
> To see exactly where Unlimited JCE policy files should be added on your system, look at the message shown if you view keystore settings in the preferences for two-way SSL.
>
> This "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files may need to be installed" message is displayed only if Unlimited JCE policy files are not yet installed to the JRE that is running on. After the files are installed properly and product is restarted, the message will no longer be shown.

Server Settings

The Server panel allows you to configure the following settings for the SOAtest and/or Virtualize server. The SOAtest server allows you to work with the Call Back tool and Asynchronous testing—as well as Message Stub tools integrated into end-to-end test scenarios.

- **Start server**: Enable this option to automatically start the server when SOAtest and/or Virtualize starts up.
- **Server port**: Specifies the ports that the server use for HTTP/HTTPS.

SOA Registry Settings

SOAtest can create tests that enforce policies applied to Web service assets that are declared in a BEA AquaLogic Enterprise Repository or Software AG Centrasite repository as described in Using Oracle or BEA with SOAtest and Using Software AG CentraSite Active SOA with SOAtest. The SOA Registry panel allows you to specify settings that SOAtest will use by default in forms that reference such repositories. For instance, if you specify settings for BEA ALER here, SOAtest will use these values by default in the wizard for creating tests from BEA ALER.

SOAP Settings

The SOAP panel allows you to specify the following settings:

- **Default Transport**: Allows you to set the default transport protocol.
- **Attachment Encapsulation Format**: Allows you to choose **MIME**, **DIME**, or **MTOM**, for the default attachment encapsulation. See Working with Attachments for details.
- **SOAP Version**: Allows you to select **SOAP 1.1** or **SOAP 1.2**.
- **Outgoing Message Encoding**: Allows you to choose the encoding for outgoing messages. You can choose any **Character Encoding** you wish to read and write files, but the **Outgoing Message Encoding** provides additional flexibility so you can set a different charset encoding for the SOAP request.

Customizing SOAP Serialization Settings

You can also customize how SOAtest and/or Virtualize serialize the SOAP objects they transmit and deserialize the SOAP messages they receive, but you cannot do so within the Preferences panel.

SOAP messages are deserialized from XML into some native format and objects are serialized into XML format so that they can be sent as responses.

To add a `serializer/deserializer` pair, you add a line to the `register.py` file in the `<INSTALL_HOME>/plugins/com.parasoft.xtest.libs.web_<version>/root/startup directory`. You must programmatically use Jython register Apache Axis-compliant serializers.

For Axis, you can retrieve the TypeMappingRegistry used by calling soatest.api.SOAPUtil.getDefaultAxisRegistry(). After you retrieve that registry, you can use the Axis API to register the serializer as needed.

System Properties Settings

The System Properties panel lets you add JAR files, class folders, and Java projects to the classpath if needed. Use the available controls to add or remove JAR files, class folders, and Java projects. The specified JAR files, classpaths, and Java projects will be added to the system's classpath and the corresponding classes will be loaded into the JVM after SOAtest or Virtualize is restarted.

Click **Reload** to force classes from the class path entries to reload.

Enable the **Automatically reload** classes option if you want SOAtest/Virtualize to reload classes from your Eclipse project after being modified or recompiled.

Adding Jar Files in Bulk and in Headless Instances

If you want to quickly add a large number of jar files—or add jars to a headless instance of your Parasoft solution— copy them into one of the following directories within your workspace:

- `TestAssets/system_jars`
- `stubs/system_jars`
- `VirtualAssets/system_jars`

Jar files in those directories will automatically be loaded upon startup or after **Reload** is clicked in the Preferences page.

On a headless instance, if you want to reload the jars without having to restart SOAtest or Virtualize, call post /v<version>/preferences /systemProperties/reload from the REST API.

Traffic File Processing Settings

The Traffic File Processing panel lets you globally indicate that you want certain values (such as times-tamps) ignored whenever:

- you are creating parameterized .pvas from traffic, and
- request body correlations are configured automatically.

Virtualize is automatically configured to ignore timestamps—based on the regular expression [0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}([.][0-9]{1,3})?(([+-][0-9]{2}:[0-9]{2})|Z)?

