Events and Audit Architecture

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Description** |
| 1.7 | 6/3/2013 | Slava Minyailov | First version |
| 1.7 | 6/5/2013 | Slava Minyailov | Added a section on event collectors |
| 1.7 | 7/2/2013 | Slava Minyailov | Changed column names and default parameters of the events filter. Added a section about the description column. |
| 1.7 | 7/3/2013 | Michael Lampi | Minor edits and spelling changes |

**Document Approvers & Sign-Off**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Approver** | **Role** | **Document Accept/Reject** |
|  |  |  |  |

Contents

[1 Introduction 4](#_Toc362860713)

[2 System Design 4](#_Toc362860714)

[2.1 Components 4](#_Toc362860715)

[2.1.1 xBRMS 4](#_Toc362860716)

[2.1.2 xBRMS UI 5](#_Toc362860717)

[2.1.3 XGreeter App, IDMS, JmsListener 5](#_Toc362860718)

[2.1.4 xBRC 5](#_Toc362860719)

[2.1.5 xBRC UI 5](#_Toc362860720)

[2.2 Authentication and Authorization 5](#_Toc362860721)

[2.3 Event Categories 6](#_Toc362860722)

[2.4 Events Message Format 7](#_Toc362860723)

[2.5 Description Column 8](#_Toc362860724)

[2.6 Events Collection and Aggregation 9](#_Toc362860725)

[2.7 Cleanup Procedure 10](#_Toc362860726)

[2.8 Event Interceptors 10](#_Toc362860727)

[2.9 Configuration properties 11](#_Toc362860728)

# Introduction

Create an enterprise database of security and system health events for all xConnect components. The database will enable the maintenance personal to examine functionality of the system over a period of time and find answers to such questions as who did what and when, as well as what happened to the system and why. It will be possible to query the database either directly or via specialized reporting tools that can be developed over time.

Besides providing the historical data the system will be able to take real-time actions in response to critical systems events. For example, it can send an SMS or email to an administrator when a certain event occurs.

# System Design

## Components

The following xConnect components are producers of the audit events:

* xBRMS
* xBRMSUI
* xBRC
* xBRC UI
* xGreeter App (xBRMS)
* IDMS (via xBRMS)
* JmsListener (via xBRMS)

### xBRMS

Global xBRMS is the owner of the enterprise database. No matter in what place in the system an event occurred it will ultimately appear this database. The global xBRMS periodically collects events from park xBRMSs, which in turn pull them from xBRCs.

In addition to the describe collection model, any xConnect component can push events to either park or global xBRMS directly via “PUT@XBRMS/rest/audit/push” endpoint. Currently, only xBRMS UI uses this method.

xBRMS itself can generate audit data on:

* User logins, logouts, secure token requests and token invalidations (Greeter App)
* Initial health status of monitored components (xBRCs, IDMS, JmsListener, Park xBRMSs)
* Change in health status of monitored components (xBRCs, IDMS, JmsListener, Park xBRMSs)
* Access to all protected endpoints.

### xBRMS UI

Since xBRMS UI does not have its own database for events caching, it pushes generated events to the global xBRMS. xBRMS UI generates audit data on users' logins and logouts.

### XGreeter App, IDMS, JmsListener

Currently, these components neither push nor cache audit events for later collection by xBRMS. xBRMS itself generates events on their behalf in the following situations:

* xGreeter App requests a secure token for user authentication from xBRMS
* xBRMS detects a change in health status of one of the monitored systems (IDMS, JmsListener, or xBRC). In this case, xBRMS generates an event as if it was originated on the monitored system.

### xBRC

xBRC caches events in the Mayhem database. It generates audit data on:

* Authenticated or anonymous access
* Read, write, update and delete operations in the Mayhem database

### xBRC UI

Writes directly to the Mayhem database. Generates audit data on user logins and logouts.

## Authentication and Authorization

User access to all protected xConnect components requires authentication. The authentication process is performed by the xConnect Authentication Gateway (xAG), which delegates authentication operations to a Keystone server. As a result of successful authentication, a secure token is created. The token includes a list of roles and functional abilities granted to the user.

