K2 blackpearl Logging and Auditing

ERROR AND AUDITING LOGGING FRAMEWORK ON K2 BLACKPEARL

October 6

CORPORATE HEADQUARTERS	EMEA HEADQUARTERS	APAC HEADQUARTERS	[WWW.K2.COM]
4042 148th Avenue NE Redmond, WA 98052 USA **PH** +1 (425) 883 4200 **FAX** +1 (425) 671 0411	26 Worple Road Wimbledon London UK **PH** +1 44 (0) 845 612 0912 **FAX** +1 44 (0) 845 612 0911	9 Shenton Way #06-02 Singapore 068813 **PH** +1 65 6327 4110 **FAX** +1 65 6327 4120	

CONTENTS

INTRODUCTION

K2 blackpearl includes a native auditing and logging framework that enables Administrators to monitor and troubleshoot the K2 blackpearl environment. System Logging is initiated automatically upon installation and continues through to runtime, whereas system auditing features are initiated manually at process design time and leveraged during runtime. The logging and auditing features of K2 blackpearl are built into the platform but operate and are configured in different ways. This whitepaper discusses each feature in detail.

Logging and Auditing Features List

Auditing	• Processes • Default Reports • Data fields * • XML Fields * • Workflow Server
Logging	• Installer • SmartBroker • SmartObjects*

*** Some additional coding setup and or configuration may be required for this feature to function correctly**

DATA CAPTURE METHODS

The product of logging and a requirement for auditing is a data stream that must be captured and stored. This provides a historical review of system activity. The following options are available for the capture of the data as it is produced.

- Log file
- Database
- MSMQ
- Console
- Event Extension

K2 BLACKPEARL LOGGING

The logging framework is fully integrated into the product starting from the installation of K2 blackpearl and continues through to runtime.

The logging framework produces two types of messaging -- short hand and long hand. The short hand version is preferably written only to the console and less descriptive in the amount of detail that it provides. When additional detail is required, the long hand version is enabled and provides extended information, but is written to a database. Given the extensiveness of the detailed information, using a log file as the destination for the data is not advised as the result is a text file that is very difficult to find meaningful information in the data. What cannot be ignored as well is the need for an individual, skilled in the error reporting of K2 blackpearl to interpret the content of the file. This method may also be more time consuming to prepare meaningful reporting data as well.

When stored in a database, the K2 out-of-the-box (OOB) reports can also be employed to produce meaningful reports on specific elements with the system that have been selected as the basis for the report.

The diagram below illustrates the options. The configuration file configures the K2 Server on startup. The Administrator can write the log details to a console or to a log file and the log database. From the log database the detail is accessible using the K2 OOB reports or a custom report using the K2 Service Broker.

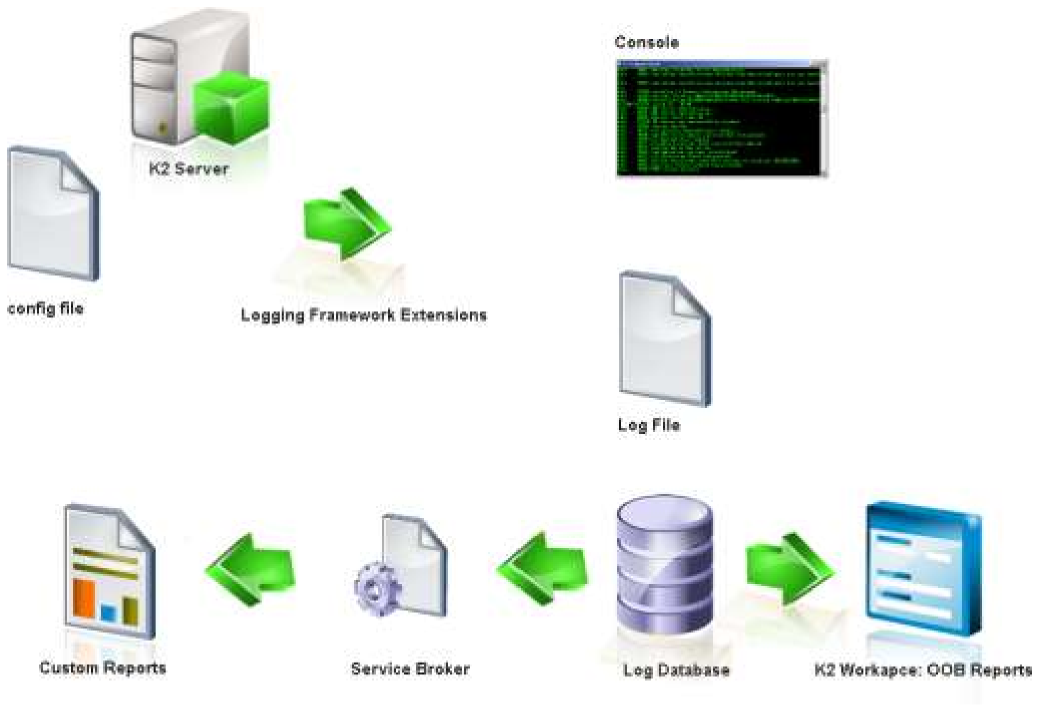


Figure 1 - Logging Options

K2 BLACKPEARL AUDITING

Auditing is primarily a runtime system attribute which enables the Managers, System Administrators or Team of Auditors to examine system activity. The use of auditing is not to troubleshoot errors or manage system performance but rather to determine the correctness of system activity and to determine definitively if the procedures followed by a process were actually carried out when compared with a procedural protocol.

System auditing must be enabled manually during process design time and is never enabled automatically. Specifically for data fields, if auditing has not been enabled at design time, reports cannot be generated from auditing data that are not available.

When data on a process is required, the audit begins at deployment. As soon as the developer deploys the process to the server, a system audit trail starts on the process at its first version. If a data field is not enabled for audit at design time the process must be redeployed to the server with auditing enabled.

In this case, auditing data is only available on the new version of the process and not on the existing active process instances based on the first version. The new process version does not affect the existing process instances and they can be completed normally however with a lack of audit data.

