Troubleshooting

In This Chapter

Servoy Application Server troubleshooting

When experiencing problems with the Servoy Application Server, there are a number of known reasons, which are discussed below.

When the discussed problems below do not offer a solution, the nest steps would be:

- Check out the application server log, located in {servoyInstall}/application_server/servoy_log.txt and also displayed on the Servoy Admin page under Server Log
- When running the Servoy Application Server using the Service component, check the {servoyInstall}/application_server/.service_log.txt
- When using the bundled PostgreSQL database engine and experiencing database related issues, check the PostgreSQL log "postgres_log.txt" (by default located in {servoyInstall}/application_server/postgres_db/)

Servoy Admin page > Server Log
When analyzing the Servoy Application Server log on the Servoy Admin page, it's its possible to change the level of displayed log messages using the dropdown at the top of the page, with "DEBUG" giving the most details and "FATAL" the least.

Tracing
The Server Log also provides an option to temporarily enable Tracing, to provide even more detailed information in the log. Do note that tracing should only be enabled when investigating issues, as it generates a large quantity of output.

Server doesn't start

When the Servoy Application Server will not start, there are a few likely reasons for it:

- The Servoy Application Server cannot access the Servoy Repository database. As the Servoy Application Server required the Servoy Repository database to be available on launch, failure to connect to the database will prevent the Servoy Application Server from starting. The most likely reasons for the Repository Database not being accessible are:
  - The Database Server that hosts the Servoy Repository database is not running. Note that the Servoy Application Server does NOT start any database server!
  - The Database Server runs on another Server which cannot be reached by Servoy due to network configuration
  - The Database Server doesn't accept the JDBC connection the Servoy Application Server is trying to make. See Database Connectivity fo r database specific trouble shooting.
  - Invalid configuration of the database server connection named "repository_server" in Servoy
  - No database server setup within Servoy with the name "repository_server"
- The Servoy Application Server fails to start after updating the SSL configuration or disabling SSL. Any property in the servoy.properties file that contains the word "password" is automatically encrypted using the SSL settings configured (if SSL is enabled). If the configuration is changed, the Servoy Application Server is unable to decrypt the passwords. The solution is to set the passwords back to plain text in the servoy.properties file and starting the server again.
  When this scenario happens, the following error appears in the console, when starting the Servoy Application Server in a console/command window: **javax.crypto.BadPaddingException: Given final block not properly padded**
- The Servoy Application Server fails to start due to network errors that result from Java starting in IPv6 mode, while the network hardware actually runs on IPv4. The Servoy Application Server can be forced to start using IPv4 by adding '-Djava.net.preferIPv4Stack=true' to the startup command

Server crashes

The Servoy Application Server is very robust and is known for its stability and extreme long uptime. The Servoy Application Server is not likely to actually crash, in a sense that the entire process is killed. What can happen is that due to incorrect configuration the performance decreases to the point where it seems like the Servoy Application Server has become unresponsive.

The most likely reason for this to happen are:

- Database Connection pool exhaustion: The Servoy Application Server manages a pool of connections for each Database Server it is connected to. When the pool is dimensioned too small, requests from Servoy Clients must wait on echother to finish, reducing the performance. Errors like "**ja va.util.NoSuchElementException: Timeout waiting for idle object**" in the log indicate th4e Connection Pool being exhausted. For more information on connection pool configuration, see Connection pooling in the Database Connections chapter

- OutOfMemory exceptions: If the Servoy Application Server requires more memory than is configured maximum, OutOfMemory exceptions will be raised. When these type of exceptions occur

Servoy Smart Client troubleshooting

Smart Client related issues usually are in one of the following categories:

- Startup related issues
- Connectivity related issues
- Performance related issues

When investigating issues related to Smart Clients, the following tools are available:

- Java WebStart console
- Java WebStart logging and tracing
- Checking the contents of the JNLP file that is used to start the Smart Client, by appending a "raw" parameter with the value "true" at the end of the URL of the JNLP file. With this extra parameter appended, the contents of the JNLP file will become the source of the web page, which can then be viewed in the browser:
  {serverURL}/servoy-client/{solutionName}.jnlp?raw=true
- Server log: It can be that the Smart Client has issues due to something happening on the Servoy Application Server, thus it's important to also check the Application Server log when debugging Smart Client issues

