# OpenInsight OEngineServer Configuration

**Version 1.1**

REVELATION
SOFTWARE
**A Division of Revelation Technologies, Inc.**

Part No. 414-966

Printed in the United States of America.

# Table of Contents

# Section I: Requirements

The OpenInsight OEngineServer requires the Java Runtime Environment installed on the system where the OEngineServer is installed. The Java Runtime Environment can be downloaded from Oracle at:

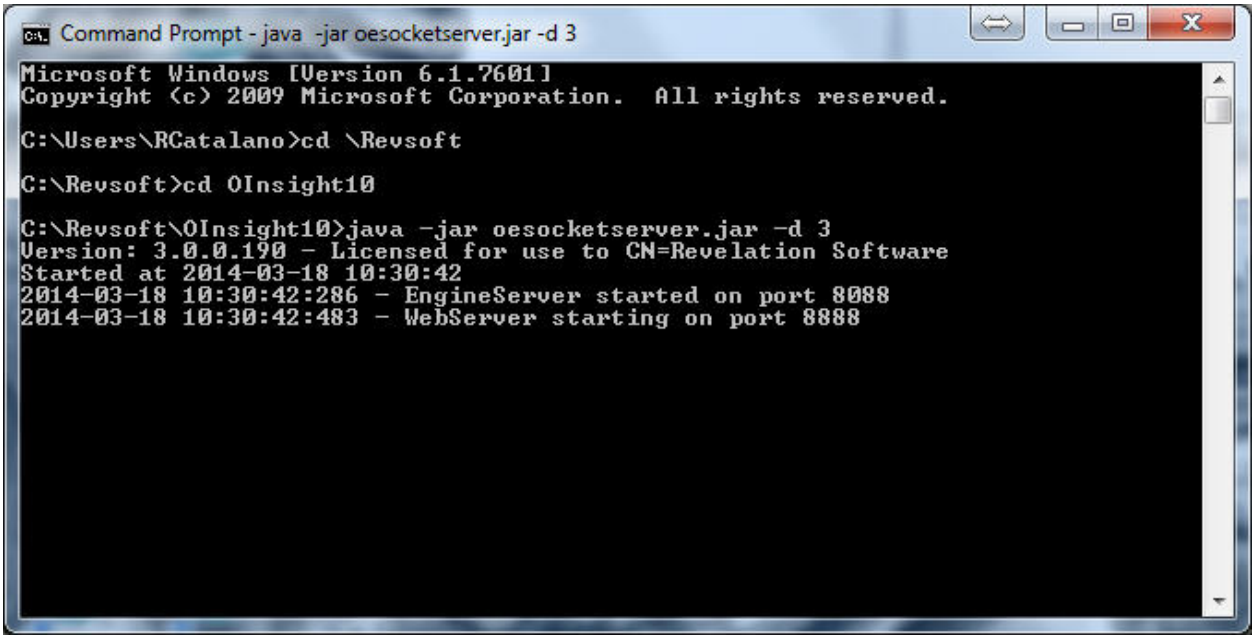
https://www.java.com/en/download/

For Windows 7, Windows 8, Windows Server 2008 and Windows Server 2012, 64-bit systems, please make sure that you install the 32-bit version of the Java Runtime Environment.

# Section II: Running the OEngineServer in debug mode

You can run your OEngineServer in debug mode by typing the following command from a DOS command prompt in your OpenInsight directory:

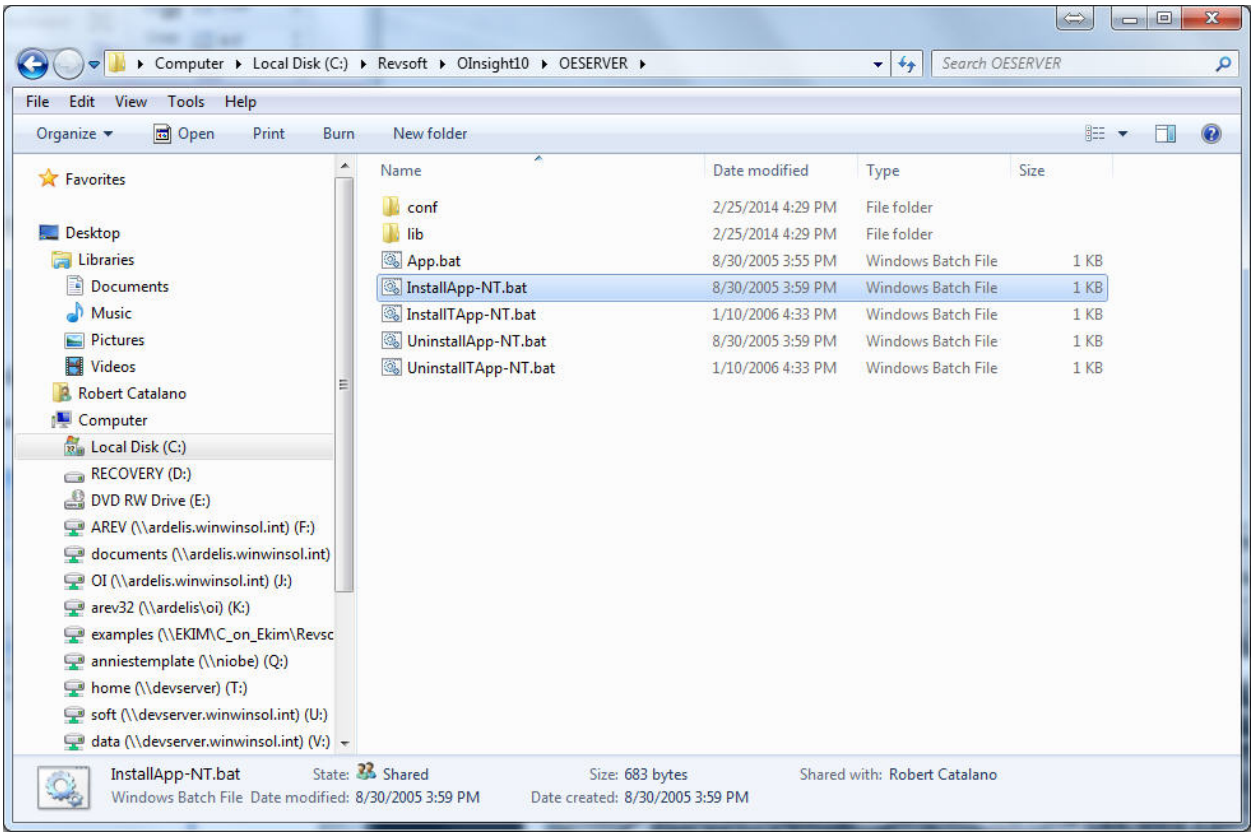
**java –jar oesocketserver.jar –d 3**




The OEngineServer can be stopped by hitting Ctrl “C”.