When one xConnect component calls another, it authenticates to it using HTTP Basic Authentication and service account credentials. If the call is made on behalf of a user, it additionally passes the user’s secure token, thus effectively impersonating the user. Besides the roles and functional abilities, the token contains a user session ID, which once set never changes until the user logs out or the user’s HTTP session expires. As a result, if all xConnect components are configured for SSO, i.e., use the same xAG server, all audit events generated by different systems in response to a user action will be marked with the same user name and session ID.

xBRC authentication mechanism is different from the one used by the servlet-based web applications like xBRMS, xBRMS UI or xBRC UI. In xBRC:

* “Fast” endpoints used by the readers are fully excluded from the authentication and audit activity
* Some endpoints used by systems external to xConnect do not require authentication.

All xBRC endpoints are described in xbrc-resources.xml file. xBRC looks for this file in /etc/nge/config directory. If the file is not present xBRC checks the default location /user/share/xbrc. The file entries look like this:

<resource method="GET" path="/currentconfiguration">

<audit>true</audit>

<authenticate>true</authenticate>

<ssl>false</ssl>

</resource>

The file makes it possible to enable/disable audit or authentication on a single endpoint or on a set of endpoints. Note that SSL attribute is not currently used.

## Event Categories

All events belong to one of two classes: *audit* or *health*.

The primary focus of the audit events is accountability. The audit events are marked with the user and session IDs. The latter can be used to correlate user-initiated operations spanning multiple systems.

*Health* events carry data related to the system operational status. They are not associated with real users and do not have session ID.

The following table summarizes the events classification:

|  |  |  |
| --- | --- | --- |
| **Event Type** | **Event Category** | **Description** |
| AUDIT\_SUCCESS | AUDIT\_FAILURE | LOGIN | User logon event |
| AUDIT\_SUCCESS |AUDIT\_FAILURE | LOGOUT | User logout event. |
| AUDIT\_SUCCESS | AUDIT\_FAILURE | READ | Information retrieval event generated by a data provider, e.g., xBRC. |
| AUDIT\_SUCCESS | AUDIT\_FAILURE | WRITE | Information creation or modification event generated by a data provider. |
| AUDIT\_SUCCESS | AUDIT\_FAILURE | ACCESS | Data access event typically generated by a front end system, e.g., a RESTful web service. On success, it may be followed by READ or WRITE events generated by a data provider. |
| FATAL | [Application specific] | Denote a total system failure. Not currently used. |
| ERROR | [Application specific] | Error condition, which may degrade system functionality or even result in total system failure. |
| WARN | [Application specific] | Unexpected or abnormal condition, which may or may not degrade system functionality. |
| INFO | [Application specific] | Informational event. |
| ERROR | WARN | INFO | STATUS | Initial health status of a monitored system. xBRMS generate this event on startup. |
| ERROR | WARN | INFO | STATUS\_CHANGE | Change in health status of a monitored system. |

## Events Message Format

All events include data to uniquely identify:

* Event source (application class, instance ID, etc.)
* Event date and time
* User (audit events only)
* User session (audit events only)

The audit WRITE events additionally contain details of a change request. The description includes a unique property ID, and a new property value. The property ID is similar to a REST resource ID. The following table specifies the detailed format of an event message.