You can add or modify exclusions using the table controls. Element names are specified as exact matches or using a wildcard (*) to match everything. Values are specified as regular expressions.

UDDI Settings

The UDDI panel lets you set the UDDI inquiry endpoint, which is the endpoint that you want SOAtest to reference when performing dynamic router resolution. If you specify a UDDI registry here, the SOAP Client tool can search for a service by querying that registry using the UDDI serviceKey specified in the SOAP Client parameters. If you do not specify a UDDI registry here, you have to configure your SOAP Client tool so that the server endpoint is hard-coded as a router value.

WSDL History

The WSDL panel lets you review or modify the WSDLs that have been used in tools and projects. These WSDLs will be available for selection in relevant drop-down boxes. This way, if you need to specify the same WSDL multiple times, you don't need to constantly type it in over and over again.

Enable the **Save WSDLs used in message responders, SOAP clients, and projects** if you want SOAtest/Virtualize to save tests' or assets' WSDL URIs. If you are using SOAtest only, the option will read **Save WSDLs used in SOAP clients and projects**. If you are using Virtualize only, the option will read **Save WSDLs used in message responders and projects.**

The **WSDL URI** field lists the WSDL URIs that will be available in tools' **WSDL URI** drop down menu. By default, all WSDL URIs used in related tools are added to this list. Click on a URI in the field and click **Refresh WSDL** to refresh the WSDL from the given location URL and re-parse it.

Enable the **WSDL/Schema Parsing** section to check all schema locations in order to locate components belonging to a give target namespace. Disable this option to use only the first schema location encountered in order to resolve components for a given target namespace.

XML Conversion Settings

The XML Conversion settings panel lets you register data models for fixed length messages.

For details on using this setting in Virtualize, see Fixed Length Message Responder. For details on using this setting in SOAtest, see Fixed Length Client and Fixed Length Call Back.

XML Schema History Settings

The XML Schema History panel lets you review or modify the XML Schemas that have been used in Messaging Clients (SOAtest), message responders (Virtualize), and projects. These Schemas will be available for selection in relevant drop-down boxes. This way, if you need to specify the same Schema multiple times, you don't need to constantly type it in over and over again.

XML Schema Locations Settings

The XML Schema Locations panel lets you view, add, and remove schema locations.The XML Validator tool needs to know where to find the schema that it should use to validate the document of concern. In most cases this is a URI and is supplied within the document being validated. If, however, the URI for the schema is not supplied or if you want to use a different location, then disable the **Use namespace as location URI for Schemas** option for the XML Validator tool. For more information on the XML Validator tool, see XML Validator. When the tool is run with this option disabled, SOAtest will use the schema location(s) indicated in this panel. To add a new schema location:

1. Click the **Add** button beneath the Namespace and Location columns.
2. In the dialog that opens, specify the Namespace and Schema Location.
3. Click **OK** after you have added all of the necessary locations.

To specify namespaces to skip:

1. Click the **Add** button beneath the List of namespaces to skip during XML Validation table.
2. In the dialog that opens, specify the namespace you want to skip.
3. Click **OK**.

To add OASIS XML Catalog Locations:

1. Click the **Add** button beneath the **OASIS XML Catalog Locations** section of the Schema Locations tab. The **Location** dialog box displays.
2. Type in the **OASIS XML Catalog Location** or Browse to it by clicking the **Browse** button.
3. Click **OK** after you have added all of the necessary locations.

Virtualize Localsettings

There are two ways to define localsettings files:

- Enter them manually in a simple text file simple text file. There are no name or location requirements. Each local setting should be entered in a single line.
- Export your GUI preferences, then adjust or extend them as needed. To export, choose **Parasoft> Preferences**, select **Parasoft** (the root element in the left tree), click the **Share** link, specify which settings you want to export.

Localsettings files can specify the following settings.