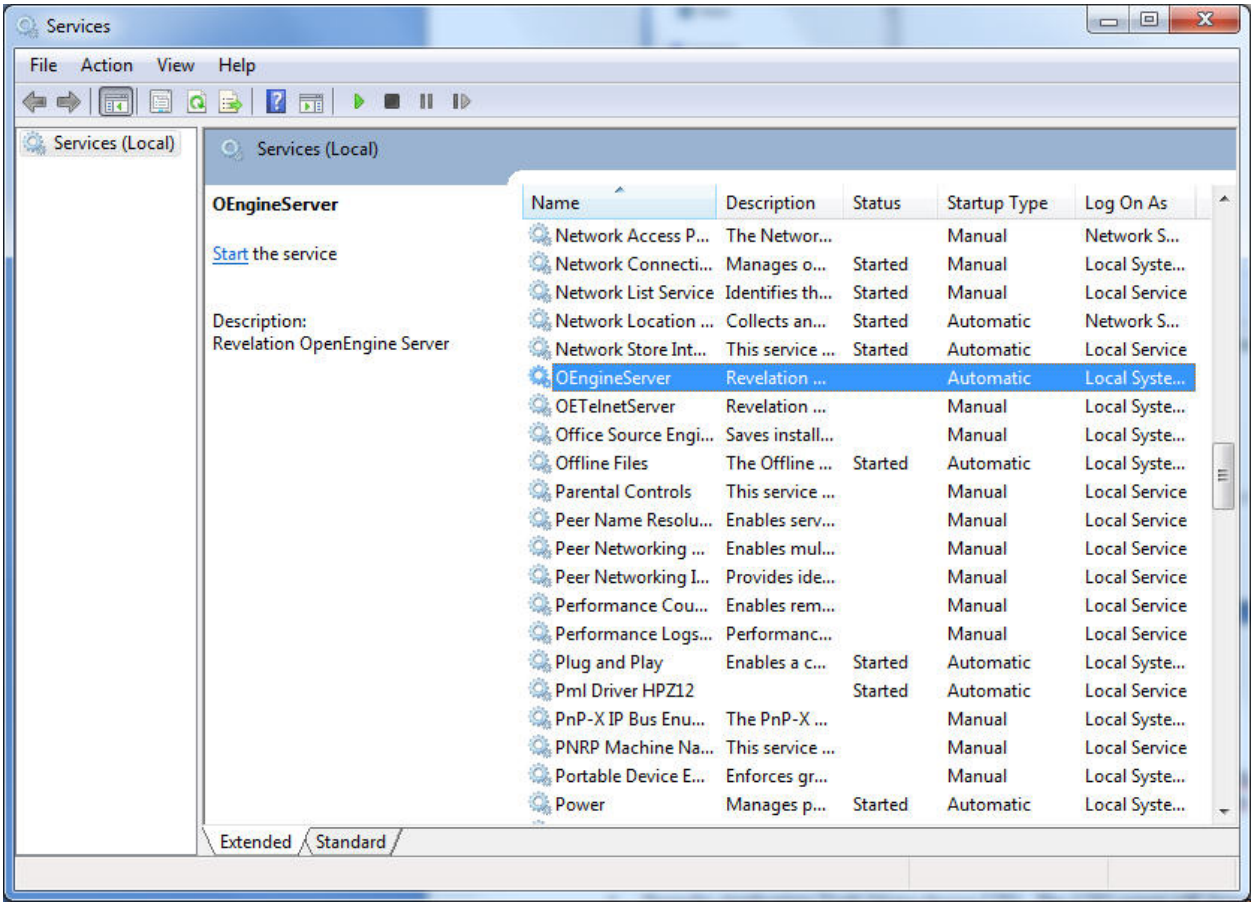
# Section III: Installing the OEngineServer as a service

To install the OEngineServer as a service double-click on the InstallApp-NT.bat file found in the OESERVER folder within your OpenInsight directory.

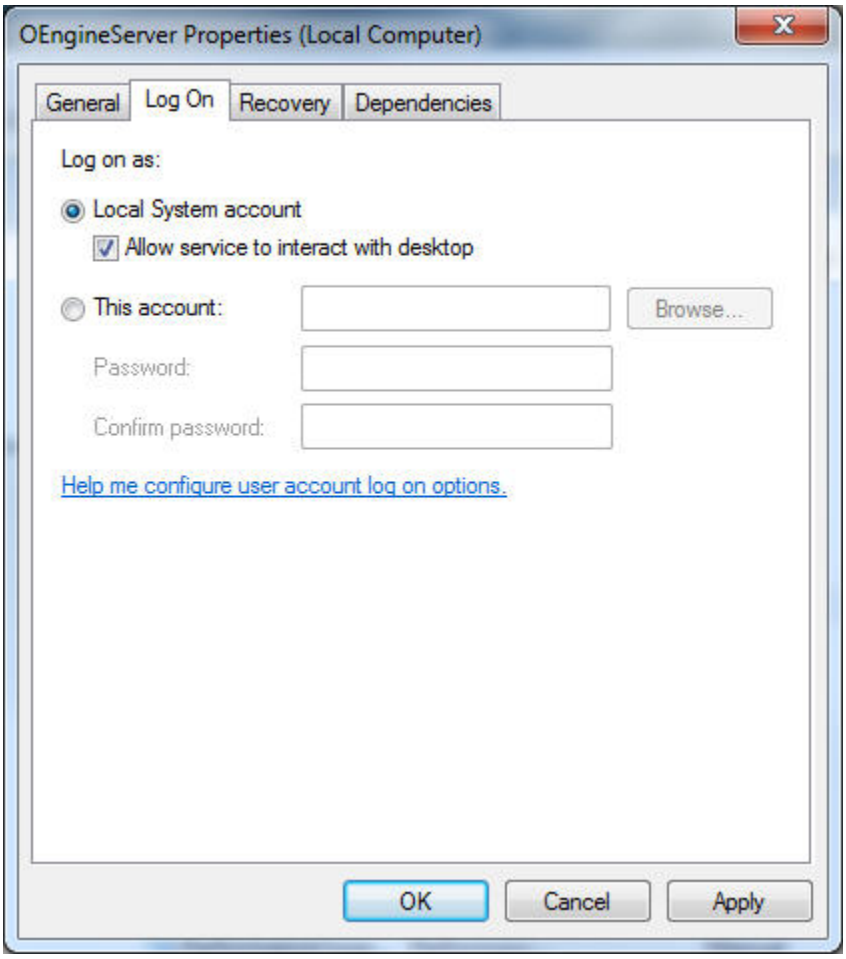



The OEngineServer service may be removed by double-clicking on the UninstallApp-NT.bat file found in the OESERVER folder within your OpenInsight directory.

The OEngineServer will be installed as a service.  It will need to be started.



Right click on the OEngineServer service to access the properties panel. **Make sure that Allow service to interact with desktop is checked.**

