xBRC Test Plan

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Description** |
| 1.0 | 5/1/12 | Timothy Sharpe | Release Version |
| 1.5 | 1/16/13 | Michael Lampi | Updated to include xTable 1.1 and High Availability |
| 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**

|  |  |  |  |
| --- | --- | --- | --- |
| **Pri** | **Due Date** | **Owner** | **Issue** |
| 2 | 5/30/12 | Manny | /etc/init.d/bootcserverssp does not reliably manage tcserver execution – will not stop if started as root, or if pid file is deleted. |
| 2 | 5/30/12 | Manny | Final file locations and names under /opt/apps |

**Closed Issues**

|  |  |  |  |
| --- | --- | --- | --- |
| **Pri** | **Due Date** | **Owner** | **Issue** |
| 2 | 4/20/12 | Ric | Secure ID will be obscured and/or encrypted in xBRC and xBRMS logs and databases for Test and Survey, with tighter controls going forward. |

**Table of Contents**

[1 Introduction 4](#_Toc361228016)

[1.1 Purpose 4](#_Toc361228017)

[1.2 Reference Documentation 4](#_Toc361228018)

[1.3 Related Persons 5](#_Toc361228019)

[1.4 Scope 5](#_Toc361228020)

[1.5 External Dependencies 6](#_Toc361228021)

[1.6 High Level Use Cases 7](#_Toc361228022)

[1.7 Exit Criteria 8](#_Toc361228023)

[1.8 Performance Targets 9](#_Toc361228024)

[2 Test Strategies 10](#_Toc361228025)

[2.1 Core Strategies 10](#_Toc361228026)

[2.2 Model 10](#_Toc361228027)

[2.3 Reader 11](#_Toc361228028)

[2.4 API 12](#_Toc361228029)

[2.5 User Interface 13](#_Toc361228030)

[2.6 Database 14](#_Toc361228031)

[2.7 Events and Audit Subsystem 15](#_Toc361228032)

[2.8 Test Strategy Details 15](#_Toc361228033)

[3 Data Needs 17](#_Toc361228034)

[4 Validators 18](#_Toc361228035)

[5 Test Tools 18](#_Toc361228036)

# Introduction

Please note that throughout this document page names are highlighted in **bold** while page element names are in *italic*.

## Purpose

This document describes how the xBRC component will be tested. The xBRC will be tested in an integrated manner in an environment with both the input readers and external ticketing components that can be turned on and off as needed by the tests.

The xBRC component consists of UI, API and Model 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 model aspects will be tested using simulated software readers and hardware readers by generating and pushing events through these readers using REST calls and by observing the results through web UI and database level verifications.

## Reference Documentation

| **Document Name** | **Relevance and Key Sections** |
| --- | --- |
| Rally | Describes Use Cases for this component |
| Fogbugz \* | Describes Bugs for this feature |
| xBRC High Level Technical Design | Describes Feature Set for this component |
| xBRC Interface Control Document | Describes REST API for this component |
| Test Strategy | Test Strategy Document |
| Synapse Environment Server List | Description of VMs used in test environments |
| GxP ICD | External component REST API |
| xTP ICD | Touch Reader REST API |
| IDMS ICD | IDMS 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** |
| --- | --- |
| xBRC Tester | Michael Lampi |
| Test Manager | Mark Mecham |
| Release Manager | Ric Merrifield |
| xTP Developer | Greg Strange |
| xTP Tester | John McLean |
| GxP Developer |  |
| GXP Tester | Robert Lindley |
| Events and Audit System Tester | Francisco Gonzalez |