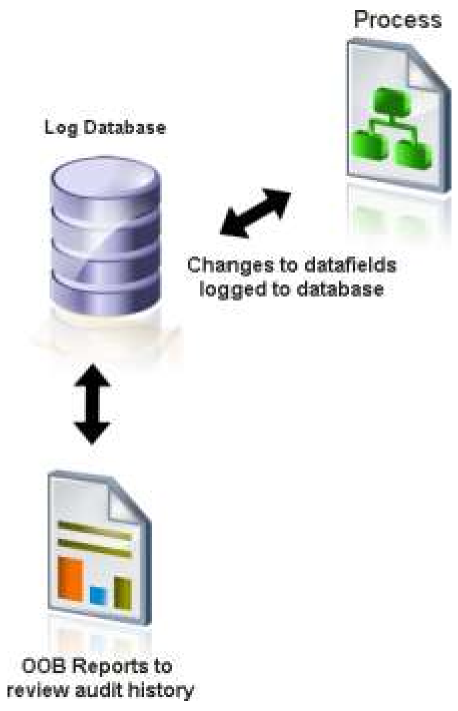

Process
Log Database
Changes to datafields logged to database
OOB Reports to review audit history

Figure 2 - Process Audit

SYSTEM LOGGING TYPES AND FORMATTING

There are several different types of runtime logging that can occur with K2 blackpearl.

INSTALLATION

When installing K2 blackpearl, the checkbox located on the welcome screen and the finish screen notifies the Administrator of the presence of logging data that provides details on the installation event. This instance of system logging is used primarily for error reporting and troubleshooting the system installation. The logging provides details of each installation event attempted or the results of attempting an event. For example, the Configuration Manager may attempt to write a registry entry and the attempt fails because the system does not have sufficient permissions to perform this action. That detail is recorded to the log file as an error for debugging purposes post installation.

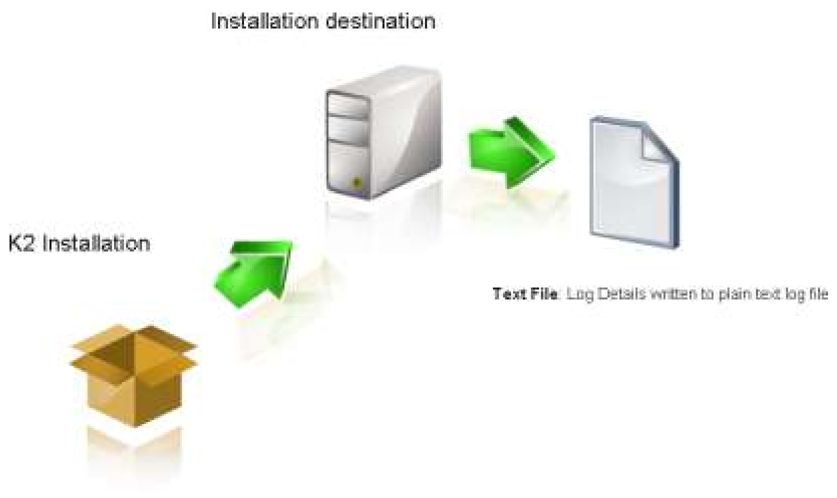


Figure 3 - Installation Logging

SYSTEM EVENT LOGS

For server-based operating systems, the system event log is used to view events within the context of the operating system that have occurred and caused an error, such as authentication and system errors system errors. However, not every item or event related to the internal running of K2 blackpearl would necessarily surface in the system event logs.

SERVER EVENTS AND LOGGING

The various servers and services installed with K2 blackpearl do, to varying extents, provide automated logging. This logging activity can be viewed in real-time for some servers, such as K2 server where as other server's logging results are viewed using the OOB Reports.

Real-time logging is available when the K2 blackpearl server is run in console mode, where each event that takes place is written to the console window. However, this form of logging is effective only as a means to troubleshoot events related to the startup and basic operation of the server. Since the K2 Server should be run as a service, once the system is restarted or the console window closed, the reporting is lost.

EVENT LOGGING CONFIGURATION

The event logging is dependent on the configuration setup in the K2 Server's configuration file. The settings in this file are read by the K2 Server on startup and the logging system operates according to these setting. When changes to the initial configuration are required, the configuration file can be updated but the K2 Server will require a restart for the changes to take effect. The configuration file is typically found at "C:\Program Files\K2 blackpearl\Host Server\Bin", as shown in the image below. Since it is a standard XML configuration file, you can use notepad or Visual Studio to edit the file.

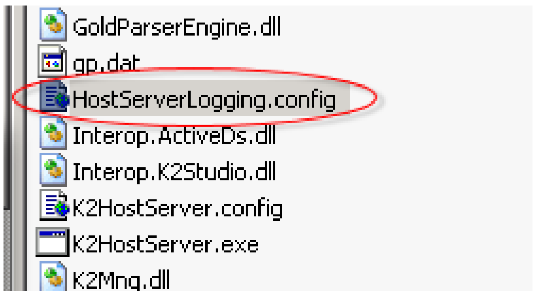


Figure 4 - Configuration File Location

EXTENSIONS

The Extensions element controls which methods of logging are used and the location where the logging information is written. The available extensions are:

- Console
- Event Log
- File
- MSMQ
- Archive

Custom logging extensions can also be written for K2 blackpearl. For more information and a sample project, including documentation, that illustrates how to do this, see the SourceCode.Logging Plug-in Home on the K2 blackmarket (http://k2underground.com/k2/ProjectHome.aspx?ProjectID=2).

In the following code sample, the available extensions are displayed:

```
<Extensions>
    <Extension Name="ConsoleExtension" type="SourceCode.Logging.Extension.ConsoleExtension">
      <Property Name="Shorthand" value="true"/>
    </Extension>
    <Extension Name="EventLogExtension" type="SourceCode.Logging.Extension.EventLogExtension" />
    <Extension Name="FileExtension" type="SourceCode.Logging.Extension.FileExtension">
      <Property Name="LogFileName" value="HostServer.log"/>
      <Property Name="LogFilePath" value="" />
      <Property Name="HashAlgorithm" value="CRC32" />
    </Extension>
    <Extension Name="MSMQExtension" type="SourceCode.Logging.Extension.MSMQExtension">
      <Property Name="QueuePath" value=".\private$\SCQueue"/>
    </Extension>
    <Extension Name="ArchiveExtension" type="SourceCode.Logging.Extension.ArchiveExtension">
            <Property Name="HostServerConfigFileName" value="K2HostServer.config"/>
      <Property Name="ConfigDBConnectionName" value="HostserverDB"/>
    </Extension>
</Extensions>
```