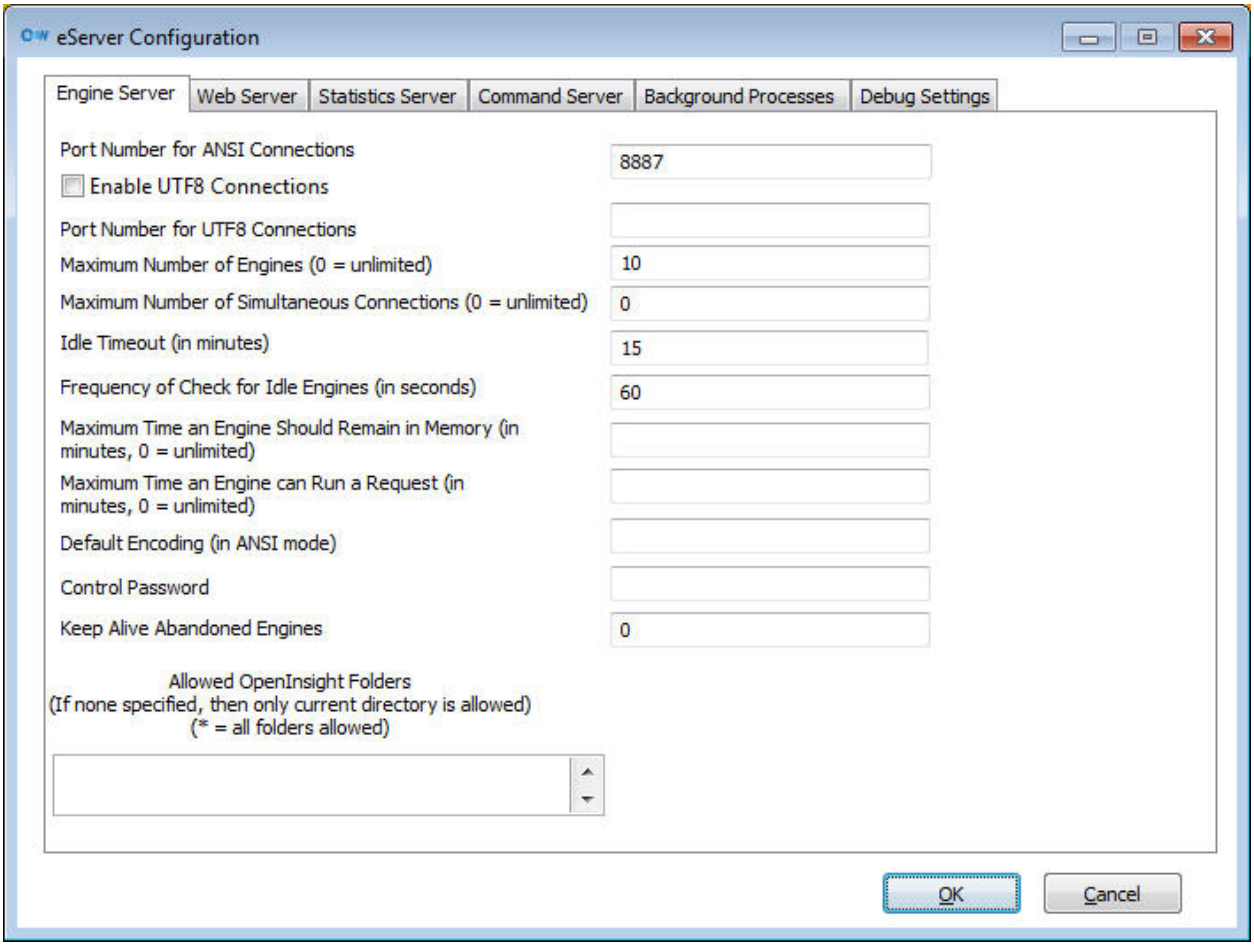


# Section IV: Configuring your OEngineServer port

By default the OEngineServer is configured to listen on port 8088. You can change this port by editing the **eserver.cfg** file located in your OpenInsight directory using the eServer Configuration form from either the Application Manager or the OI Console.

The default contents of the eserver.cfg file are as follows but should be maintained using the eServer Configuration form:

```
#Tue Mar 18 10:28:36 EDT 2014
//Statistics_Days=14
IdleCheck=60
Password=
MaxConnections=0
MaxEngines=10
TimerProc=
MaxUpTime=0
StartupFlags=65
StartupProc=
ShutdownFlags=1
Procedure=
//Statistics_Directory=stats
UserName=
Server=
MaxRunTime=0
//Statistics_ID=MAIN
PortNumber=8088
Procedure_1=REVCMD_LISTENER
IdleTimeout=15
EControlPassword=XMl04WqFXR9U8MgyNh647UgQnbEG2yqw
//Statistics_Input=8090
Mode=
//Statistics_Output=localhost\:8090
ShutdownProc=
Procedure_=JD3_LISTENER
Application=
//=Process to run for mode -1,0,2 at 'idle check'
```




Basic configuration of the engine server allows clients to communicate (through either ANSI/ASCII or UTF8 encoding) with OEngines.

**Port Number for ANSI Connections** (eserver.cfg setting **PortNumber)**: Specifies the port number the engine server will "listen" on for ASCII/ANSI requests. Default is 8088.

**Enable UTF8 Connections** (eserver.cfg setting **UTFPort_Disabled**): select whether UTF8 connections will also be allowed.

**Port Number for UTF8 Connections** (eserver.cfg setting **UTFPortNumber**): If UTF8 connections are allowed, this field specifies the port number the engine server will "listen" on for UTF8 requests.

**Maximum Number of Engines** (eserver.cfg setting **MaxEngines**): Specifies the maximum number of oengines that the engine server can start up (0=unlimited). Default is 10.

**Maximum Number of Simultaneous Connections** (eserver.cfg setting **MaxConnections**): Specifies the maximum number of simultaneous connections to the engine server (0=unlimited). Default is 0.

**Idle Timeout** (eserver.cfg setting **IdleTimeout**): Specifies the time (in minutes) after which an idle oengine will be shut down by the engine server. Default is 15.

**Frequency of check for idle engines** (eserver.cfg setting **IdleCheck**): Specifies the time (in seconds) between checks by the engine server for idle engines, MaxUpTime engines, and MaxRunTime engines. Default is 60.

**Maximum time an engine should remain in memory** (eserver.cfg setting **MaxUpTime**): Specifies the maximum amount of time (in minutes) an Oengine should be allowed to stay active in memory (not necessarily running that entire time, just not idle long enough to trigger the idle timeout when it _is_ idle). Default is 0, which means "unlimited". Note that the idlecheck value will also be used to determine the frequency of the MaxUpTime checks.

**Maximum time an engine can run a request** (eserver.cfg setting **MaxRunTime**): Specifies the maximum amount of time (in minutes) that an oengine should be allowed to run a particular process before it's considered "hung" and forcibly terminated. Default is 0, which means "unlimited". Note that the idlecheck value will also be used to determine the frequency of the MaxRunTime checks.

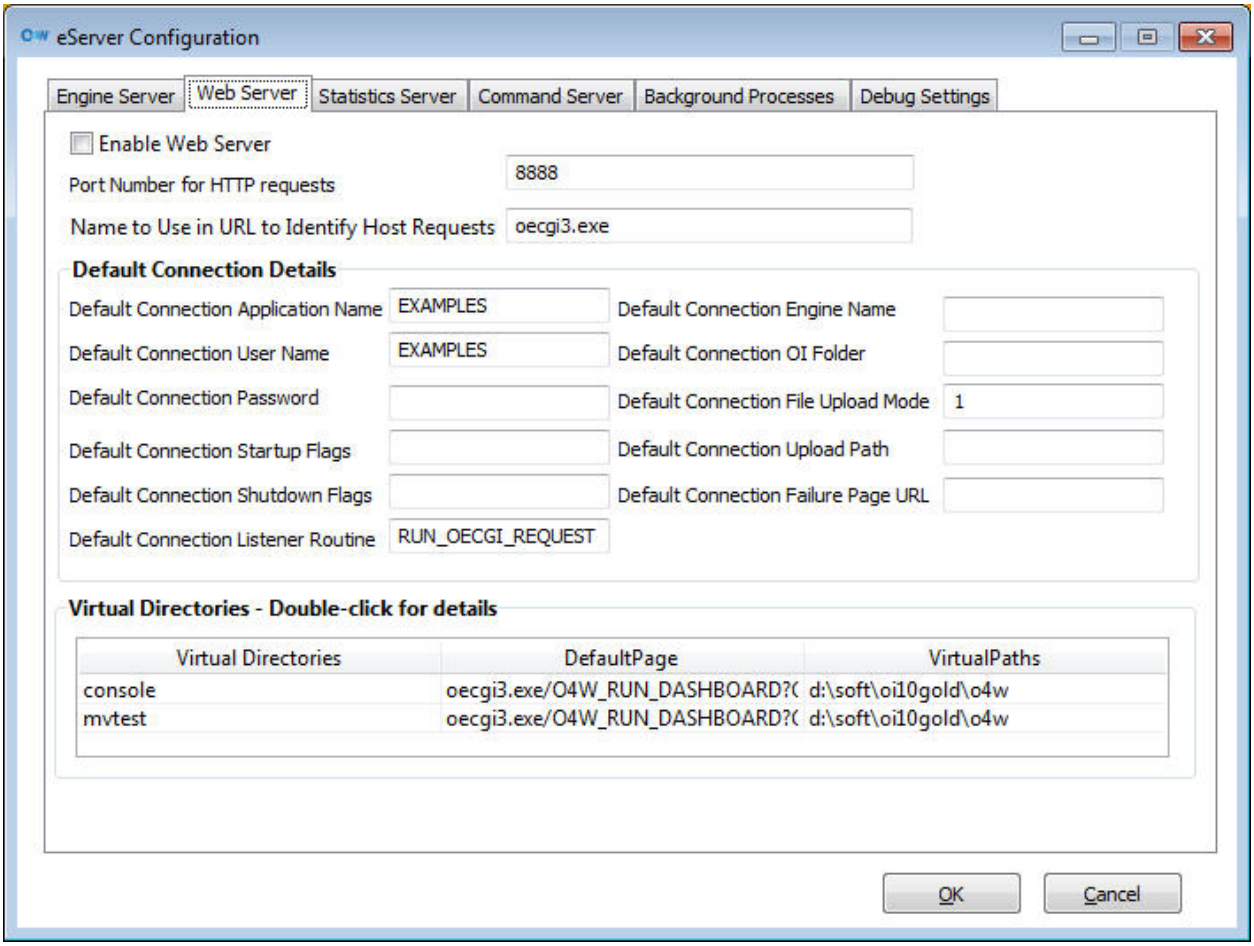
(eserver.cfg setting **MaxNumToClose**): The maximum number of oengines to close at one time. Default is "0", which is unlimited. If there are more oengines that need to be closed, they will be closed during the next idle check. This value is designed to help "stagger" the logging on and off of OEngines.