## Scope

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

| **Feature** | **Pilot** | **May** | **June** | **Oct** | **R1** | **xTable 1.1** |
| --- | --- | --- | --- | --- | --- | --- |
| Attraction Model (Touch) | test | x | X | x | x |  |
| Attraction Model (Mobile Touch) | test | x | x | x | x |  |
| Space Model (Kiosk) | test | x | x | x | x |  |
| xBRC API | test | x | x | x | x | x |
| xBRC UI | test | x | x | x | x | x |
| xTP Readers | test | x | x | x | x |  |
| Mobile DLI Readers | test | x | x | x | x |  |
| Facility View | test | x | x | x | x |  |
| Self-Test | test | x | x | x | x |  |
| Location Editor | test | x | x | x | x | x |
| Parks supported | 1 | 1 | 1 | 2 | 4 |  |
| Attractions supported | 10 | 10 | 10 | 17 | 91 |  |
| Kiosk xTP | 4 | 4 | 8 | 8 | 91 |  |
| xBR | cast only | cast only | x | x | x | x |
| Attraction Model (Full – Entry, Merge, Load, Exit) | cast only | cast only | x | x | x | x |
| xTP + xBio Readers | - | 9 | 9 | 9 | 91 |  |
| Park Entry Model | - | test | x | x | x |  |
| GFF | - | - | test | x | x | x |
| Vehicle Association | - | - | test | x | x |  |
| Band Transmit | - | - | Test | x | x | x |
| Media Sequence Management | - | - | Test | x | x |  |
|  |  |  |  |  |  |  |

## External Dependencies

| **External** | **For** | **Description** |
| --- | --- | --- |
| GxP | Test | Integrated GxP redemption API on SIT, INT, and BVT benches |
| GxP | Test | DAP UI for granting or denying blue lane overrides |
| GxP | Test | REST API for Blue lane events on DAP UI |
| GxP | Pilot | JMS message stream for redemption, override, denial, offer sets |
| GxP | Pilot | Blue lane callback with override status |
| GxP | Pilot | Attraction Inventory and Entitlement management |
| GxP | Pilot | End to End Booking and gxp-link-id API set |
| xTP | Test | automation ECHO statement for driving tap load |
| xBR | Test | automation ECHO statement for driving read load |
| IDMS (test harness) | Pilot | Card/Band Lookup, Guest Creation and Lookup for xBRC and GXP |

## High Level Use Cases

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

* In the typical **Attraction Use Case** for the Pilot, a guest taps a card or a band onto a touch reader inside the attraction. This touch is passed on to the GxP ticketing system that either grants or rejects admission to the attraction. The cast member is able to grant an override to the guest or not and this decision is traceable by placing the results in the xBRMS database and other GXP databases. The response to this touch event moves the guest through a predefined attraction model from entry to merge to exit. The transitions from one state to another are communicated through JMS messages and provide detailed queue metric information for all attractions in the pilot.
* In the typical **Mobile Reader Use Case,** a portable DLI reader (running on a tablet device) is held by the cast member and a guest taps the card on a reader in the front. The unit verifies the card against either an offline GxP cache or the online GxP service. When the guest has entered or merged, the DLI unit sends a tap event to the xBRC to move the guest through the attraction model and to generate queue metrics [though no calls to GxP are made by the xBRC in this transaction].
* In the typical **Kiosk** **Use Case,** a guest taps a card (or later on a band) onto a touch reader in front of a kiosk machine. This touch is immediately raised as a JMS message and caught by the kiosk application that shows the upcoming entitlements for that guest on the associated kiosk screen for a short period of time before assisting the next guest.
* The typical **UI** **Use Case** for the Pilot is to manage the association to readers, configure them as entry or merge readers, and verify that the readers are correctly managing guest touch events by observing guest movement on a subway diagram.
* The typical **JMS** **Use Case** for the Pilot is to verify that all subway diagram events are made available to the JMS table within the xBRMS database.
* The typical **API** **Use Case** for the Pilot is to manage the xBRC configuration of properties, readers, and subway map. The API allows saving and reapplying configurations in a hierarchical layer method.
  + Another typical **API** **Use Case** is to watch xBRC and reader status from xBRMS (this is tracked in the xBRMS test plan).
* The typical **GXP** **Use Cases** are used to discover the readers using the reader API, to show reader status on the DAP UI by readerID, to call back to the xBRC and cause green or blue lane lighting patterns on the correct reader (by readerID).
* All interactions within the xBRC utilize the SecureID. This SecureID is safely written to the log file by removing the beginning digits. This SecureID is passed to GXP for verification. This SecureID is not written in plain text in any downstream database.
* The xBRC manages the software version of attached hardware readers. If a newer software version is installed on the xBRC, when the xTP device sends the next HELLO message, that software will be pushed down to the reader and the reader will be restarted with the new software and reconnect automatically.
* The **High Availability (HA) use case** uses two or more xBRCs and a F5 BigIP virtual server appliance. Readers communicate with a VIP (virtual IP address), and traffic to and from that VIP are directed to a single xBRC in the HA pool. This xBRC is designated as the master, and the other xBRC is designated as the slave. When the master xBRC goes offline, the BigIP system selects an available xBRC from the pool and directs traffic to that system. When an xBRC in the pool comes online it will detect traffic on the JMS bus and become a slave, copying its configuration from the current master xBRC in preparation to take over if the master goes offline.