Licensing Settings

Setting	Purpose

`virtualize.license. use_network=true|false`	Determines whether the current installation retrieves its license from LicenseServer.
`virtualize.license. network.host= [host]`	Specifies the machine name or IP address of the machine running LicenseServer Configuration Manager. Example: `virtualize.license.network.host=10.9.1.63`
`virtualize.license. network.port= [port]`	Specifies the LicenseServer port number. Example: `virtualize.license.network.port=2222`

`virtualize.license.network. edition=[edition_name]`	Specifies the type of license that you want this installation to retrieve from LicenseServer.
	
	`[edition_name]` can be `server_edition`. To use a custom edition, do not set anything after the "="; simply leaving the value empty.
	
	Example:
	
	`virtualize.license.network.edition=desktop_edition`
	
	`virtualize.license.network.edition=server_edition`

`virtualize.license.autoconf. timeout=[seconds]`	Specifies the maximum number of seconds this installation will wait for the license to be automatically configured from LicenseServer. Default is 10.
`virtualize.license. local.expiration=[expiration]`	Specifies the local license that you want this installation to use.
`virtualize.license. local. password=[password]`	Specifies the local password that you want this installation to use.
`virtualize.wait.for.tokens.time= [time in minutes]`	Specifies the time that this installation will wait for a license if a license is not currently available. For example, use `virtualize.wait.for.tokens.time=3` to configure Virtualize to wait three minutes for a license token to become available.

See Manually Adding the License to localsetttings for additional notes and examples.

Virtualize Settings

Setting	Purpose
`server.startup`	Determines whether the server is automatically started upon Virtualize startup.
`server.port.http`	Specifies the port that the Virtualize Server uses for HTTP.
`server.port.https`	Specifies the port that the Virtualize Server uses for HTTPS.
`server.port. monitoring`	Specifies the port that the Virtualize Server uses for monitoring.
`system.properties. class- path=[path1];[path2]; [path3] ...`	Specifies which jar files and class folders are in the classpath. For example: `system.properties.classpath=C\:\\myjars\\myLib1.jar;C\:\\myjar s\\myLib2.jar`
`scripting.timeout. minutes`	Specifies the number of minutes after which Virtualize will attempt to stop an unresponsive script and log an error message.
`scripting.jython.home`	Specifies the Jython installation directory. This must be a single directory.
`scripting.jython.path`	Used to add to your path modules that are not in your jython.home/Lib directory. Multiple paths can be listed.
`datasources.jdbc. classpath=[path1]; [path2];[path3] ...`	Specifies the location of JDBC driver jar files and class folders. Special characters (spaces, slashes, colons, etc.) must be escaped; for instance: %20 = SPACE %3A = : %5C = \

	%7B = {
	
	%7D = }
	
	%24 = $
	
	If listing multiple jars, use \n as a delimiter.
	
	For example,
	`C:\temp\with space\${example}\jar.jar`
	
	becomes
	
	`C%3A%5Ctemp%5Cwith%20space%5C$2 4%7Bexample%7D%5Cjar.jar\n`

`traffic.wizard.xpath.ignores`	Lets you globally indicate that you want certain values (such as times-tamps) ignored whenever:
	
	• you are creating parameterized .pvas from traffic, and
	• requiest body correlations are configured automatically.
	
	Exclusions are specified in the format
	
	`traffic.wizard.xpath.ignores=[element name 1]:[value pattern 1];[element name 2]:[value pattern 2];[element name 3]:[value pattern 3]`
	
	For example:
	
	`traffic.wizard.xpath.ignores=*:[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0- 9]{2}:[0-9]{2}([.][0-9]{1,3})?(([+-][0-9]{2}:[0- 9]{2})|Z)?;uuid:[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}`
	
	Note that when the backslash character (\) is used in the regular expression, it needs to be escaped. For example, the regex [\d], which represents a single digit, would be entered as [\\d].

