xBRMS Test Plan

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Description** |
| 1.0 | 5/1/12 | Timothy Sharpe | Updated to match GXP Test and Survey functionality |
| 1.5 | 1/16/13 | Michael Lampi | Added testing for xTable 1.1 readers |
| 1.7 | 7/10/13 | Francisco Gonzalez | Updated to include Events and Audit System |

**Document Approvers & Sign-Off**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Approver** | **Role** | **Document Accept/Reject** |
| 5/1/12 | Mark Mecham | Synapse - QA Manager | Accept |
|  |  |  |  |

**Open Issues**

|  |  |  |  |
| --- | --- | --- | --- |
| **Priority** | **Due date** | **Owner** | **Issue** |
|  |  |  |  |

**Closed Issues**

|  |  |  |  |
| --- | --- | --- | --- |
| **Priority** | **Due date** | **Owner** | **Issue** |
| 1 | 4/10/12 | Arek | NO – the facilities API endpoint will not use the same basic authentication as the xBRMS UI. |
| 1 | 4/8/12 | Arek | demouser|demouser – the username and password for the xBRMS pages to edit the xBRC properties |

**Table of Contents**

[1 Introduction 4](#_Toc361229523)

[1.1 Purpose 4](#_Toc361229524)

[1.2 Reference Documentation 4](#_Toc361229525)

[1.3 Related Persons 4](#_Toc361229526)

[2 High Level Use Cases 5](#_Toc361229527)

[3 Scope 6](#_Toc361229528)

[4 External Dependencies 6](#_Toc361229529)

[5 Exit Criteria 7](#_Toc361229530)

[6 Performance Targets 7](#_Toc361229531)

[7 Test Strategies 7](#_Toc361229532)

[7.1 Core Strategies 7](#_Toc361229533)

[7.2 JMS Messages 7](#_Toc361229534)

[7.3 xBRC / IDMS 9](#_Toc361229535)

[7.4 xi 9](#_Toc361229536)

[7.5 Reader 9](#_Toc361229537)

[7.6 API 10](#_Toc361229538)

[7.7 UI 10](#_Toc361229539)

[7.8 Database 11](#_Toc361229540)

[7.9 Events and Audit Subsystem 11](#_Toc361229541)

[7.10 Test Strategy Details 12](#_Toc361229542)

[8 Data Needs 13](#_Toc361229543)

[9 Validators 13](#_Toc361229544)

[10 Test Tools 14](#_Toc361229545)

# Introduction

## Purpose

The purpose of this document is to describe how the xBRMS component will be tested. The xBRMS will be tested in an integrated manner in an environment with both input readers and external ticketing components. These components can be turned on and off as needed by the tests.

These components consist of UI, JMSListener, XI and API aspects. The UI and API aspects will be tested as standard internet protocols using both manual tests and automation based on Selenium and node.js. The JMSListener aspect will be tested by observing the results within the xBRMS database and by scanning the log files for exceptions. The XI aspect reads from xBRMS and visualizes both guest and entitlement activity across the various attractions.

## Reference Documentation

|  |  |
| --- | --- |
| **Document Name** | **Relevance and Key Sections** |
| Rally | Describes Use Cases for this component |
| Fogbugz \* | Describes Bugs for this feature |
| xBRMS High Level Technical Design | Describes Feature Set for this component |
| xBRMS Interface Control Document | Describes REST API for this component |
| Synapse Test Plan | Describes overall test approach for Synapse |
| Synapse Environment Server List | Description of VMs used in test environments |
| xBRC ICD | External component REST API |
| Events and Audit – System Design Document | Describes an enterprise database of security and system health events for all xConnect components. |
| Events And Audit – Test Cases | An excel document with Events and Audit test cases to be executed. |

## Related Persons

|  |  |
| --- | --- |
| **Role** | **Name** |
| xBRMS Tester | Michael Lampi |
| xBRMS Tester | Francisco Gonzales |
| Synapse QA Manager | Mark Mecham |
| Synapse Release Manager | Ric Merrifield |
| GxP Developer | Brad Sokola |
| GXP Tester | Robert Lindlay |
| Events and Audit System Tester | Francisco Gonzalez |

# High Level Use Cases

These use cases are the high priority scenarios for the GXP Test and Survey milestone.

The typical health monitoring use case is to validate that each xBRC and IDMS server involved in this deployment is in a green or healthy state. If the service falls out of the green status, an actionable status message is immediately available for troubleshooting actions.