EVENT LOGGING

Events are only logged as they occur to the LogDatabase if the AsyncQueueEnabled attribute is set to false. The default setting is **True**, and this results in events not being logged as they occur. When the actual time of the event taking place is crucial as data, then this attribute should be set to false. However, although this may suit auditory requirements it can impact negatively on the K2 Server's performance since each event will require time to write to the database.

```
<appSettings>
    <!-- Set Date format-->
    <add key="DateFormat" value="yyyy-MM-dd hh:mm:ss" />
    <!-- Set base Log Level; should be "All"-->
    <add key="LogLevel" value="All" />
    <!-- Create Checksum/Hash for each row in logfile -->
    <add key="RowHash" value="False" />
    <!-- Adust Logger Thread Priority Low/Normal/High -->
    <add key="ThreadPriority" value="Low" />
    <!-- Enable/Disable Async Queueing True/False-->
    <add key="AsyncQueueEnabled" value="True" />
    <!-- Message Logged was an Exception, Include Stack Trace -->
    <add key="IncludeStackTrace" value="False" />
    <!-- Keep Log Files below this size. 0 = No Limit -->
    <add key="MaxFileSizeKB" value="0" />
    <!-- Keep Log Files below Timespan days:hours:minutes:seconds -->
    <add key="MaxFileTimeSpan" value="0:0:0:00" />
    <!-- Preserve Log Files SequenceNumber between restarts -->
    <add key="PreserveSequence" value="True" />
    <!-- Add GUID per log entry -->
    <add key="AddGUID" value="True" />
    <!-- Set MSMQ Path-->
    <add key ="MSMQActive" value="false"/>
    <add key="MSMQPath" value=".\private$\SCQueue"/>
  </appSettings>
```

LOGGING LEVEL

The level of detail that is captured is configured by setting or clearing two attributes per logging extension in the ApplicationLevelLogSettings element. The default logging level is **Debug** and this level will exclude some details from the messaging trace. For example, no license key information (set LogLevel attribute to ALL) will be listed in the K2 Server startup trace. This is a useful feature especially when you the product is under evaluation and details on the server expiration date are required.

Attributes	
Active	True / False: sets the Active attribute
LogLevel	The log level determines the type of detail that will be logged to the extension selected. *

*** See section below : "Message Severity Levels"**

```
<ApplicationLevelLogSettings>
    <ApplicationLevelLogSetting Scope="Default">
      <LogLocationSettings>
        <LogLocation Name="ConsoleExtension" Active="True" LogLevel="Info" />
        <LogLocation Name="FileExtension" Active="False" LogLevel="All" />
        <LogLocation Name="EventLogExtension" Active="False" LogLevel="Debug" />
        <LogLocation Name="ArchiveExtension" Active="False" LogLevel="Debug" />
        <LogLocation Name="MSMQExtension" Active="False" LogLevel="Debug" />
      </LogLocationSettings>
    </ApplicationLevelLogSetting>
</ApplicationLevelLogSettings>
```

LOGGING LEVEL SETTINGS

The logging level determines the type and volume of detail that will be captured. For example, setting the attribute to **All** would capture all detail and surface it via the extension selected. Selecting an alternate value includes certain items, and overrides others. The table below describes the attributes that can be selected and what their outcomes, benefits will be.

The Logging Level acts as a kind of filter depending on what is required by the user and the type of extension used. When designing a process, you may have your logging level set to **All** so that all messages can be viewed as they are written to the extension selected. During run time, the logging level would be set to a higher level for example **Info**. This would effectively filter out all messages with a priority level that is 2 or lower, surfacing only messages that are critical to system operation.

MESSAGING SEVERITY LEVELS

The Severity level is internal to the K2 blackpearl logging framework and is not extendible. The messaging object manages the incoming messages and depending on what the ApplicationLevelLogSetting attribute is set to, this will determine which messages are surfaced and logged to the extension or extensions that have been set to active. If you set the attribute to **Info**, then all messages from Info (example notification that a service has started) right through to error message will be displayed. However, if you set just for Errors then only errors will be surfaced and written to the selected extension.

Note: When writing to a database to log the data, setting the LogLevel attribute to **All** will result in a database that increases in size quite rapidly.

Attribute	Setting
All	Provides a full item trace on all events
Debug	Items that can assist with troubleshooting
Info	Non critical items, provides operational messaging for the Administrator to view system operation
Warning	Items that will not prevent the system from functioning. These may cause issues or errors in time
Error	Critical item where a service, resource has failed or is not reachable.

Note: In the table above, the severity of the error is form lowest to highest. Ie All is the lowest, with Error being the highest.

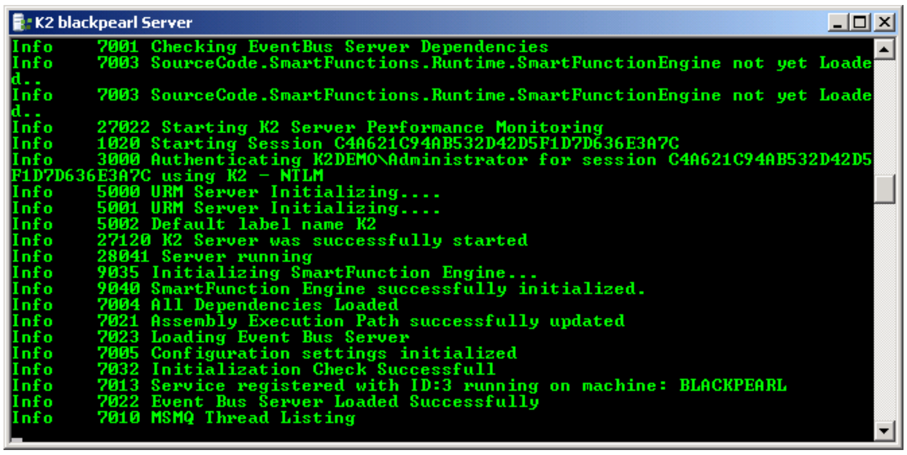
LOG TO CONSOLE

Using the console to view the logging details is real time and can be viewed in shorthand mode or this mode can be disabled to view the time stamp. Using the console does not record the trace and once the console more is closed, the trace is lost.

```
 <Extensions>
    <Extension Name="ConsoleExtension" type="SourceCode.Logging.Extension.ConsoleExtension">
      <Property Name="Shorthand" value="true"/>
</Extension>
```

In the code sample, the Shorthand attribute is set to true. To view the time stamp in line with the message trace, set the value to false.
The difference in display output is shown below. The second image displays the console output with the shorthand attribute set to false. Notably the output to screen is more verbose and a great deal more information is logged to the console; included with this is the time stamp for each event.