Continuous Testing Platform Settings

Setting	Purpose
`env.manager.server`	Specifies the location of the CTP server. *Required* Example: env.manager.server=http://em_hostname:8080/
`env.manager.server.name`	Specifies the name that will be used to label this server on CTP.You can use whatever value will help you identify this server. *Optional* Example: env.manager.server.name=MyVirtServerLabel
`env.manager.notify`	Determines whether the Virtualize server notifies Parasoft CTP when virtual assets are deployed. *Optional* Example: env.manager.notify=true
`env.manager.username`	Specifies the username for logging into CTP. *Optional* Example: env.manager.username=me
`env.manager.password`	Specifies the password for logging into CTP. *Optional* Example: env.manager.password=12345

Miscellaneous Settings

Setting	Purpose
`dtp.autoconfig=true|false`	Enables autoconfiguration with Parasoft Test settings stored on the DTP server. Default: `false`
`dtp.enabled=true|false`	Determines whether the current Parasoft Test product is connected to DTP. Default: `false`
`dtp.server=[host]`	Specifies the host name of the DTP server.
`dtp.port=[port]`	Specifies the DTP server port.

`console.verbosity.level=low|normal|high`	Specifies the verbosity level for the Console view. Available settings are: `low`: Configures the Console view to show errors and basic information about the current step's name and status (done, failed, up-to-date). `normal`: Also shows command lines and issues reported during test and analysis. `high`: Also shows warnings.
`parallel.mode=Manual|Auto|Disabled`	Determines which of the following modes is active: • **Auto**: Allows the product to control parallel processing settings. • **Manual**: Allows you to manually configure parallel processing settings to suit your specific needs. • **Disabled**: Configures the product to use only one of the available CPUs.
`parallel.max_threads=<number>`	Specifies the maximum number of parallel threads that can be executed simultaneously. The actual number of parallel threads is determined based on the number of CPUs, available memory, and license settings.
`parallel.free_memory_limit=<percentage>`	Specifies the amount of memory that should be kept free in low memory conditions (expressed as a percentage of the total memory available for the application). This is used to ensure that free memory is available for other processes.
`parallel.no_memory_limit=true|false`	Indicates that you do not want to place any restrictions (beyond existing system limitations) on the memory available to the product.
`tasks.clear=true|false`	Clears existing tasks upon startup in cli mode. This prevents excessive time being spent "loading existing results." The default is true.
`security.trust.all.certificatestrue|false`	Tells Virtualize that you want it to accept any certificate. This is useful if you want to load pages whose certificates are not "trusted."
`security.use.default.java.cacertstrue|false`	Tells Virtualize that you want it to accept only certificates from the standard list of Java trusted certificate vendors.

Manually Adding the License to localsetttings

To add or change license settings vialocalsettings:

1. If you will be using a custom edition license, define the appropriate license features in the localsettings as follows:
   `[product].license.custom_edition_features= All enabled features`
   Note that license feature settings apply only to custom edition licenses.
2. Define the main license settings in the localsettings as follows:

```
virtualize.license.network.edition= Type of license edition
virtualize.license.use_network= Value (true or false)
license.network.host= Host name
license.network.port= Port number
```

Here are several examples of properly-configured license settings in localsettings file:

Virtualize network license - desktop edition

```
virtualize.license.network.edition=desktop_edition
virtualize.license.use_network=true
license.network.host=main1.parasoft.com.pl
license.network.port=2222
```

Virtualize network license -custom edition with various features

```
virtualize.license.custom_edition_features=Virtualize, Server, Message Packs, Unlimited Million Hits / Day
virtualize.license.network.edition=custom_edition
virtualize.license.use_network=true
license.network.host=main1.parasoft.com.pl
license.network.port=2222
```

Virtualize network license -custom edition with various features

```
virtualize.license.local.expiration=2014-04-15
virtualize.license.local.password=PARASOFT_353E2A7DA4F3D4B2FF142B0A262AF62B9DEC3449
C124773BAF0B4B508FF21139E867D9772F3702716FCE6D8EA16ACE668DE0EE629D154713599203BD85CE1213_7937E7ED374E70FDD62EE84
11C2BB2D8EB465019E64BF3EF3A87DE6B67FB10BBCAF8611B08F70 D9420AC574FC5B3E5EB7241B20506DE2C60B0D80462CBEDBD
virtualize.license.use_network=false
```

Note that with local licenses, the enabled features are specified via the generated password.