The typical xBRC configuration use case is to apply property updates to one or more xBRCs. This includes: setting the GXP URL, setting the attraction ID and venue name, setting the JMS URL and credentials, and setting other properties as needed.

The typical API case is to expose the facilities endpoint for GXP to discover the IP address of the xBRC managing the attraction that the cast member is logging into.

The typical JMS Listener use case is to capture JMS messages from GXP and all xBRCs. These events are written to the xBRMS database in tables specific to the message class. These events are available to the xi component within seconds of the guest action being performed:

* GXP events are available for entitlement redemption, blue lane denials, and blue lane overrides.
* xBRC events are available for reader events (kiosk), guest moving to entry state, guest moving to merge state, and guest abandoning.

For the Assign Readers trial the network DNS will direct xBRs and xTPs to the xBRMS system for placement with xBRCs, or retained for assignment on the Assign Readers page.

The Edit XBRMS Configuration case handles changing the settings for the xBRMS to work with different network and guest database configurations.

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.

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.

# Scope

The following features of the xBRMS are considered in scope in the following phases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Feature** | **Pilot** | **May** | **June** | **Oct** | **R1** | **xTable 1.1** |
| xBRC Health and Reader Health UI | test | x | x | x | x | x |
| IDMS Health UI | test | x | x | x | x |  |
| xBRC Property Editor UI | test | x | x | x | x | x |
| Facilities API | test | x | x | x | x | x |
| xBRMS Database for JMS messages | test | x | x | x | x |  |
| JMS Listener writes to xBRMS DB | test | x | x | x | x |  |
| xBRC configuration storage | test | x | x | x | x | x |
| Parks supported | 1 | 1 | 1 | 2 | 4 |  |
| Attraction xBRC supported | 10 | 10 | 10 | 17 | 91 |  |
| Kiosk xBRC | 1 | 1 | 1 | 2 | 4 |  |
| Park Entry Model | - | test | x | x | x |  |
| GFF | - | - | test | x | x | x |

# External Dependencies

|  |  |  |
| --- | --- | --- |
| **External** | **For** | **Description** |
| GxP | Test | Calls Facilities endpoint in xBRMS API |
| GxP | Pilot | JMS message stream for redemption, override, denial, offer sets |
| xBRC | Pilot | Status of xBRC and xTP (and xBR) health |
| xBRC | Pilot | Configuration API endpoint for applying xBRC property updates |
| IDMS (test harness) | Pilot | Status of IDMS health |
| IDMS (test harness) | Pilot | Card/Band and Guest Lookup for message serialization |

# Exit Criteria

The exit criteria are as follows:

1. After 72 hour load test the system is still meeting all performance and health targets.
2. Guest entitlements correctly create xBRMS database records.
3. Each cast member deny and grant override behavior is tracked in xBRMS database.
4. Guests moving correctly through state transition is tracked accurately in xBRMS database records
5. Automated install and BVT test suite execution passes.

# Performance Targets

* Five seconds or less from Touch to xBRMS database record.
* 30,000 guests per hour can move through the xBRMS during load testing.

# Test Strategies

## Core Strategies

The core testing strategy is event driven testing used to generate the touch, scan and read events using a combination of software and hardware readers. These events will then cascade through the xBRC, GXP, JMS and xBRMS modules. The test fixture focuses on simulating each of these actions in software simulators to push those same events to real hardware and to monitor the results in the subway diagram, JMS messages and other outputs.

The breadth strategy is to allow all test suites to run on all environments through configuration files so the final test lab in Orlando can execute the same test suites. Potentially, even some of these tests can run on the actual park implementation.

The load strategy is that a realistic model of guest activity driving all key use cases to completion (that can run day in and day out) will provide the best overall test of whether the system is ready to go. So building out a realistic load test environment with as many partner systems fully operational is the key testing goal for this release.

## JMS Messages

Each xBRC model generates a different set of JMS messages that need to be serialized to the xBRMS database.