Smart Client won't start

- The JNLP file gets downloaded as file: When this happens, it means can be one of two reasons:
  - The browser and/or the OS has not associated with Java WebStart: Associate the jnlp file extensions with Java WebStart
  - Java/Java WebStart is not installed at all: Sun/Oracle Java Runtime environments ship with Java WebStart by default. Install a Sun /Oracle JRE to solve this issue
- Error: **cannot find dataservice, it may not be running on server**
  This error is displayed in the Servoy Smart Client when it can't properly connect to the Servoy Application Server at startup. Most likely cause of this message is incorrect configuration of the Network Settings in combination with firewalls and proxies. See Network related settings for more information on configuring the network settings.
- Error: **Connection to Servoy server is invalid**
  Invalid SSL Certificate or using a self-signed certificate, which is not supported for Smart Clients. See Network related settings for more information on configuring SSL
- Error: **sun.security.validator.ValidatorException: PKIX path validation failed: java.security.cert.CertPathValidatorException: Path does not chain with any of the trust anchors**
  The Certificate used for SSL is issued by a Certificate Authroity (CA) that is not recognized by Java. For example certificates form startcom.org. The solution is to use a Certificate issues by a different CA that is trusted by Java
- Error: **Java Web Start error: Problem downloading the application**
  Something goes wrong in Java WebStart with downloading and starting the Smart Client. Check the Details tab for pointers

Smart Client looses connections

Smart Clients operate connected to the Application Server. When the connection is lost, the Smart Client will automatically try to reconnect. When reconnection is not possible, the Smart Client will automatically provide the user with an error message and the option to close the Smart Client or to continue to try and reconnect.

If disconnections occur without apparent reason like no internet/wifi/network connection, the following items could be the cause of the problem:

- One of the network components is closing the connection automatically. Most likely the network component is configured to automatically close inactive or long running requests. If altering the network setup is not an option, it's possible to lower the ping frequency between the Smart Client and the Server to make sure the connection remains active. Depending on the configuration of the Smart Client related Network setting, different settings might require updating. See Network related settings for more information.

Smart Client performance issues

When the solution that is run in the Smart Client doesn't provide the expected performance, it can be that it's the solution that requires tuning. Besides required tuning of the solution, the following items could negatively impact the Smart Client performance

- Too little memory available: The Smart Client is a Java process and as such it needs to be explicitly told the maximum amount of memory it is allowed to use. For more information on memory management for Smart Clients see Smart Client memory management in the Memory Management chapter
- Connection issues: if the network connection is not stable, this could have a negative impact on the Smart Client performance. If for example a firewall or NAT router automatically starts closing connections because it can't handle many open connections, there will be a performance penalty due to the fact Servoy tries to use a connection that is already closed and then need to establish a new connection. By lowering the RMI Connection timeout the use of already closed connections can be prevented. See Network related settings for more information RMI connection timeout
- Database connection pool exhaustion: When the solution that is run in the Smart Client required database access, it sends the request to the Servoy Application Server that in turn communicates with the database. It does so using a pool of connections. If all clients do more requests at the same time than the maximum allowed number of connections in the pool, the requests will take longer to execute, as they will be on hold untill a used connection is released back to the pool. For more information on Connection pooling, see Connection pooling in the Database Connections chapter

Smart Client crashes

Servoy Smart Clients are not prone to crash. In the unlikely event that this does happen, it's best to enable Java WebStart logging and tracing, start a Smart Client and use it until it crashes and then analyse the generated log

Smart Client automatic update issues

The Servoy Smart Client will by default automatically update itself if a new version of the Smart Client itself or a Solution that is run in the Smart Client is available on the Servoy Application Server.

When issues are experienced with the auto update behavior of the Servoy Smart Client, it can be due to the following reasons:

- The setting "servoy.fastClientStartup" allows starting the Smart client from the cache, without checking with the Servoy Application Server first to see if there are updates. When set to "true" it provides a faster launch, as it starts the Smart Client from the Java Web Start cache. The downside it that it can happen after the Servoy Server Server or Solution updated that the Smart Client launch will fail. If the Smart Client gets started again after the fail. When this happens the cache will be cleared and on the next launch the latest version will be automatically downloaded from the Servoy Application Server.

