Server Log Settings

The log functionality of the Servoy Application Server is based on Log4j of the Apache Logging Services project.

The logging mechanism is highly configurable and enables control over:

- the level of logging per area
- the format of the log file
- log file rollover/location/size settings

All log settings are stored in:

- **before Servoy 8.4: servoy.properties** located in the <servoy>/application_server directory. The specific log properties are properties prefixed with 'log4j'.
- **Servoy 8.4 and later**: all log settings are in a separate file: **log4j.xml** located in the <servoy>/application_server

In case of WAR deployment, the export as war wizard will allow you to specify the log4j.xml/servoy.properties that you want the war deployment to use.

Besides the log4j properties, administrators can also configure the **servoy.client.report.max.valuelist.items** property on the Servoy admin page, or directly in the servoy.properties file. This property is used to disable/enable the logging of messages that not all records can be loaded into the valuelist, i.e. when the valuelist contains more than 500 records. It is not recommended to set this property to false when running from the Servoy developer, otherwise the developers will not be aware of these warnings at all.

Changing the Log Level

There are 6 levels of logging (in order of severity):

1. FATAL: The FATAL level designates very severe error events that will presumably lead the application to abort.
2. ERROR: The ERROR level designates error events that might still allow the application to continue running.
3. WARN: The WARN level designates potentially harmful situations.
4. INFO: The INFO level designates informational messages that highlight the progress of the application at coarse-grained level.
5. DEBUG: The DEBUG Level designates fine-grained informational events that are most useful to debug an application.
6. TRACE: The TRACE Level designates finer-grained informational events than the DEBUG.

By default, the log level is set to WARN. This means that all WARN, ERROR and FATAL messages will be logged. Here are some of the entries we have there:

Some log level settings from servoy.properties (< 8.4)

```
log4j.logger.com.servoy.j2db.util.Debug=WARN
log4j.logger.org.apache.wicket=WARN
log4j.logger.com.org.sablo.websocket=WARN
log4j.rootCategory=WARN, file, configservlet
```

The same log level settings as configured in log4j.xml (>= 8.4))

```
    <Loggers>
        <Logger name="com.servoy.j2db.util.Debug" level="WARN" />
        <Logger name="org.apache.wicket" level="WARN" />
        <Logger name="org.sablo.websocket" level="WARN" />
        <Root level="WARN">
            <AppenderRef ref="asyncfile" />
            <AppenderRef ref="configservlet" />
            <AppenderRef ref="debugconsole" />
            <!-- <AppenderRef ref="stdout" /> -->
        </Root>
    </Loggers>
```

The first logger entry sets the log level for the internal Servoy Java class used for most of the application server logging done by Servoy.
The second entry sets the log level for the Java class used for all logging of the Wicket framework, used for the Servoy Web Client.

The third entry sets the log level for websocket logger(s) which are used in Servoy NGClient.
The forth entry sets the overall log level and specifies the output channels for the log.

Check the actual configuration file contents to see more logger entries and other configuration options that are used by default (and can be altered to suit your needs).

To change the the log level for various loggers, replace 'WARN' in the corresponding entry with, for example 'TRACE' - to get the most logging information. The overall log level could also be set, but this would generate a lot of log data. You can also target a subset of loggers by using the common prefix in the logger name for example, but you can look at log4j documentation for more details.

Here is an example of a log entry; the pattern it uses is defined via PatternLayout in the configuration:

Log entry

```
2019-03-25 10:48:52,965 ERROR [Executor,uuid:3f0bccdb-0dec-42fd-8c06-81b67fad3bce] com.servoy.j2db.util.Debug -
error getting data from global method valuelist
```

It starts with date, log level (Error in this case), thread that logged this error, logger name then message. In this case, this message is logged under: com.servoy.j2db.util.Debug logger. So, in order to hide/show/customize this message you have to use that logger name in log4j.xml (or servoy.properties before v 8.4)

Some of the loggers that Servoy registers:

- com.servoy.j2db.util.Debug
- persistence.Server
- ClientManager
- com.servoy.j2db.persistence.XMLExporter
- com.servoy.j2db.persistence.XMLImportHandlerVersions1to10
- com.servoy.j2db.persistence.XMLInMemoryImportHandlerVersions11AndHigher
- com.servoy.j2db.dataprocessing.editedRecords
- WebServer
- datasource.TransactionConnection
- com.servoy.performance.timing.methods
- com.servoy.performance.timing.sql
- ResourceProvider
- loggers related to jsunit tests:
    - com.servoy.automation.jsunit.runner.ImportClient
    - com.servoy.automation.jsunit.SolutionJSTestSuite
    - com.servoy.automation.jsunit.mobile.ServoyMobileJSUnitTestRunner
    - com.servoy.eclipse.jsunit.runner.JSUnitToJavaRunner
    - com.servoy.mobile.test.server.service.TestSuiteController
- sablo (part of NGClient impl) related loggers :
    - org.sablo.BrowserConsole
    - org.sablo.eventthread.Event
    - org.sablo.eventthread.EventDispatcher
    - org.sablo.services.server.FormServiceHandler
    - sablo property type related loggers (start with org.sablo.specification.property):
        - org.sablo.specification.property.types.TypesRegistry
        - org.sablo.specification.property.types.DoublePropertyType
        - org.sablo.specification.property.types.EnabledSabloValue
        - org.sablo.specification.property.types.FloatPropertyType
        - org.sablo.specification.property.types.IntPropertyType
        - org.sablo.specification.property.types.LongPropertyType
        - org.sablo.specification.property.CustomJSONPropertyType
        - org.sablo.specification.property.CustomPropertyTypeResolver
        - org.sablo.specification.property.BrowserConverterContext
    - loggers related to sablo packages and specification (start with org.sablo.specification):
        - org.sablo.specification.Package
        - org.sablo.specification.WebObjectSpecification
        - org.sablo.specification.WebLayoutSpecification
        - org.sablo.specification.WebComponentSpecProvider
        - org.sablo.specification.WebServiceSpecProvider
        - org.sablo.specification.WebSpecReader
    - loggers related to classes close to web socket communication (start with org.sablo.websocket):
        - org.sablo.websocket.impl.ClientService
        - org.sablo.websocket.utils.JSONUtils
        - org.sablo.websocket.utils.PropertyUtils
        - org.sablo.websocket.BaseWebsocketSession
        - org.sablo.websocket.BaseWindow
        - org.sablo.websocket.WebsocketEndpoint
        - org.sablo.websocket.WebsocketSessionManager
    - org.sablo.BaseWebObject
    - org.sablo.CustomObjectContext
    - org.sablo.IndexPageEnhancer
- ngclient related loggers:
    - com.servoy.less.Compiler
    - ngclient property type related loggers:
        - com.servoy.j2db.server.ngclient.property.types.FormatPropertyType
        - com.servoy.j2db.server.ngclient.property.ComponentTypeSabloValue
    - com.servoy.j2db.server.ngclient.startup.resourceprovider.ResourceProvider
    - com.servoy.j2db.server.ngclient.MessageLogger
    - com.servoy.j2db.server.ngclient.api
- com.servoy.extensions.plugins.rest_ws.RestWSPlugin
- com.servoy.j2db.server.main.Activator
- com.servoy.j2db.server.headlessclient.ServoyModificationWatcher
- com.servoy.j2db.util.ImageLoader
- com.servoy.PersistIndexCache
- plugin.oauth

The logging level can be set independently of for subsets of all these loggers, by adding extra 'log4j.logger' entries in the servoy.properties / log4j.xml file (depending on what version of Servoy you are using); for example:

Set the loglevel to the most finegrained level for the ClientManager class (in servoy.properties, so for Servoy <8.4))

```
log4j.logger.ClientManager=TRACE
log4j.logger.org.sablo.websocket=TRACE
```

Set the loglevel to the most finegrained level for the ClientManager class (in log4j.xml, so for Servoy >=8.4))

```
<Logger name="ClientManager" level="TRACE" />
<Logger name="org.sablo.websocket" level="TRACE" />
```

Setting the log level to DEBUG or TRACE will generate a lot of logging data. This is not recommended in a production environment, unless absolutely necessary. Make sure enough disk space is available for the log file(s).

Logging Pattern

The pattern in the *log4j.xml* defines what information is logged. Log4j supports a number of lookups, Servoy has added servoy-lookup (starting from release 2021.06) that can be used to print Servoy-specific data in the logfile.

The lookup is called *servoy* and can be used like in this example. Note that the double dollar-sign is needed to ensure that the value is looked up for each message, otherwise Log4j may cache the value.

```
                      <PatternLayout
                              pattern="%d %p [%t] %c - %m [$${servoy:clientid:-(NO CLIENT)} $${servoy:solution:
-(NO SOLUTION)}]%n" />
```

The supported lookup keys are:

- clientid
- solution
- username
- useruuid
- clienttype
- hostname
- hostaddress
- sessionkey (ng client only)

More information on Log4J

For more information on log4j, visit the Log4j site (Frequently Asked Log4j Questions section can help with some questions).