The test strategy is to create all variations of xBRC model messages:

1. Attraction: Guest to Entry (\*1,2,3)
2. Attraction: Guest to Merge (\*1,2,3)
3. Attraction: Guest to Load (\*3)
4. Attraction: Guest to Exit (\*3)
5. Attraction: Guest Abandoned (\*1,2,3)
6. Kiosk: Reader Event (\*1)
7. Guest: observed (\*1,2,3)
8. GXP: blue lane deny (1,2)
9. GXP: blue lane override (\*1,2)
10. GXP: entitlement redemption (\*1,2)
11. GXP: entitlement booked (UIE)

The parenthetical values indicate the following:

\*1 – xTP reader

\*2 - Mobile GXP reader

\*3 – xBR reader

The overall test strategy for testing the message variations

1. With GxP connected, build out an xBRMS in this variation using software simulated readers and a mix of valid entitlements and cards/bands without entitlements
2. ECHO statement supported by all hardware readers allows the test framework to drive desired load without needing to manually activate the readers.
3. Software reader simulators must behave identically to hardware readers
4. Send specific events to the software readers and observe guest movement using facility view by both manual event generation and by automated test generation.
5. The load test sends high volumes of data through the system to verify scale and performance
6. Replace software readers with hardware readers on integration lab and repeat tests
7. Push entitlements directly into GXP DAP for Load test usage
8. Book entitlements by automating UIE server to book entitlements against GXP SOR inventory
9. Verify correct JMS output by validating xBRMS contents

The following model variations are supported for the GXP Test and Survey release:

1. Attraction (Touch Readers) [one entry xTP reader, one merge xTP reader, GxP ticketing]
2. Attraction (Mobile Touch Readers) [one entry DLI reader, one merge DLI reader, DLI handles GXP]
3. Space (Kiosk Touch Readers) [four xTP readers supporting four Kiosk screen installation]

GXP Test and Survey Cast Member trial only:

1. Attraction (Full) [one entry xTP, one merge xTP, xBRs at entry, merge, load and exit, GxP ticketing]

Post Pilot functionality

1. Park Entry [eight xTP + xBio readers, OMNI ticketing]
2. Space (GFF) [multiple xTP and xBR readers]

## xBRC / IDMS

The xBRC and IDMS servers send a frequent STATUS message to the xBRMS which is serialized in to an xBRMS table capturing at least the following: the version, model, name, venue, timestamp of last message, and reader status.

The test strategy is to connect a variety of xBRC and IDMS servers to the xBRMS and verify the status messages by observing the Health UI page and the database contents.

## xi

The xBRMS database contains entitlement data booked through UIE and redeemed through the xBRCs. It also contains xBRC guest state transition details showing the time spent at each stage of the queue.

The test strategy is to generate a combination of specific use case events and a comprehensive set of load data and to:

* Verify that the xi component correctly displays entitlement and guest activity
* Verify that it behaves reasonably if only some of the data is available, since this will be the case in some testing environments.

## Reader

A software reader simulator provides the exact same functionality as the hardware readers in the following areas and, as such, it is useful for the majority of xBRMS testing.

Connecting hardware readers later in the testing cycle will validate the similarity assumption and allow further testing for memory leaks and for any long running issues with the hardware readers themselves.

The core functionality between the reader and xBRMS includes:

* The reader sends a HELLO message on a short time interval of roughly one minute broadcasting its IP address, name, and version number among other properties. The xBRMS adjusts to a new IP address of the reader, and pushes back any name changes made on the xBRMS for the reader.
* The xBRMS scans its xBRC tables for where the reader should be assigned. If a reader is found to be assigned to a given xBRC the xBRMS updates the address in the reader to where future HELLO messages are to be sent to the address of the xBRC. If there is a VIP Address definition in the xBRC configuration, then the reader is given the VIP Address instead.
* If there are no assignments for the reader then the reader is placed in the Unassigned Readers list on the xBRMS.
* If there are multiple xBRCs for which a reader is assigned, then the reader is placed in the Unassigned Readers list on the xBRMS and an error notification on the xBRMS web UI is displayed when the user accesses xBRMS.
* Reader health is obtained from the xBRC and displayed on a Reader Health page.

## API

The xBRMS API is expressed as a standard REST interface. In this release the REST API is provided over a plain HTTP protocol with no security model. This provides the maximum test surface area with minimum decoding overhead.

The REST API simply consists of a single facilities endpoint which describes the attached xBRC servers. This endpoint is exercised by viewing in the browser after creating a variety of test configurations and to validate that the information is correct.

The authentication model on this endpoint must be no authentication.

## UI

The xBRMS UI is written in struts and executed by TCServer located in the /opt/apps directory structure. Web pages use only HTML style calls. Note: The xBRMS UI supports only Firefox in this release.

The xBRMS UI will be exercised using a variety of tools including: hard coded hyperlinks available for manual testing, verifying the API interactions, and observing system health during manual and load testing.