**Default encoding** (eserver.cfg setting **Encoding**): Specify the encoding used for ansi communication. The default is 8859_1. The default (8859_1) is appropriate for most sites.

**Control Password** (eserver.cfg setting **ControlPassword**): Specify the new password to use to control access to the eserver.cfg.

**Keep alive abandoned engines** (eserver.cfg setting **KeepAlive**): Set to "1" if telnet sessions that are dropped should be kept alive as "zombie processes" (with the possibility of being reattached"), or "0" if sessions that are dropped should be terminated. Note that this only applies to telnet sessions. Default value is "0".

**Allowed OpenInsight folders** (eserver.cfg setting **OILocations**): Specifies one or more folders the engine server is allowed to start an oengine in (when a client makes a connection, it can request that the oengine be started in a different folder; that different folder must be specified in the OILocations value). An asterisk ("*") means all locations are allowed. The default value is null, which means only the current directory is allowed.




The engine server can provide web server functionality to serve up static (HTML) and dynamic (O4W) output directly, without requiring Microsoft IIS or Apache. The web server (based on the open-source Jetty server) intentionally provides limited functionality, designed to handle the OI Console and less-heavily-loaded O4W sites with an easy to manage and configure system.

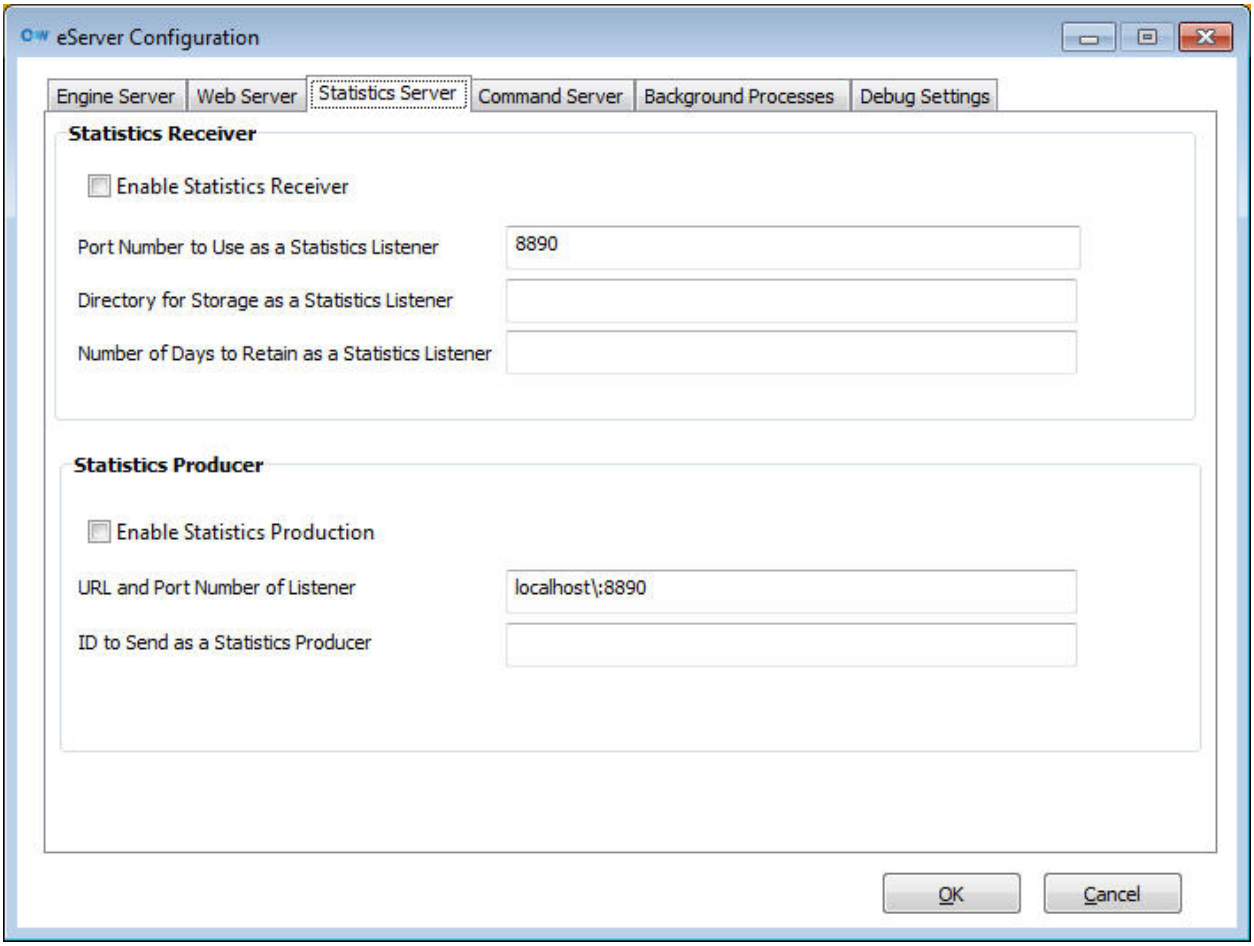
**Enabled Web Server** (eserver.cfg setting **WebServer_Disabled**): Select to enable or disable the built-in web server functionality.

**Port Number for HTTP requests** (eserver.cfg setting **WebServerPortNumber**): If the web server functionality is enabled, this is the port number on which the web server should "listen." Default value is 8888.

**Name to use in URL to identify host** (eserver.cfg setting **WebServerCGIProcedure**): When using IIS or Apache, clients normally use OECGI3.EXE or OECGI4.EXE (or OECGI3P.PHP or OECGI4P.PHP) to initiate the CGI request that communicates between the web server and the engine server. Since, in this case, the engine server is acting *as* the web server, there is no actual cgi routine needed to perform the communication. However, the web server must still know when a request is for a static (HTML) page, versus a dynamic (CGI) page, so that it can be properly processed. The WebServerCGIProcedure specifies the name in the url that signifies this is a dynamic page, and thus this should be processed as though it were coming from the cgi procedure. Multiple values may be specified, semicolon delimited. By default, this value is oecgi4.exe;oecgi3.exe

**Default Connection Details** (eserver.cfg setting **WebServerConnection_default**): Specifies all the default values needed to "connect" to an OEngine. This list of settings includes application name, user name, password, startup flag, shutdown flag, "listener" routine name, engine name, OI folder, file upload mode, file upload path, and "failure page" URL. These values are identical to those used in the registry for the actual OECGI3/OECGI4.

**Virtual Directories** (eserver.cfg setting **WebServerVirtualDirs**): A list of the "virtual directories" that the web server should recognize. These are comparable to the "virtual directories" that are set up in IIS, for example. For each virtual directory, you will need to specify all the values needed to "connect" to an OEngine. This list of settings includes application name, user name, password, startup flag, shutdown flag, "listener" routine name, engine name, OI folder, file upload mode, file upload path, and "failure page" URL. These values are identical to those used in the registry for the actual OECGI3/OECGI4. In addition, you will specify the "default page" (the name of the html document to return if no specific page is specified), and the full path to the actual folder that the virtual directory represents.