These use cases are intended to verify the Event and Audit System functionality from the xBRC point of view (in which a new event will be written to the Audit table in the xBRC database.)

* User logs in to the xBRC UI.
* User makes and saves a change to a location.

## Exit Criteria

| **#** | **Exit Criteria** |
| --- | --- |
| 1 | After 72 hour load test the system is still meeting all performance and health targets |
| 2 | Guests can successfully redeem all entitlements |
| 3 | Cast members can deny and grant overrides |
| 4 | Guests correctly move through state transition and this movement is shown accurately on Facility View |
| 5 | Automated install and BVT test suite execution passes |
| 6 | (stretch) GFF and PE are tested for the Alpha lab deployment |
| 7 | High availability and failover from master to slave (and back) works transparently |
| 8 | A mix of xBR v3 and xBR v4 readers are accommodated by a single xBRC |

## Performance Targets

* 1 second or less from Touch to Green light for guest interactions with xTP
* 3,000 guests per hour can move through the xBRC during load testing

# Test Strategies

## Core Strategies

The core testing strategy is event driven testing. To generate the touch, you 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 that 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) provides 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.

## Model

There are several models which can be loaded into the xBRC, but fundamentally they all connect readers to ticketing systems and to monitoring systems. In fact all xBRC models require at least one reader and create a set of JMS messages that other services and monitoring tools can consume.

The overall test strategy for testing the model variations is:

* Build out an xBRC in this variation using software simulated readers and with GxP stubbed out
* ECHO statement supported by all hardware readers allows the test framework to drive desired load without needing to manually activate the readers.
* Software reader simulators must behave identically to hardware readers
* Send specific events to the software readers and observe guest movement using facility view by both manual event generation and by automated test generation.
* The load test sends high volumes of data through the system to verify scale and performance
* Replace software readers with hardware readers on integration lab and repeat tests
* Verify correct JMS output sent from xBRC

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

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

GXP Test and Survey Cast Member trial only:

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

Post Pilot functionality

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

## Reader

A software reader simulator provides the exact same functionality as the hardware readers in the following areas and, as such, is useful for the majority of xBRC 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 xBRC include:

* 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 xBRC adjusts to a new IP address of the reader, pushes back any name changes made on the xBRC for the reader.
* The reader sends status messages describing the internal status of the reader and the xBRC adds additional reader status information including if the reader is responsive or not and sends this information up to the xBRMS health page.
* The xBRC manages the software version of the attached readers. The hardware reader has a specific type and if newer versions of the software for that type of hardware are on the xBRC, they are downloaded to the xTP at the next hello message that happens about every minute.
  + The new hardware reader packages are opack files and are brought to the xBRC as a yum package.
  + These can only be applied to xTP, xTB+xBio, and xBR readers; they cannot be applied to DLI readers or to software readers.

## API

The xBRC 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 will be exercised using a variety of tools including:

* Fiddler (or SOAP UI) to create specific API calls and to observe the results
* Python to send API calls to multiple VMs within a test environment
* Node.js to validate the API responses and hard coded hyperlinks available for manual testing

The simplest tests are hyperlinks on a test links page which make it very easy to gather status information from the xBRC including: status, readers, configuration, current guest states, and recent JMS messages. These provide a quickly observable view into the status of the xBRC.

The next level of tests involve using Python to execute a set of REST calls across the entire test environment to configure readers, subway maps, external component URLs and other properties.

In future releases, the set of automated tests in this area will be built out to validate throughput, correctness, limits and responses.

The xBRC API expresses a variety of endpoints that fall into three categories: configuration, status, and testing.