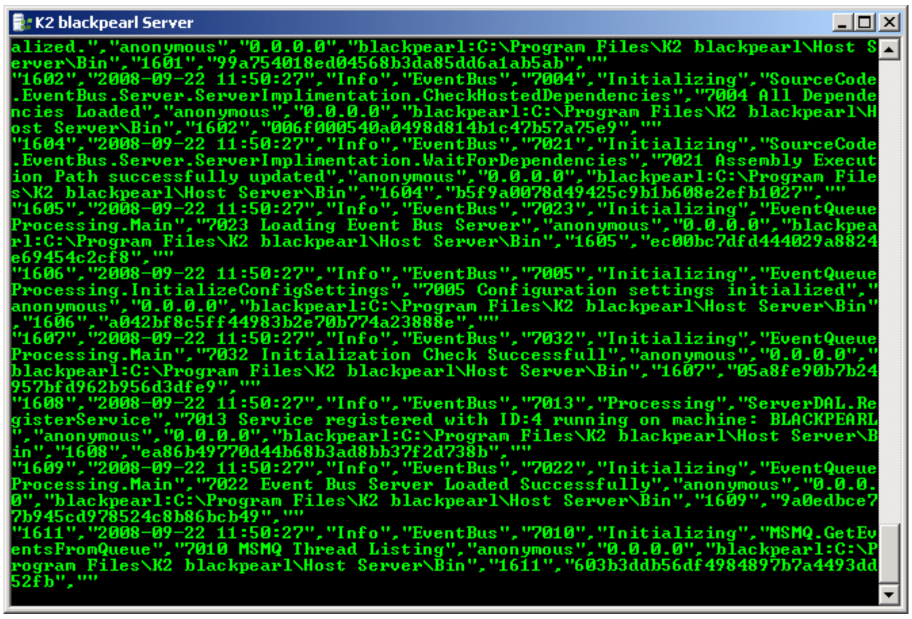
Console with the shorthand attribute set to true,



```
Info    7001 Checking EventBus Server Dependencies
Info    7003 SourceCode.SmartFunctions.Runtime.SmartFunctionEngine not yet Loade
d..
Info    7003 SourceCode.SmartFunctions.Runtime.SmartFunctionEngine not yet Loade
d..
Info    27022 Starting K2 Server Performance Monitoring
Info    1020 Starting Session C4A621C94AB532D42D5F1D7D636E3A7C
Info    3000 Authenticating K2DEMO\Administrator for session C4A621C94AB532D42D5
F1D7D636E3A7C using K2 - NTLM
Info    5000 URM Server Initializing....
Info    5001 URM Server Initializing....
Info    5002 Default label name K2
Info    27120 K2 Server was successfully started
Info    28041 Server running
Info    9035 Initializing SmartFunction Engine...
Info    9040 SmartFunction Engine successfully initialized.
Info    7004 All Dependencies Loaded
Info    7021 Assembly Execution Path successfully updated
Info    7023 Loading Event Bus Server
Info    7005 Configuration settings initialized
Info    7032 Initialization Check Successfull
Info    7013 Service registered with ID:3 running on machine: BLACKPEARL
Info    7022 Event Bus Server Loaded Successfully
Info    7010 MSMQ Thread Listing
```

Figure 5 - Console mode (Shorthand = "True")

Note the difference in the volume of detail with the console when the shorthand attribute is set to false, and the Loglevel value remains as **Info**.

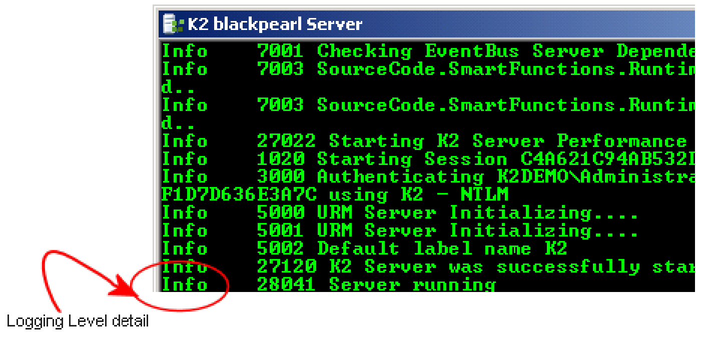


```
alized.","anonymous","0.0.0.0","blackpearl:C:\Program Files\K2 blackpearl\Host S
erver\Bin","1601","99a754018ed04568b3da85dd6a1ab5ab",""
"1602","2008-09-22 11:50:27","Info","EventBus","7004","Initializing","SourceCode
.EventBus.Server.ServerImplimentation.CheckHostedDependencies","7004 All Depende
ncies Loaded","anonymous","0.0.0.0","blackpearl:C:\Program Files\K2 blackpearl\H
ost Server\Bin","1602","006f000540a0498d814b1c47b57a75e9",""
"1604","2008-09-22 11:50:27","Info","EventBus","7021","Initializing","SourceCode
.EventBus.Server.ServerImplimentation.WaitForDependencies","7021 Assembly Execut
ion Path successfully updated","anonymous","0.0.0.0","blackpearl:C:\Program File
s\K2 blackpearl\Host Server\Bin","1604","b5f9a0078d49425c9b1b608e2efb1027",""
"1605","2008-09-22 11:50:27","Info","EventBus","7023","Initializing","EventQueue
Processing.Main","7023 Loading Event Bus Server","anonymous","0.0.0.0","blackpea
rl:C:\Program Files\K2 blackpearl\Host Server\Bin","1605","ec00bc7dfd444029a8824
e69454c2cf8",""
"1606","2008-09-22 11:50:27","Info","EventBus","7005","Initializing","EventQueue
Processing.InitializeConfigSettings","7005 Configuration settings initialized","
anonymous","0.0.0.0","blackpearl:C:\Program Files\K2 blackpearl\Host Server\Bin"
,"1606","a042bf8c5ff44983b2e70b774a23888e",""
"1607","2008-09-22 11:50:27","Info","EventBus","7032","Initializing","EventQueue
Processing.Main","7032 Initialization Check Successfull","anonymous","0.0.0.0","
blackpearl:C:\Program Files\K2 blackpearl\Host Server\Bin","1607","05a8fe90b7b24
957bfd962b956d3dfe9",""
"1608","2008-09-22 11:50:27","Info","EventBus","7013","Processing","ServerDAL.Re
gisterService","7013 Service registered with ID:4 running on machine: BLACKPEARL
","anonymous","0.0.0.0","blackpearl:C:\Program Files\K2 blackpearl\Host Server\B
in","1608","ea86b49770d44b68b3ad8bb37f2d738b",""
"1609","2008-09-22 11:50:27","Info","EventBus","7022","Initializing","EventQueue
Processing.Main","7022 Event Bus Server Loaded Successfully","anonymous","0.0.0.
0","blackpearl:C:\Program Files\K2 blackpearl\Host Server\Bin","1609","9a0edbce7
7b945cd978524c8b86bcb49",""
"1611","2008-09-22 11:50:27","Info","EventBus","7010","Initializing","MSMQ.GetEv
entsFromQueue","7010 MSMQ Thread Listing","anonymous","0.0.0.0","blackpearl:C:\P
rogram Files\K2 blackpearl\Host Server\Bin","1611","603b3ddb56df4984897b7a4493dd
52fb",""
```