The engine server can collect statistics on the requests and responses it generates. In order to prevent the collection of statistics from impacting on performance, these statistics may be processed by a different engine server (one that is not being used to handle real-time production requests, for example). Any engine server can be configured as a "receiver" of statistics information from another server, a "producer" of statistics for consumption by another engine server, or both a receiver and producer (which is most appropriate for less heavily-loaded systems).

**Enable Statistics Receiver** (eserver.cfg setting **Statistics_Input_Disabled**): Select whether or not this engine server should be able to receive statistics information.

**Port Number to Use as a Statistics Listener** (eserver.cfg setting **Statistics_Input**): If statistics receiving is enabled, this is the port number to use when acting as a statistics receiver.

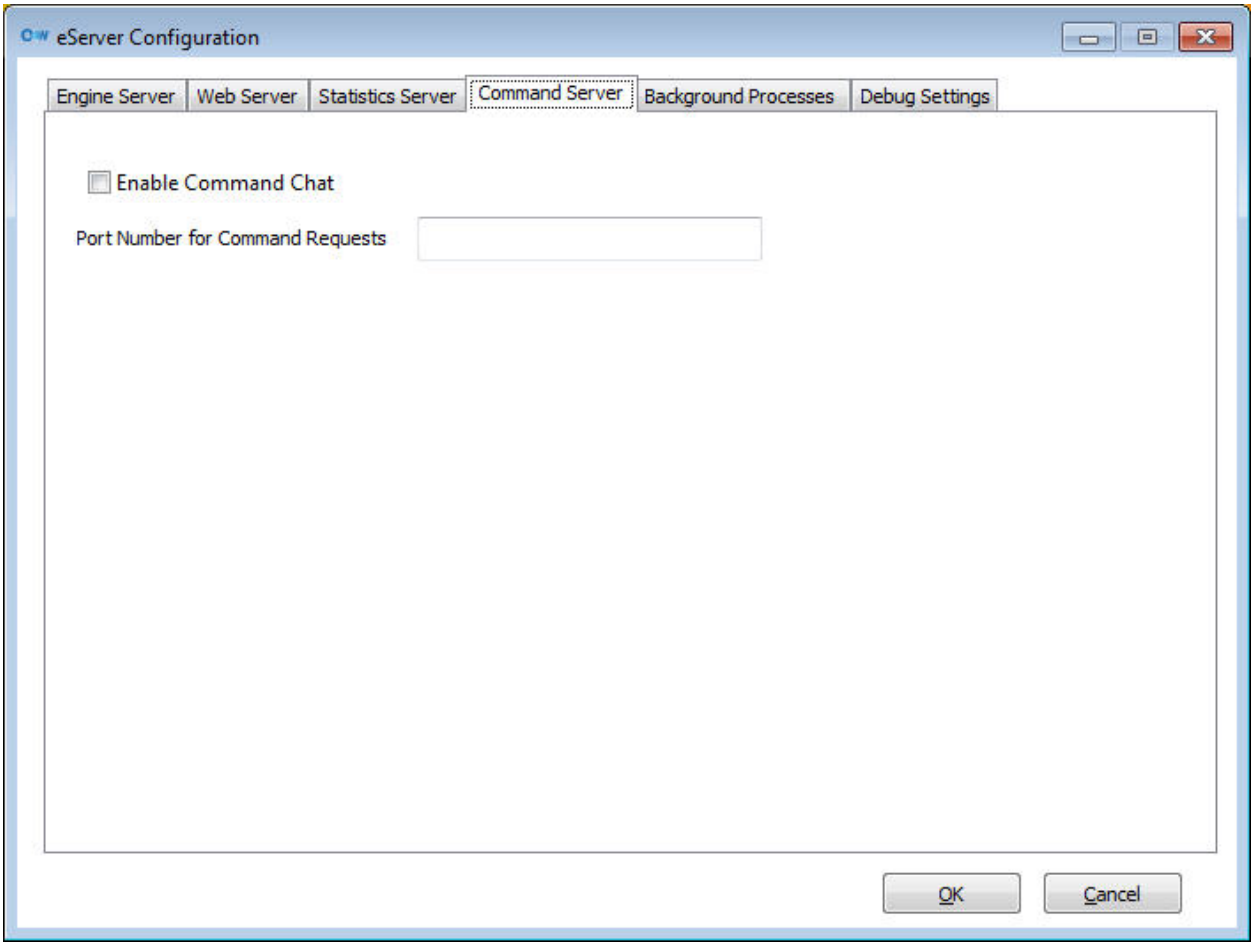
**Directory for Storage as a Statistics Listener** (eserver.cfg setting **Statistics_Directory**): The directory (which may be a full path) where the statistics information will be written. The default is "stats" (ie, a relative path under the OI folder named "stats").

**Number of Days to Retain as a Statistics Listener** (eserver.cfg setting **Statistics_Days**): The number of days of statistics information to retain. Default is 14.

**Enable Statistics Production** (eserver.cfg setting **Statistics_Output_Disabled**): Select whether or not this engine server should produce statistical information.

**URL and Port Number of Listener** (eserver.cfg setting **Statistics_Output**): If this engine server is generating statistics information, this is the URL and port number (colon-delimited) of the statistics _receiver_ this statistics _producer_ should be sending the output to.

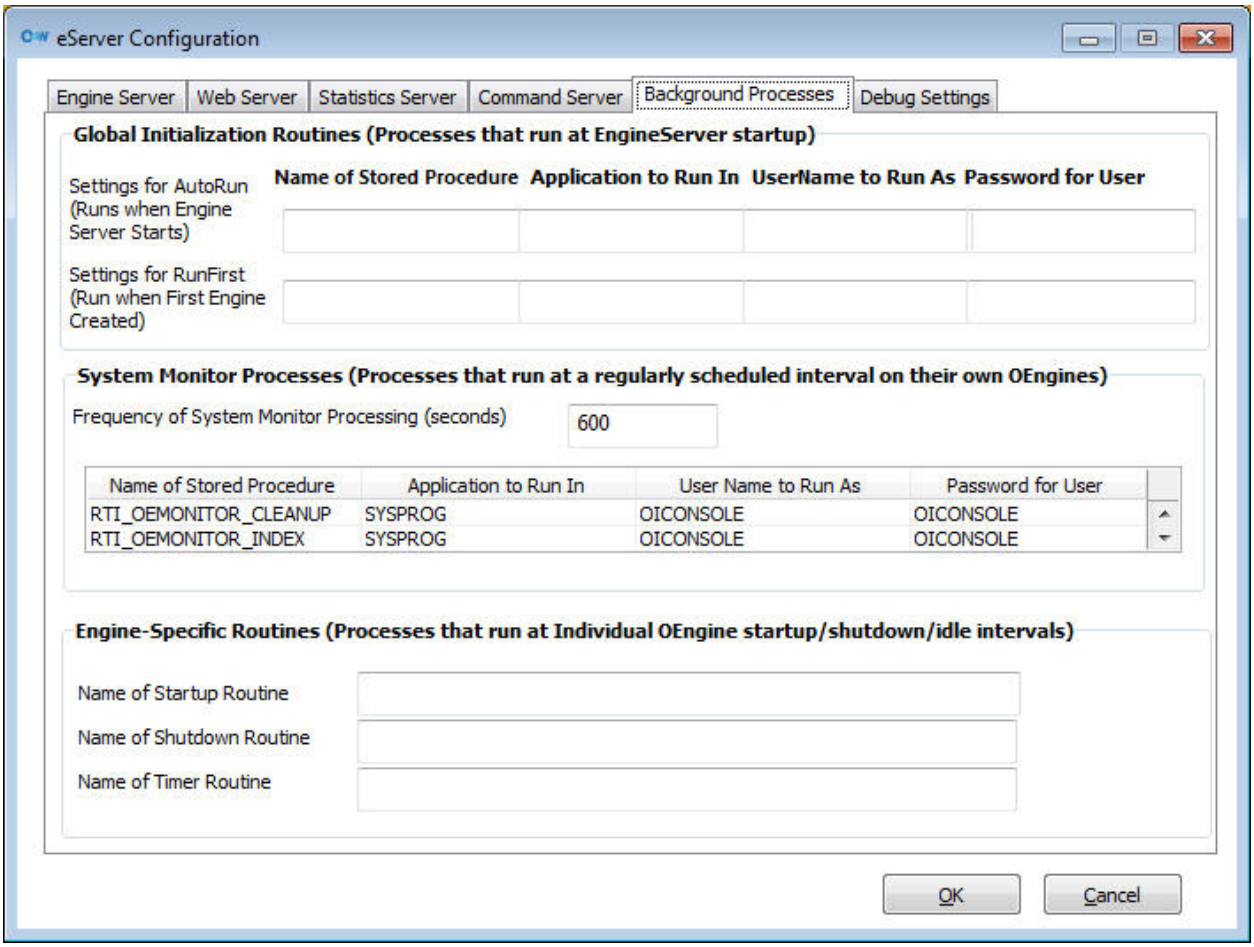
**ID to Send as a Statistics Producer** (eserver.cfg setting **Statistics_ID**): The unique short identifier that should be used to identify the statistics coming from this engine server. This is only important if multiple engine server statistic producers are all sending their information to a single statistic receiver. Default is "01".