If the above described situations are not applicable, there are three options to force the update:

- Flush the solution on the Servoy Admin page: Each Solution listed on the Solutions page of the Servoy Admin page has a flush button. When clicked, the internal version id for the Solution is updated with a new value, indicating to connecting Smart Clients that a new version is available and should be downloaded
- Remove the Smart Client application from the Java WebStart cache: Cached applications can be removed from the cache through the Java Control Panel. See Client side settings through the Java Control Panel in the Smart Client chapter for more information
- Remove the temporary internet files from the Java WebStart cache: The WebStart JNLP file is also cached by WebStart in the temporary Internet files. If the above mentioned solutions fail, deleting the JNLP could be the last resort. See the Temporary Internet Files in the Java Control Panel.

Other Smart Client issues

- Microsoft JScript compilation error: syntax error (Windows only):
  When during launching the Smart Client a *Syntax error* dialog pops up with the title *Microsoft JScript compilation error*, this means that on the client machine Java WebStart is configured to look at the browser for the Proxy settings. On Windows this means that it is looking at the *Local Area Network (LAN) Settings* under *Connections* of *Internet Properties*. Where these setting are located differs per Windows version, but they can usually be accessed through the (advanced) network settings of a browser. The solution is to either configure Java WebStart to use "Direct connection" (Not to be confused with Servoy's Direct Connection mode) or to uncheck all options in the *LAN Settings* of Windows. As both options are system-wide settings they can influence other applications.
- Plugin or bean not loading in the Client:
  If one specific plugin or bean doesn't seem to load in the Client, check the server log around the time when the Servoy Application Server got started. At server startup, all the plugins and beans are loaded. If there is an issue with the plugin or bean, it will be excluded from the list of plugins and beans send to the client. Any issue with the plugin or bean would result in errors in the server log when the server gets started. You can review the list of plugins that are send to the client by adding "?raw=true" to the url of the Smart Client (http://domain:post/servoy-client/solutionName.jnlp) and then viewing the content of the JNLP file by looking at the source code of the page in the browser.

Proxy Settings in Java Webstart, causing fallback to Direct Connection, for which requirements aren't met

Database Connectivity

General

- Is the required JDBC driver available? Servoy comes bundled with a limited set of JDBC drivers, due to licensing. For example, it is not allowed to bundle JDBC drivers of Oracle or the official Microsoft SQL server JDBC driver. See the database section of the Servoy stack info for a list of the shipped JDBC drivers. If no JDBC driver is listed for the database to which a connection needs to be made, it needs to be added manually. See Database Connectivity > JDBC Drivers for more details on which JDBC driver to use and how to add it.
- Is the database up and running? While obvious that the database needs to be up and running in order for it to be used, it does happen quite often that the source of connectivity issues boils down to the database not being started. Note that the Servoy Application Server doesn't start any database!
- The Tables & views do not show up in Servoy Developer/errors on the server about missing tables/views while the connection itself seems to work: check that the schema and catalog properties of the database server settings are set correctly. These two properties are send to the database and based on those the database/JDBC driver filters the visible Tables/views for the connection

Oracle

When connecting to Oracle databases, make sure to use the correct version of the Oracle JDBC driver. Each Oracle database version has it's own version of a JDBC driver and when connecting to Oracle with a JDBC driver that was meant for another version of Oracle, the results can be mixed

- Servoy doesn't see the tables in the database: Most likely cause is the 'Schema' property for the connection. This property must be set to the appropriate 'Schema' in the Oracle database. The value for the 'Schema' property must be all uppercase when connecting to an Oracle Database.

SQL Server

- Failed to establish a connection: In order to connect to SQL Server using JDBC, the TCP/IP option of SQL server needs to be enabled, which it isn't by default. If this is the case, a error similar to the following can be found in the log: "com.microsoft.sqlserver.jdbc.SQLServerException: The TCP/IP connection to the host has failed."
- System tables and Views show up in Servoy Developer, despite the "Skip System Tables" in the Database Server definition: This is a known issue with the SQL Server driver

MySQL

- Connection to database lost: MySQL is notorious for dropping idle connections after a certain timeout. This can be prevented by using the following "validation query" for the Database Server connection: "select 1", at the cost of a little overhead. It's preferred to configure MySQL to use longer timeouts.