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| id | BIGINT | Private event ID local to the system owning the events cache. |
| aggregationId | BIGINT | Aggregation ID. This ID gets assigned to an event by the event aggregator as it pulls data from a child node. This ID equals the private ID of the event on the child node. |
| eventType | VARCHAR(16) | There are six event types: AUDIT\_SUCCESS, AUDIT\_FAILURE, FATAL, ERROR, WARN, INFO |
| eventCategory | VARCHAR(32) | For audit events is one of the following: READ, WRITE, ACCESS, LOGIN, or LOGOUT. For health events the category is application specific. |
| DateTime | DATETIME | Data and time of the event. |
| applicationClass | VARCHAT(16) | Unique application name, e.g. xBRMS, xBRMSUI, xBRC, xBRCUI, xGreeterApp, xi, IDMS, etc. |
| applicationId | INT | Application instance ID, e.g., park ID for xBRMS or venue ID for xBRC |
| sourceAddress | VARCHAR(128) | Host name/IP address of the event source. |
| sourceVirtualAddress | VARCHAR(128) | Host name/VIP address of the event source. |
| userId | VARCHAR(32) | User ID (Audit events only) |
| userSessionId | VARCHAR(32) | HTTP session ID (Audit events only) |
| description | VARCHAR(256) | Optional description. |
| resourceId | VARCHAR(128) | Optional resource ID (Property path for audit WRITE events) |
| resourceData | VARCHAR(max) | Optional type/specific data, e.g., new property value for audit WRITE events, or error message for “ERROR-STATUS\_CHANGE” events. |
| clientAddress | VARCHAR(256) | Host name or IP address of the client that made the request. |
| collectedFromXbrmsAt | VARCHAR(256) | Address of the event collector (typically park xBRMS) from where the event was collected. |

## Description Column

*Descriptio*n is a special column for a user-friendly message about an event. The text of the column is processed using a template that contains placeholders for other fields in the event. Each combination of “<eventType>/[eventCategory]” can have a separate template specified in the environment.properties file via *com.disney.xband.common.lib.audit.desc.template.[0-20]* property. By default the *description* field is processed using the following templates. Note that *%<column name>* is a placeholder for an event field from the above table and *<eventType>/[eventCategory]* is a pattern that tells what events the template applies to:

1. AUDIT\_SUCCESS/ACCESS: User %userId got access to end-point %resourceId
2. AUDIT\_FAILURE/ACCESS: User %userId failed to get access to end-point %resourceId
3. AUDIT\_SUCCESS/LOGIN: User %userId was successfully authenticated by xAG on host %sourceAddress
4. AUDIT\_FAILURE/LOGIN: User %userId failed to authenticate to xAG on host %sourceAddress
5. AUDIT\_SUCCESS/LOGOUT: User %userId successfully logged out from xConnect on host %sourceAddress
6. AUDIT\_SUCCESS/WRITE: User %userId modified the following %applicationClass property on host %sourceAddress: %resourceData
7. AUDIT\_FAILURE/WRITE: User %userId failed to modify an %applicationClass property on host %sourceAddress
8. ERROR/STATUS: Initial status of %applicationClass on host %sourceAddress is %message
9. WARN/STATUS: Initial status of %applicationClass on host %sourceAddress is %message
10. INFO/STATUS: Initial status of %applicationClass on host %sourceAddress is %message
11. ERROR/STATUS\_CHANGE: Status of %applicationClass on host %sourceAddress changed from %message
12. WARN/STATUS\_CHANGE: Status of %applicationId on host %sourceAddress changed from %message
13. INFO/STATUS\_CHANGE: Status of %applicationId on host %sourceAddress changed from %message
14. ERROR/: %applicationId on host %sourceId reported an error: %message
15. WARN/: %applicationId on host %sourceId issued a warning: %message
16. INFO/: %applicationId on host %sourceId said: %message

## Events Collection and Aggregation

Events collection and aggregation mechanism is hierarchical.

Initially generated events are cached in a private storage of an event provider, e.g., MySQL database in the case of xBRC or MS SQL Server database in case of xBRMS.

Event collectors (xBRMSs) periodically pull the stored events from the managed elements and write them in their own databases. Upon successful retrieval the event collectors delete the collected events on the remote system up to the last successfully saved event.

The global xBRMS periodically pulls the events from the park xBRMSs.

xConnect components that do not have a local storage, e.g,. xBRMS UI, or do not participate in the management hierarchy, e.g., xi, can publish events to the event collectors directly.

The following picture illustrates the events collection and aggregation architecture.