The engine server can communicate with individual OpenInsight desktop clients via the "command channel." This will allow for management of the desktop clients from the central server.

**Enable Command Chat** (eserver.cfg setting **CommandPort_Disabled**): Select whether the command port functionality is enabled.

**Port Number for Command Requests** (eserver.cfg setting **CommandPort**): The port number that will be used for the "command chat" (not yet implemented).




The engine server can run various processes automatically, for system maintenance, initialization, etc. Some of these processes run only one time (AutoRun and RunFirst), creating their own OEngines, running, and then terminating those OEngines. Others run at a specified interval (SystemMonitor), again creating their own background OEngines, running, and then terminating. Still others run in any OEngine that has been started in response to a client request (StartupProc, ShutdownProc, and TimerProc).

**Settings for AutoRun** (eserver.cfg setting **AutoRun**): If specified, this is a list of routine name, application name, user name, and password, which specifies the routine to run (and where to run that routine) when the engine server initially starts up.

**Settings for RunFirst** (eserver.cfg setting **RunFirst**): If specified, this is a list of routine name, application name, user name, and password, which specifies the routine to run (and where to run that routine) when the very first OEngine is created.

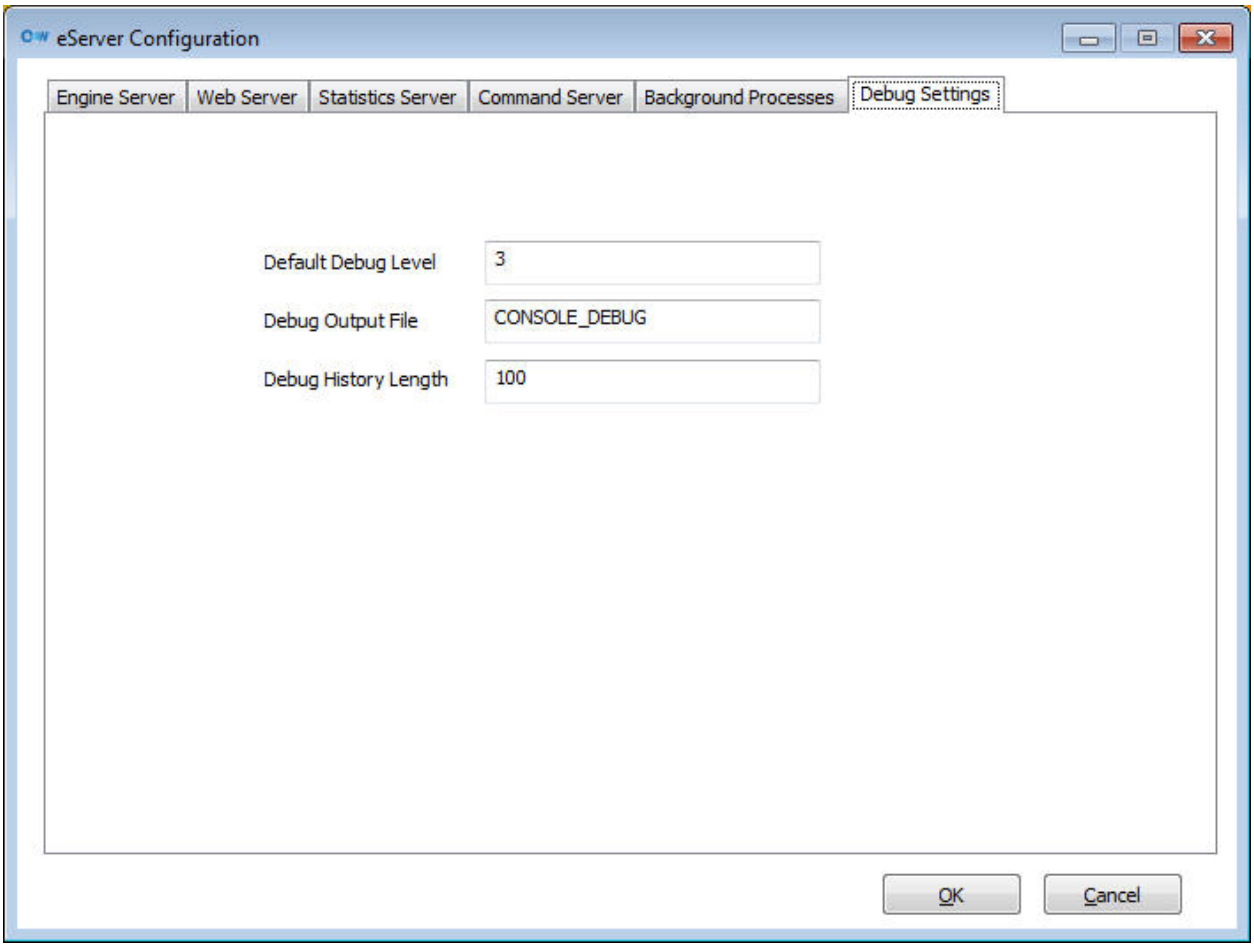
**Frequency of System Monitor Processing** (eserver.cfg setting **SystemMonitorTime**): The time (in seconds) between invocations of background processes. Default is 60.

**System Monitor Processes** (eserver.cfg setting **SystemMonitor**): list of processes to run at a regular interval (as specified in SystemMonitorTime) Each entry consists of the procedure name, application name, user name, and password (comma-delimited).

**Name of Startup Routine** (eserver.cfg setting **StartupProc**): The name of the stored procedure to run when an OEngine starts up.

**Name of Shutdown Routine** (eserver.cfg setting **ShutdownProc**): The name of the stored procedure to run when an OEngine shuts down.

**Name of Timer Routine** (eserver.cfg setting **TimerProc**) : The name of the stored procedure to run every time the "idle check" is run.




The engine server can generate more verbose output to aid in debugging and diagnostics. This information can be displayed on-screen, when the engine server is run via the java command line, or captured to a file.

**Default debug level** (eserver.cfg setting **DebugLevel**): A number indicating the amount of detail the diagnostic output should display. Higher numbers indicate more output (0=none).

**Debug output file** (eserver.cfg setting **DebugFile**): The full path and file name of the output file the diagnostic information should be recorded in.

**Debug history length** (eserver.cfg setting **DebugHistoryLength**): The maximum number of lines of output that should be stored.