The operation of the xBRC Property Editor, xBRC Health, and Reader Health will be verified using manual testing for the GXP Testing and Survey release.

In future releases, a rich automation framework will be built using the following strategy.

* Overall UI strategy:

1. Set up a test framework in Selenium and node.js framework that supports the browser matrix
2. Run all UI tests against all supported browsers

* The supported browser matrix for this release is Firefox. Additional browsers for future release are:
  + IE 6 on Windows XP with Chrome Frame
  + IE 6 on Windows XP without Chrome Frame
  + Safari on IOS 5
* Additional browsers which may be tested / used for fun:
  + Chrome

## Database

The xBRMS stores internal state in a SQL database named XBRMS.

Key database behaviors to verify:

1. Each message is logged to the correct table with the correct properties
2. The JMS Listener log file does not contain any serialization errors or warnings

## Events and Audit Subsystem

Events and Audit Subsystem is a new feature that will be released as part of 1.7. The goal is to 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.

In the scope of this document are included events and audit entries generated by the xBRMS to the Audit table in the xBRMS database; 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.
* 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.

## Test Strategy Details

Strategies are as follows:

1. Configure all core use cases with software reader simulators in BVT, SIT, and INT environments. This includes four xBRCs: two attractions (one with xTP, one with DLI readers), one Kiosk with multiple readers, and one park entry. This also includes all other necessary servers to build out the environment including IDMS, GXP and xBRMS.
   1. Configure INT bench as close to the final park as possible with one xBRC per attraction supported, the latest GXP version, and if time permits, the current version of UIE and Kiosk for manual integration testing.
   2. BVT Bench will be configured for and provided with the set of additional tests to perform build regression testing.
   3. SIT Bench will be configured for High Availability and failover testing.
2. Configure software and hardware readers with an ECHO statement to drive the same load from automation bench through both types of readers, allowing the reader to be stubbed or to be stress tested.
3. BVT test suite touches each API and UI endpoint.
4. Attach hardware readers to the benches at various times to validate that the hardware reader provides status messages as expected and can handle the load over time without performance degradation.
5. Load test suite runs a high average traffic simulation through each test environment constantly. The load test is updated to include the top test cases that represent significant capacity load starting with green light, then adding blue lane and other use cases. The load test does not verify actions, rather it is monitored through Subway maps, xBRMS database tables, SNMP and performance metrics to verify that system is still performing after 72 hours. The goal is to observe each touch event as a JMS message that is written out to the xBRMS in a timely manner.
6. One touch = one event – generate touch events through the ECHO statement and capture JMS events emitted then match up the events with the touches.
   * Requires: GxP is populated with correct entitlements before running test
   * Automation Requires: JMS test hook tool (or xBRMS sniffer) that sends specific JMS messages back to the test suite
   * Requires: writing GxP blue lane deny and blue lane override messages to xBRMS
   * Requires: error cases must generate a trappable event probably on a test topic (blue lane rejected, illegal state transition (e.g., merge back to entry)
7. Build to build update is fully automated
   * Requires configuration scripts for each environment and each attraction
8. Changes made to the configuration and readers are verified in mysql
   * Requires: reading and validating the Config, Stored Config, and Reader tables

# Data Needs

Data needs are as follows:

1. Load test data for a single day, a single week, and for a week compressed without park closures for a full day of happy guests.
2. An easy method to call to add a single entitlement for a single band to run a single test case.
3. An easy method to clear GxP entitlements when starting a new test suite.
4. IDMS data set with all card/band ids needed for all test cases (segregated test data which is easy to filter out in xi).
5. Self-Test Data Set – to include in the production run and make sure the system works correctly end to end ideally through automation.
6. JMS event expected results for the load test scenario.
7. Configuration scripts in a central location.
8. UIE test harness capable of booking hundreds of offer set entitlements.

# Validators

* xBRMS database validator
* JMS message validator
* SQL schema validator tool
* SQL diff between various test benches

# Test Tools

* Bash script to automatically update an environment to a new build
* Python script to apply stock configuration files to one or more machines from within test case
* Test links page for each test environment to facilitate manual testing suite
* DAP UI override through REST call can be made from test case
* Load test generating a day in the life of the park across attractions and kiosks
  + --- POST GXP Test and Survey ---
* xBR readers support ECHO statement
* Selenium test execution engine that writes to Rally
* Node.js test execution engine that writes to Rally
* Rally test case management
* DLI readers support ECHO statements