## Cleanup Procedure

Normally, audit events are automatically deleted from the event providers on successful retrieval by the event collectors. If for some reason no event collector pulls events from an event provider, e.g., because the provider is not being monitored by any collector, the cleanup procedure may kick in when the number of cached events exceeds an allowed maximum. The cleanup procedure checks the number of cached events every 3 hours. If the number exceeds a limit (10000 by default), 1/3 of the oldest records get deleted automatically. Such a cleanup procedure is currently used by xBRCs and park xBRMSs.

The global xBRMS uses a different cleanup procedure which does not have a limit on the number of stored events. Instead, it automatically deletes records that have been kept in the enterprise database for more than a certain number of days (14 by default).

## Event Interceptors

Events interceptors can make xConnect components take immediate action when an audit event flowing through them satisfies certain criteria. One of the possible actions is deleting events from the normal flow. In this case, the interceptor acts as a filter.

Currently, there is only one event filter implementation - com.disney.xband.common.lib.audit.interceptors.SimpleFilter

This filter is registered during audit system initialization. It can be disabled or its default parameters changed through the environment.properties file.

The filter reduces the amount of “noise” in the audit database, e.g., the activity of the event collectors themselves. It takes parameters in the following format (BNF):

Params := Param | Param “,” Params

Param := SimpleParam | ComplexParam

SimpleParam := <Resource ID regular expression> // Filter out events with the matching “rid” field

ComplexParam := EventTypeSpec “:” EventCategorySpec “:” UserIdSpec “:” Interval // Filter out events for the same resource within a specified interval in seconds

EventTypeSpec := EventType | “\*”

EventCategorySpec := EventCategory | “\*”

UserIdSpec := UserId | “\*”

Interval := INTEGER // if Interval <= 0 events frequency will be ignored

On xBRC the filter uses the following default parameters:

“AUDIT\_SUCCESS:ACCESS::,

AUDIT\_FAILURE:ACCESS::120,

AUDIT\_SUCCESS:ACCESS:xconnect-service:,

AUDIT\_SUCCESS:READ:xconnect-service:,

AUDIT\_SUCCESS:WRITE:xconnect-service:”

On xBRMS the parameters default to:

“AUDIT\_SUCCESS:ACCESS::,

AUDIT\_FAILURE:ACCESS::120,

AUDIT\_SUCCESS:ACCESS:xconnect-service:”

## Configuration properties

Most of the configuration properties used by the audit subsystem on xBRCs can be changed either through the xBRC UI or directly in the Mayhem database.

The same audit properties can be changed on xBRMS systems through the xBRMS database.

// Is Audit enabled?

@PersistName("isEnabled")

// Maximum number of records to keep in the events cache. On reaching this number 1/3 of the records

// get automatically deleted. This setting does not apply to the enterprise (global) database.

@PersistName("keepInCacheEventsMax")

// Maximum number of days to keep audit events in the enterprise (global) database.

@PersistName("keepInGlobalDbDaysMax")

// How often event collectors pull events from event providers.

@PersistName("pullIntervalSecs")

// See AuditEvent.Type (AUDIT\_FAILURE, AUDIT\_SUCCESS, FATAL, ERROR, WARN, INFO)

@PersistName("level")

For the convenience of developers and testers the following properties can also be specified in the environment.properties file, in which case they override those in xBRMS database:

nge.xconnect.audit.enabled

nge.xconnect.audit.keepInCacheEventsMax

nge.xconnect.audit.keepInGlobalDbDaysMax

nge.xconnect.audit.pullIntervalSecs

nge.xconnect.audit.level

There are also a few properties that can be changed only through the environment.properties file:

nge.xconnect.audit.batchSizeMax // How many events try to pull in a single collect operation

com.disney.xband.common.lib.audit.interceptor.class.[0-9] // Full name of the event interceptor classes

com.disney.xband.common.lib.audit.interceptor.params.[0-9] // Corresponding parameters to the event interceptors

Note that if at least one event interceptor is specified in the environment.properties file the default interceptor will not be registered on startup.

## 