# Section V: OEngineServer Specifications

**Introduction**

The Revelation Software EngineServer is a java application designed to:
1. Listen (on a configurable tcp/ip port) for requests from various client applications;
2. Start up, manage, and terminate OpenEngines (OEngines) - the database engine for OpenInsight;
3. Route requests and responses from the clients to the OEngines

It manages communication with the OEngines via Revelation Softwares' REVCAP32 interface (through a "Native Methods Interface"). This means that, although java is a cross-platform language and the EngineServer per se can be run on any support java platform, we are in practice limited to Windows platforms unless/until a REVCAP32-like interface is available in any other environment.

It manages communication with the various clients via tcp/ip, using a protocol first described for the "JD3" open source project. This protocol has been extensively extended for the EngineServer, but at its heart it consists of an 8 character "length" string, followed by the actual message content. The content is typically a command code, followed by a delimiter, and then any other parameters needed for that command code.

The EngineServer can also perform some operations independent of any client requests - specifically, a "startup" process, "shutdown" process, and "timer" process can be executed on any OEngines that are created, and an overal "startup" (also known as "coldstart") process can be defined to run when the EngineServer is first started.

The EngineServer is configurable via the "eserver.cfg" configuration file (located in the OI directory), which contains overall server settings as well as default settings for the various types of client connections.

The types of client connections available include:

mode -1: "JD3" (stateless, block mode communication, responding to the JD3 list of commands);
mode 0: "Block" (stateless, block mode);
mode 1: "Char" (stateful, character interaction);
mode 2: "Web" (stateless, block mode)

In addition, it is possible to create an additional type of OEngine:
[mode n/a]: "Phantom" (stateful, character interaction - must be spawned from another OEngine)

The client connection type controls how the EngineServer communicates with the OEngine, as well as what is expected of the OEngine itself. When a request comes in while in a "stateless" mode, the EngineServer looks for an OEngine with the specified "signature" (based on the app, user, password, and dispatch routine) in a queue it maintains of "available OEngines"; if one is found, then it is tasked to execute the request, and then released back into the queue. The _client_ is responsible for maintaining any required state information; _any_ of the stateless OEngines may be used to satisfy _any_ requests from _any_ client (so long as it has the required "signature"). Stateful connections, on the other hand, are dedicated to an individual client; the EngineServer maintains a 1-to-1 connection between the client and the OEngine, and thus subsequent requests from the same client will always go to the same OEngine.

The number of OEngines kept in the "available queue", as well as when OEngines should be shut down and removed from the "available queue", is configured in the eserver.cfg file. As mentioned above, it is also possible to configure routines to run at startup, shutdown, and on a "timer-like" event, for these stateless OEngines.

Revelation Software-available clients include the CTO/AREV32 clients (using "char" mode), OECGI2/OECGI3/OIWebWizard/OI4Web (using "web" mode), and NetOI (using "JD3" or "web" mode); the OIJio library is also available for java developers wishing to communicate with the EngineServer.

Note that the EngineServer is occasionally referred to as the "Socket Server" or "OESocketServer" interchangably.

**Running the EngineServer**

The EngineServer can be run from the DOS command line via the following command:
java -jar oesocketserver.jar {-l <configfilename>} {-d <debuglevel>}
where <debuglevel> is an optional number ranging from 1 to <n>; the higher the number, the more detailed information is displayed.

Revelation Software also provides a "wrapper" to enable the java command to run as a service; it is installed via the "installapp-nt.bat" routine located in the OESERVER subdirectory of the OpenInsight directory. This bat file reads configuration information from the wrapper.conf file located in the conf subdirectory.

Note that it is possible to run multiple EngineServer services on a single Windows system, but that to do so requires a modification to the installapp-nt.bat file to read a modified wrapper.conf (specifically, the wrapper.console.title, wrapper.ntservice.name, and wrapper.ntservice.displayname should be unique for each service)

**Client Communication With The EngineServer**

As described above, the client communicates with the EngineServer via a standard message format:

nnnnnnnncd<xxxx>

where nnnnnnnn is an 8-byte length string, c is a command character, and d is a delimiter (char(1)); if the command code requires additional parameters, these then follow the delimiter (and may be delimited as well).

The supported codes from the client for the EngineServer include:

-1 - Return list of engines
-2 - Request shutdown of engine(s)
-3 - Force shutdown of engine(s)
-4 - Spawn phantom thread
-5 - Return time of engine startup
1 - Login
2 - Logoff
3 - Execute
4 - Call
18 - Continue
19 - Input
100 - break
101 - debug
400 - call Async

Note that codes > 0 require that the OEngine be logged in to an application, otherwise an error is returned.

The EngineServer responds to the client with the same protocol (nnnnnnnn<xxxx>).

CLIENT 1 - Login Command

The Login command has the following format:
1<d><uname><d><pwd><d><appl><d><mode><d><server><d><proc><d><upflags><d><downflags><d><OILoc><d>

where <d> is char(1), <uname> is the username, <pwd> is the password, <appl> is the application name, <mode> is the type of connection desired, <proc> is the name of the OI stored procedure that will act as the "listener" or "dispatch routine", <upflags> are any startup flags, <downflags> are any shutdown flags, and <OILoc> is the directory where the desired OEngine resides.

Valid <mode> choices are "-1" (JD3), "0" (block), "1" (char), and "2" (web). Provided dispatch routines are JD3_LISTENER (for JD3 mode), REVCMD_LISTENER (for char mode), and RUN_OECGI_REQUEST (for web mode).

Default values for these can be configured in the eserver.cfg for each appropriate <mode>.

If you wish an OEngine to be started in any directory other than the current directory, then you must have that target directory listed in the eserver.cfg list of allowed directories.

If the OEngine is already logged in, this message (when in "char" mode) can be used to simulate a "logto" another application/username/password.

If the login was successful, the EngineServer response will be:

{<okcode>}0<d><engineInfo><d><engineDetails><d><expInfo><d>

If the login failed, the response will be:

{<errcode>}1<d><errMsg><d>

The <okcode> and <errcode> values ("0" and "1", respectively) will be prefixed to the message if we are running in any mode other than JD3 mode.

CLIENT 2 - Logoff Command

The logoff command has the following format:

2<d>

The response from the EngineServer will be:

{<okcode>}0<d>

CLIENT 3 - Execute Command

Invokes the previously-established "dispatch routine", passing in the specified command; in JD3 mode, also passes in the "Execute" command code. In non-JD3 mode, multiple delimited parameters are turned into comma-delimited, invidually quoted strings for the "dispatch routine"; in JD3 mode, the entire command is assembled into a single delimited string.

The execute command has the following format:
3<d><param1><d>{<param2><d>{<param3><d>...}}

The response from the EngineServer will vary depending on the message(s) generated by the OEngine.

CLIENT 4 - Call Command

Invokes the previously-established "dispatch routine", passing in the specified command; in JD3 mode, also passes in the "Call" command code. In non-JD3 mode, multiple delimited parameters are turned into comma-delimited, invidually quoted strings for the "dispatch routine"; in JD3 mode, the entire command is assembled into a single delimited string.

The call command has the following format:
4<d><param1><d>{<param2><d>{<param3><d>...}}

The response from the EngineServer will vary depending on the message(s) generated by the OEngine.

CLIENT 18 - Continue Command

This command is used when the client is responding to a "PROCESSING" message from the OEngine.

The continue command has the following format:
18<d>

The response from the EngineServer will vary depending on the message(s) generated by the OEngine, if the routine is still running; if the routine has terminated, the EngineServer responds with

{<okcode>}0<d>

CLIENT 19 - Input Command

This command is used when the client is responding to an "INFO_REQUEST" message from the OEngine.

The input command has the following format:
19<d>{<response><d>}

The response from the EngineServer will vary depending on the message(s) generated by the OEngine.

CLIENT 100 - Break Command

This instructs the EngineServer to try and halt execution of the routine that has been submitted to the OEngine.

The break command has the following format:
100<d>

The response from the EngineServer will vary depending on the message(s) generated by the OEngine.

CLIENT 101 - Debug Command

This instructs the EngineServer to try and invoke the debugger during execution of the routine that has been submitted to the OEngine.