Figure 6 - Console mode (Shorthand ="False")

Note: Console mode is for debugging only, and the K2 Server should not be run in this mode indefinitely. The detail logged to the console window is also lost once the console is closed and the K2 Server restarted.

When using the console as a logging extension, the message type is listed along with the message. For example, the image below shows the K2 server starting up. There is no context to the image, except to illustrate that the message type is **Info** and that detail is placed alongside the message so that this is evident to the person using the console. If the message were a Debug or Error type message the message would have Error or Debug printed next to it.



```
Info     7001 Checking EventBus Server Depende
Info     7003 SourceCode.SmartFunctions.Runtim
d..
Info     7003 SourceCode.SmartFunctions.Runtim
d..
Info     27022 Starting K2 Server Performance
Info     1020 Starting Session C4A621C94AB532I
Info     3000 Authenticating K2DEMO\Administra
F1D7D636E3A7C using K2 - NTLM
Info     5000 URM Server Initializing....
Info     5001 URM Server Initializing....
Info     5002 Default label name K2
Info     27120 K2 Server was successfully star
Info     28041 Server running
```

Figure 7 - Console Logging Message Types

LOGARCHIVE TABLE

The Archive extension logs data to the LogArchive table. As can be seen in the image below a number of fields are captured. This database table is open and can be accessed either manually using the SQL Server Management Studio or through a SmartObject using the SQL SmartObject Service.

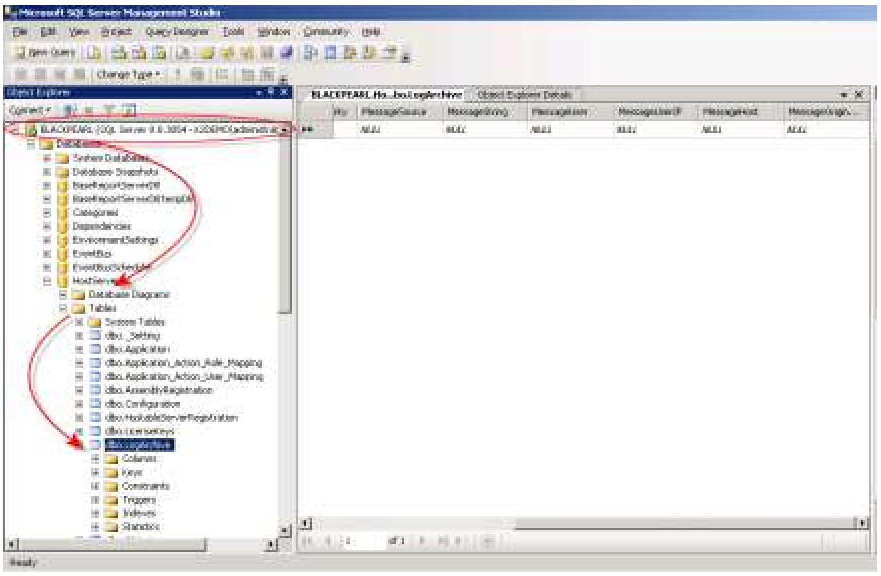


Figure 8 - LogArchive Table

The following table provides information about each field in the LogArchive table.

Table Column	Description
DateTime MessageTimestamp	The MessageTimestamp is generated by the logging framework as the message is being logged.
Int64 MessageID	The MessageID is provided by the calling code. It is the ID of the message as defined inside of HostServerLogging.config.
string MessageCategory **string MessageName** **Severity MessageSeverity** **string MessageString**	These four items correspond to the elements in the HostServerLogging.config file and are based off of the MessageID. The MessageString may contain parameters provided by the caller.
string MessageSource	MessageSource is provided by the caller and indicates the area of code that the message originates from. This should typically be the namespace and class.method of the calling code.
MessageIdentityObject MessageIdentity	MessageIdentity contains two items; MessageUser and MessageUserIP. This should be provided by the calling code when available. MessageUser would be the user who is currently logged on and MessageUserIP would be his or her IP Address.
string MessageHost	MessageHost is set by the logging framework and is used to indicate the server name of the originating HostServer. At this time, this is only useful when using MSMQ.
UInt64 MessageOriginalSeqNum	MessageOriginalSeqNum is also only used when using MSMQ. It shows the messages original sequence number from where the message originated from.

ANALYZING THE LOG

When the extension to write to a log file is enabled, the data is written to a plain text file in a comma delimited format. If the data is required for processing, analysis or reports generation the text file can be read back into an application that reformats the data, such as Microsoft Excel.