The debug command has the following format:
101<d>

The response from the EngineServer will vary depending on the message(s) generated by the OEngine, if the routine is still running; if the routine has terminated, the EngineServer responds with

{<okcode>}0<d>

CLIENT 400 - Call Async Command (JD3 mode only)

This is identical to the "call" command, but in addition it sets the "async" flag to indicate that the client wishes to respond to any messages generated during execution of the routine that has been submitted to the OEngine. Note that it is only applicable to JD3-mode connections.

CLIENT -1 - Currently-Running Engines Command

Returns a list of the currently-running engines,and their state, either in HTML or delimited text. The format of the command is:

-1<d>{<charflag><d>}

where <charflag> is "1" if the request is coming from a "char" mode connection; in this case, the data is returned in @FM/@VM delimited strings.

If not in "char" mode, the response from the EngineServer would be:

0<d><response><d>

CLIENT -2 - Shutdown Request Command

Instructs the EngineServer to attempt to shut down the specified OEngines if they are currently "idle". The format of the shutdown request command is:

-2<d><controlpwd><d>{<param1><d>{<param2><d>...}}

where <controlpwd> is the control password (defined in the configuration file); if this is not set, or is not correct, the EngineServer response is:

1<d>Access Denied<d>

If the password is specified correctly, the additional parameters specify which OEngines should be selected for shutdown. The EngineServer then responds with:

0<d>Idle Engines shut down<d>

CLIENT -3 - Force Shutdown Command

Instructs the EngineServer to attempt to shut down the specified OEngines regardless of whether they are currently "idle". The format of the shutdown request command is:

-3<d><controlpwd><d>{<param1><d>{<param2><d>...}}

where <controlpwd> is the control password (defined in the configuration file); if this is not set, or is not correct, the EngineServer response is:

1<d>Access Denied<d>

If the password is specified correctly, the additional parameters specify which OEngines should be selected for shutdown. The EngineServer then responds with:

0<d>All Engines shut down<d>

CLIENT -4 - Phantom Logon Command

Instructs the EngineServer to start up a "phantom" OEngine (one which has no connection to any client). Any output generated by the phantom process will be discarded; any requests for input (that are not satisfied by DATA statements in basic+) will cause the phantom process to terminate.

The format of the phantom request is:

-4<d>{<addlcommands><d>{<addlcommands><d>...}}

where <addlcommands> are any additional commands to pass in to the phantom creation process. These are the same commands documented here, EXCEPT that a "sub delimiter" (char(2)) is used to delimit the pieces of the command (so as to not interfere with the top-level delimiters).

Normally, the <addlcommands> will include the login command, and the call command; for example:

-4<d>
1<sd><uname><sd><pwd><sd><appl><sd><mode><sd><server><sd><proc><sd><upflags><sd><downflags><sd><OILoc><sd><d>
3<sd><param1><sd><d>

(mode should be "3" to indicate "phantom processing).

The response from the EngineServer will be:

0<d>Phantom started<d>

CLIENT -5 - Status Info Command

This requests the time that the OEngine started, and the current time. The format of the request is:

-5<d>

The response from the EngineServer will be:

0<d>Startup: <yyyy-mm-dd hh:mm:ss>; Current <yyyy-mm-dd hh:mm:ss><d>

**EngineServer Communication With The OEngine**

Once the EngineServer has started an OEngine (either through a client request, or through automatic startup), it communicates with the OEngine via the REVCAP32 API. Processing normally involves issuing a command through the RevSend API call, waiting for a response (either through RevPoll, if in "char" mode or if the request is the callAsync command, or through RevWait otherwise), acting on the response, and "wrapping up" when the request has finished processing.

The EngineServer will respond to the following REVCAP32 messages during the RevPoll/RevWait processing:

- "UNPROCESSED" - handled internally always and processing continues;
- "PROCESSING" - If in "char" mode, or executing a callAsync command, after 10 "processing" messages, a response to the client will be sent (to enable "break" functionality); in any other situation, the EngineServer just pauses before resuming processing;

- "DATA_AVAILABLE" - If in "char" mode, or executing a callAsync command, returns any provided data to client; if in "phantom" mode, discards the data and continues processing; in any other situation, the output is accumulated until processing is complete;
- "COMPLETED"/"PROC_ERROR" - any "status text" generated by the call is retrieved, and the output returned to the client;
- "INFO_AVAILABLE" - If in "char" mode, or executing a callAsync command, returns any provided data to client; in any other situation, the output is accumulated until processing is complete;
- "INFO_REQUEST" - If a special OEngine command (see below), handled internally and processing continues; if in "char" mode or executing a callAsync command, request is sent to client; if in "phantom" mode, the phantom is terminated; in any other situation, a dummy (null) response is sent to the OEngine and processing continues

**Commands From The OEngine**

In addition to commands that a client might issue to the EngineServer, a stored procedure running on the OEngine, itself, may also issue specific EngineServer commands.

The command request from the Oengine is sent to the EngineServer via the INFO_REQUEST REVCAP32 message, and has the following format:

<cmdindicator><cmdcode>

where <cmdindicator> is char(6), and command code is one of the values below:

1 - Return a list of serial numbers this EngineServer can support;
2 - Return a list of the currently-running OEngines;
3 - Logon a phantom process;
4 - Return the version and engine information;
5 - Return a list of printers available to the EngineServer;
6 - Return a flag if this is a phantom process;
7 - Return the time the Engine started, and the current time

OENGINE 1 - Serial Number List Command

Format of request to EngineServer:

\06\1

Response from the EngineServer: List of licensed serial numbers for allowed OEngines

OENGINE 2 - Currently-Running Engines Command

Format of request to EngineServer:

\06\2

Response from the EngineServer: @FM/@VM delimited list of currently running engines and their status

OENGINE 3 - Phantom Logon Command

Format of request to EngineServer:

\06\3{<addlcommands><d>{<addlcommands><d>...}}

where <addlcommands> are any additional commands to pass in to the phantom creation process. These are the same commands documented in the client commands section, EXCEPT that a "sub delimiter" (char(2)) is used to delimit the pieces of the command (so as to not interfere with the top-level delimiters).

Normally, the <addlcommands> will include the login command, and the call command; for example:

\06\3
1<sd><uname><sd><pwd><sd><appl><sd><mode><sd><server><sd><proc><sd><upflags><sd><downflags><sd><OILoc><sd><d>
3<sd><param1><sd><d>

(mode should be "3" to indicate "phantom processing).

The response from the EngineServer will be: Phantom started

OENGINE 4 - Version Command

Format of request to EngineServer:

\06\4

Response from the EngineServer: <version><space><servername>

OENGINE 5 - Printers Command

Format of request to EngineServer:
\06\5

Response from the EngineServer: @VM-delimited list of printers

OENGINE 6 - Phantom Flag Command

Format of request to EngineServer:

\06\6

Response from the EngineServer: "0" (if the OEngine is not a phantom), or "1"

OENGINE 7 - Status Info Command

Format of request to EngineServer:

\06\7

Response from the EngineServer: "Startup: "<yyyy-mm-dd hh:mm:ss>"; Current: "<yyyy-mm-dd hh:mm:ss>

REVELATION
SOFTWARE

**Revelation Software, Inc**
99 Kinderkamack Road Ste 109
Westwood, NJ 07675
U.S.A
Toll Free: 800-262-4747
Phone: 201-594-1422
Fax: 201-722-9815
www.revelation.com

**Revelation Software Ltd.**
Boundary House
Boston Road
London, W7 2QE
U.K.
Phone: +44 0 208 912 1000
Fax: +44 0 208 912 1001
info@revsoft.co.uk

**BrightIdeas New Zealand**
44 Cockle Bay Rd, Howick
Auckland, 2014
New Zealand
Phone: +64 9 534 9134
info@revelation.asia

Revelation Software is a division of Revelation Technologies, Inc.

Part No. 414-966