LOGGING AND AUDITING WITH A PROCESS

Various aspects of the process are logged to the Log Database which provides a detailed volume of information. The areas that are logged are:

- Activity Instances
    - Activity Instance Data Audit
    - Activity Instance Destination Data
    - Activity Instance Destination XML
    - Activity Instance Slot Data
    - Activity Instance Slot XML
    - Activity Instance XML
- Process Instance
    - Process Instance Data
    - Process Instance XML

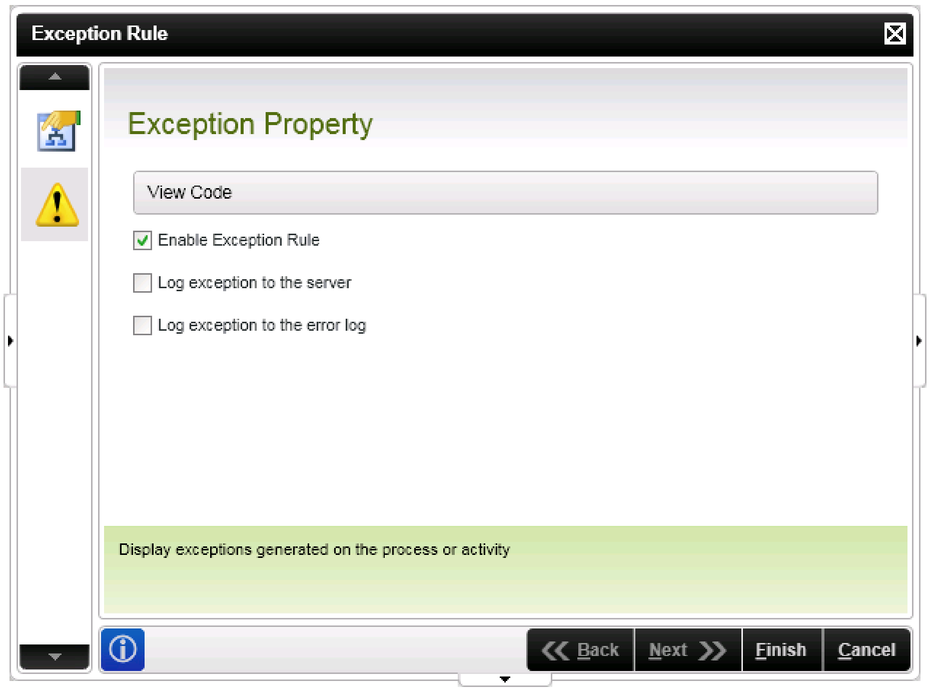
The data capture is stored in tables, which enables the SmartObject service broker to be used to access the data using K2 SmartObjects.

PROCESS EVENTS

The K2 workflow server, which is hosted within the K2 Server service, automatically logs each event that it performs. This cannot be disabled and is part of the system's functionality. The level of logging does not include all available elements that can be included. The service's configuration file is used to configure the levels of logging and logging sources. When selecting sources of data to log; the data is stored in the database which will grow in size very quickly. This may have a negative impact on the performance of the SQL server since the increase in database size will impact data access times. It is recommended that full logging detail is only turned on for a short period of time to diagnose a problem with the workflow server. After the problem has been identified, logging levels should be reset to their original values for all sources.

K2 SERVER EVENTS

From within a process, K2 Server Events can be used to populate entries into both the Error Log and the Server Log. To configure the exception captures, the exception properties are set from the wizard. The exception rule is first enabled and thereafter a selection made to write entries to the Log and Server databases as shown in the following figure.



☑ Enable Exception Rule
☐ Log exception to the server
☐ Log exception to the error log

Display exceptions generated on the process or activity

Figure 9 - Exception Property Settings

The exception rule will only function if enabled; therefore no logging will take place unless the options displayed in the dialog are enabled. The **Enable Exception Rule** option must be enabled for the Exception Rule to run.

Custom code can be added to enhance the exception rule. From here, the process designer can simply finish the wizard or the code can be viewed and modified by selecting **View Code**. A sample of this code is shown below.

```
#region - Properties_ExecuteCode -
        private void Properties_ExecuteCode(object sender, EventArgs e)

          {
            K2.AddToErrorLog = K2.Configuration.IsErrorLog;

            K2.AddToServerLog = K2.Configuration.IsServerLog;

        }

#endregion
```

The results of the above code are written to two separate log files which are plain text files located in the …Host Server\bin directory. The entries depend on the events raised.

The advantage of using custom code is that custom actions can be implemented to enhance the error or event logging. For example, writing the same errors to a database table, this makes it far easier and simpler to generated meaningful reports on the captured data.

The code sample below illustrates how exception data can be written to a database table.

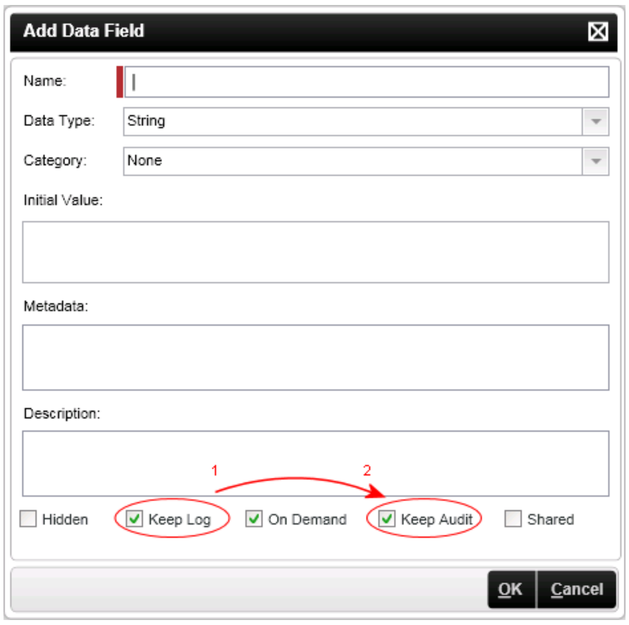
```
SqlCommand sqlCommand = new SqlCommand();
SqlConnection sqlConnection = null;
sqlConnection = new SqlConnection("MyConnectionString");
sqlConnection.Open();
sqlCommand.Connection = sqlConnection;
sqlCommand.CommandType = System.Data.CommandType.StoredProcedure;
sqlCommand.CommandText = storedProcedure;
sqlCommand.Parameters.Clear();
sqlCommand.Parameters.Add("@Exception", "My Exception Message.");
int result = sqlCommand.ExecuteNonQuery("spInsertErrorEntry", sqlCommand.Parameters);
sqlConnection.Close();
sqlConnection.Dispose();
sqlCommand.Dispose();
```

A database with one table must be created first. It is not recommended that the existing K2 databases be used as this is not supported under the K2 blackpearl license agreement.

Once these details have been written to the database, they can be retrieved as a K2 SmartObject and reports generated from the data displayed using forms.

HOW TO CONFIGURE DATA FIELDS FOR USE IN AUDITING

The information discussed here must be implemented at design time.



Data Type: String
Category: None
Initial Value:
Metadata:
☐ Hidden ☑ Keep Log ☑ On Demand ☑ Keep Audit ☐ Shared

Audit data is viewed using OOB and custom reports -- using the data field in a report will be discussed in the next section. This section covers how to setup data to be audited. To add auditing capabilities to a data field two of the check boxes at the bottom of the Add Data Field dialog must be enabled. As shown in the image below, the **Keep Log** and **Keep Audit** options are both enabled, and it is important to note that they should be enabled in the order shown here.

When the **Keep Log** option is enabled, data is written to and stored within the K2Log Database. The OOB reports will later generate audit reporting information based on the data available from with this database.

The **Keep Audit** option is dependent on the **Keep Log** option; unless **Keep Log** is enabled the **Keep**

Figure 10 - Auditing on a Data Field

Audit feature is neither available nor can it be enabled.

SYSTEM AUDIT USING THE STANDARD REPORTS

Note: The system audit features must be enabled first before auditing will take place.

The system audit feature monitors and displays the results of changes to data fields, both at the activity and process levels. What this means is that when changes are made, that the change is auditable. An audit report is generated when changes take place reflecting the previous state of a data field and its current state, and the state of the data field when the process is complete.

AUDITABLE CHANGES

Actions that the system will audit are changes to the state of the data fields. This includes the following

- From null value to a value
- From value to null
- For text, when a change in case occurs in any of the characters

USING THE STANDARD REPORTS

To demonstrate the audit trail functioning a quick process is outlined below. The process is a simple Approve-Decline, and when the request is approved a data field is updated with the appropriate details.

Note: A lengthy discussion of setting up and creating the process will not be illustrated at this point. For more information about how to design this type of process, see the K2 User Guide.

The simple process is outlined below, graphically. And ultimately, auditing can be performed on the simplest to complex processes, so long as the succeeding rule is executed successfully.

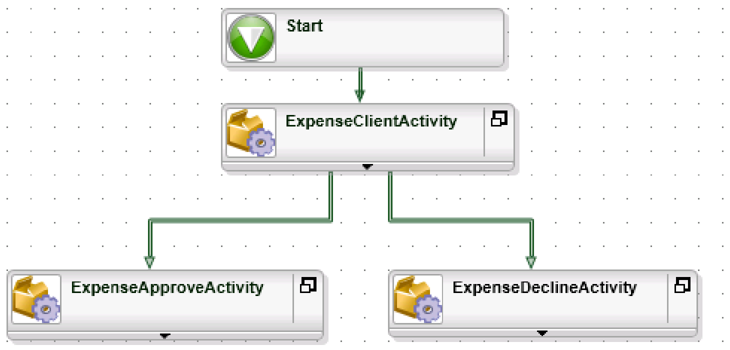
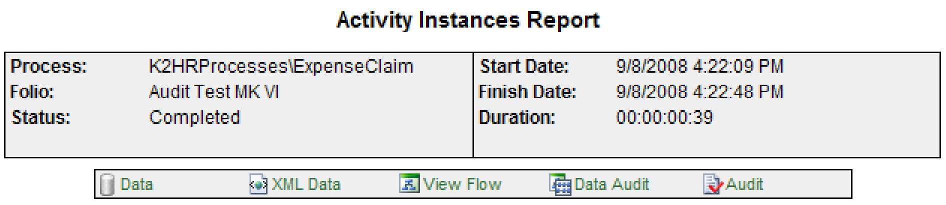


Figure 11 - Simple Process Example

ACTIVITY INSTANCE REPORT

From the report below, the analyst performing an audit of a process has access to a variety of information sources, including Data, View Flow and Audit. The features that are of interest to an Auditor are the **Data Audit** and **Audit.**



Activity Instances Report

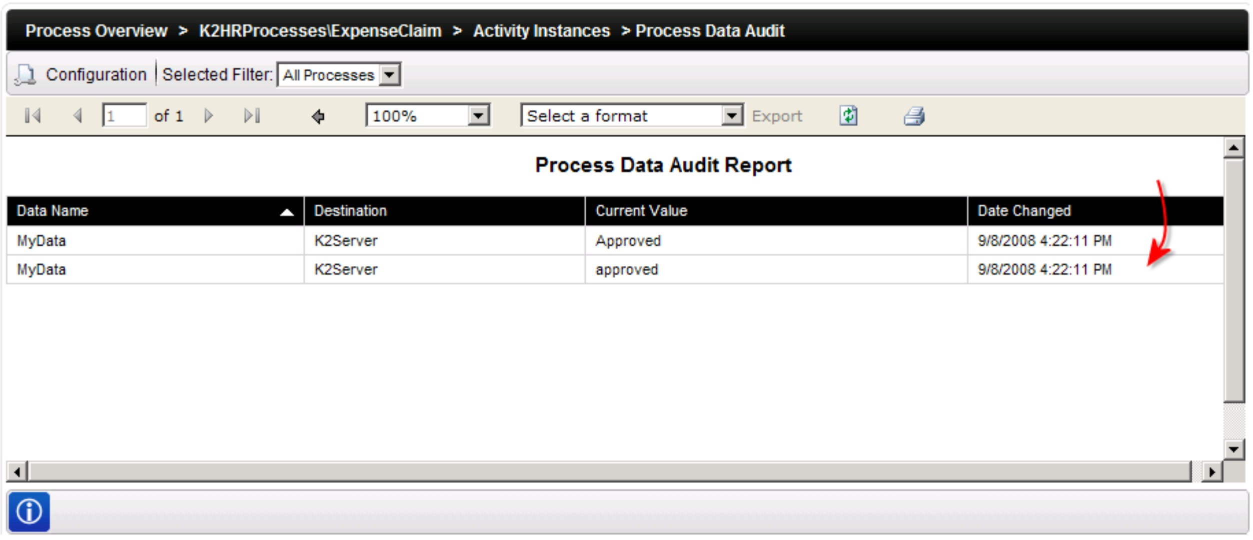
Process:	K2HRProcesses\ExpenseClaim	**Start Date:**	9/8/2008 4:22:09 PM
Folio:	Audit Test MK VI	**Finish Date:**	9/8/2008 4:22:48 PM
Status:	Completed	**Duration:**	00:00:00:39

Figure 12 - Activity Instance Report

Note: The other options such as Data and XML allow the Auditor to view the changes to process as the events were logged.

Data Audit Report

Clicking **Data Audit** generates the Process Data Audit Report. As can be seen below, the changes from the value “Approved” to “approved” are listed, which provides a clear audit trail of what happened in the process. Time based validation is provided as well as shown in the Date Changed column. This report is useful for determining the audit trail on data within a process and when changes occurred. The changes implemented in this example were performed by a single server event.


Process Overview > K2HRProcesses\ExpenseClaim > Activity Instances > Process Data Audit
Configuration
Selected Filter: All Processes
1 of 1
100%
Select a format
Export

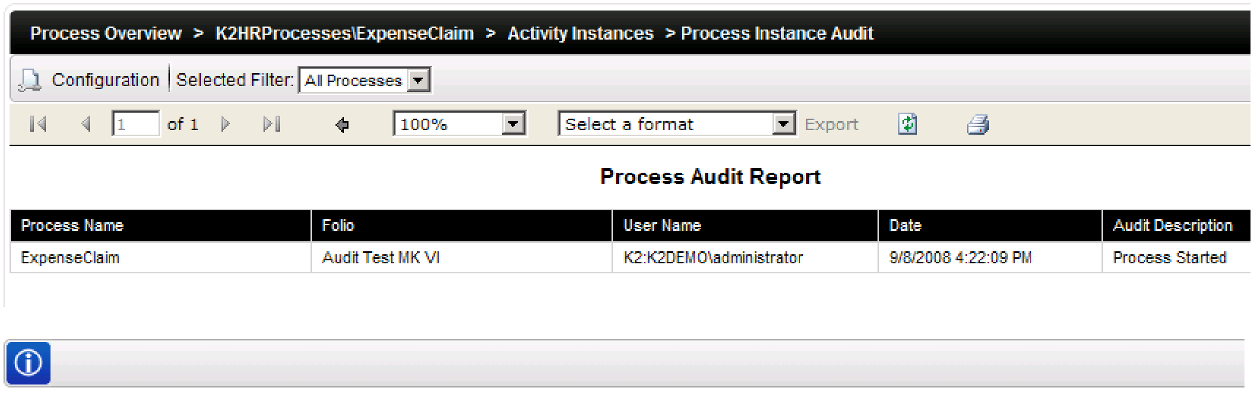
Process Data Audit Report

Data Name	Destination	Current Value	Date Changed
MyData	K2Server	Approved	9/8/2008 4:22:11 PM
MyData	K2Server	approved	9/8/2008 4:22:11 PM

Figure 13 - Process Data Audit Report

Process Audit Report

The Process Audit Report provides details on the activities of a process, for example which users actioned the process and what the outcome of the actions were. A full description is provided per action, according to process name, Folio, User Name, Date and an Audit Description. This report is useful for determining what happened during each process activity and when the change occurred. From an auditing perspective, this level of reporting provides an in-depth look at the history and life cycle of the process along with the human interaction with the process.


Process Overview > K2HRProcesses\ExpenseClaim > Activity Instances > Process Instance Audit
Configuration
Selected Filter: All Processes
1 of 1
100%
Select a format
Export

Process Audit Report

Process Name	Folio	User Name	Date	Audit Description
ExpenseClaim	Audit Test MK VI	K2:K2DEMO\administrator	9/8/2008 4:22:09 PM	Process Started

Figure 14 - Process Audit Report

Process Audit Data Availability

The data captured during the system's operation, that is stored within the K2 database will be available indefinitely. The audit data may not be available under the following conditions

1. **Process Deleted**

   When a process is deleted all the data and history can be deleted as well. When this is done the audit data for this process will be lost. Content written to a text file however will still be available

2. **Archiving**

   From K2 workspace completed processes can be archived and this will result in their data and history being archived as well. The data is however not lost and is accessible once restored to the K2LogDatabase

USING THE SERVICE METHODS FOR CUSTOM REPORTS

The data that is generated when the process is active is captured in a number of database tables (see next section for a list) and the information can be accessed using the service methods. A custom solution would need to be developed that leverages the List service method. This would require the creation of a SmartObject with the List Service method returning the data and populating a List View to display the data.

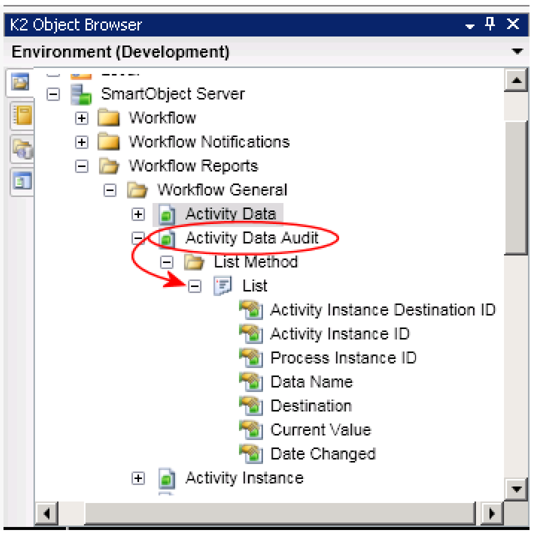


Figure 15 - K2 Object Browser

CUSTOM SOLUTIONS

The K2 blackpearl platform will allow a developer to access the resources that it uses to produce the standard reports. The Audit data is stored in one of the K2 databases the K2 ServerLog database. Solutions may be developed by the clients and customers referencing the data tables. If a custom solution is developed, this would not be supported in terms of the EULA. By custom, in this context reference is made to a solution which directly references the database and its tables. The list of database tables is below.

- ActInstAudit
- ActInstDataAudit
- ActInstDestDataAudit
- AstInstDestXMLAudit
- ActInstSlotDataAudit
- ActInstSlotXMLAudit
- ActInstXMLAudit
- ProcInstAudit
- ProcInstDataAudit
- ProcInstXMLAudit

CONCLUSION

The Auditing and Logging framework offers an extensible means of monitoring and managing the operation of the K2 blackpearl environment from installation to run time.

Leveraging these features does require some initial configuration and setup. Most of the logging framework is operational automatically and cannot be disabled, but the level of detail logged can be controlled. The Auditing framework, however, must be activated manually during process design and once deployed can provide the audit functions required by a company's policies or third-party Audit.