* Configuration
  + The configuration APIs provides access to update properties, readers and the facility view. The following aspects are tested
  + Properties
    - All properties are editable using the xBRMS configuration UI
    - Changes to properties correctly affect internal timeouts, URLs, and behavior variations
  + Reader sets
    - All readers and locations are editable using the xBRC Location Edit UI
    - Updates the Reader and Location tables within the Mayhem database
    - Readers can be pre-configured within a specific location
    - New readers can be discovered and are added to the Unknown Location
  + Facility View
  + Set the background image to match the attraction
  + Set the subway nodes to approximate the guest queue visually including guests stacking up at queue points based on increasing wait times
  + Make sure that the correct facility view is configured for each attraction
* Status
  + The status APIs provide access to performance metrics, reader health and current activity.
  + They will be utilized to measure system performance and monitored as part of the load test
  + Some are consumed by GXP and xBRMS
* Testing
  + Endpoints used specifically for testing may be used to enable test cases, but will not be tested.

## User Interface

The xBRC UI is written in struts and executed by TCServer located in the /opt/apps directory structure. Web pages use both HTML and AJAX style calls.

The xBRC UI will be exercised using a variety of tools including: hard coded hyperlinks available for manual testing, verifying the API interactions, and watching guests move through the facility views (or subway diagrams).

The operation of the Location Editor and Subway map 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:

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

The supported browser matrix is:

* Firefox
* 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 xBRC stores internal state in a mysql database named Mayhem. The user is EMUser and the password is Mayhem!23. This database schema must match downstream data types. The database must balance record refreshes against component performance.

Key database behaviors to verify:

* The Location table must contain a record at id = 0 with name = UNKNOWN or new readers will not be discovered and made available to the xBRC
* The Image table must contain a complete image file that is linked to the facility view
* Readers must have a unique reader ID
* Locations must have a section equal to the attraction ID (or otherwise provided by GXP team)
* Status table must contain a LastMessageToJMS property that is < the highest message sequence number in the Messages table.
* GST records must age out over time by going through either the EXITING or ABANDONED state

## 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 xBRC to the Audit table in the Mayhem database; audit data on:

* Authenticated or anonymous access
* Read, write, update and delete operations in the Mayhem database
* Generates audit data on user logins and logouts.

## Test Strategy Details

* **Strategy #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.
* **Strategy #1a** – 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.
* **Strategy #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.
* **Strategy #3** – BVT test suite touches each API and UI endpoint and as the top use case for each xBRC type.
* **Strategy #4** – Attach hardware readers to the benches at various times to validate that the hardware reader can handle the various xBRC variations and take the load over time.
* **Strategy #5** – Load test suite constantly runs a high average traffic simulation through each test environment. 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, xBRC 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 which is written out to the messaging bus in a timely manner.
* **Strategy #6** – Selenium and node.js test framework enables simple web and REST call development
* **Strategy #7** – Test setup is managed through configuration files including
  + 4 readers (2 entry, 2 merge)
  + 2 readers (1 entry, 1 merge)
  + 16 readers (8 entry, 8 merge)
  + No GxP (GxP disabled)
  + No IDMS (IDMS disabled)
  + No JMS (JMS disabled)
  + No Externals (GxP, IDMS, JMS disabled)
  + Yes GxP (bench specific)
  + Yes IDMS (bench specific)
  + Yes JMS (bench specific)
  + Attraction specific (10 variations)
  + Kiosk small (2 kiosks)
  + Kiosk normal (9 kiosks)
  + Facility Map variations (several)
* **Strategy #8** – 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 xBRC
  + 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)
* **Strategy #9** – watch guests move through the subway map
  + Automation Requires: the ability to sniff AJAX to discover each guest on the subway map (or directly in the GST table)
* **Strategy #10** – build to build update is fully automated
  + Requires configuration scripts for each environment and each attraction
* **Strategy #11** – Changes made to the configuration and readers are verified in mysql
  + Requires: reading and validating the Config, Stored Config, and Reader tables
* **Strategy #12** – Subway = guest behavior (load, and test) = xBRC tables
  + Requires: selecting the JMS messages for a specific test event from the xBRC databases
  + Requires: single time format across databases for traceability
  + Requires: traceability from load test inputs to xBRC outputs (guestID or bandID traceability)
* **Strategy #13** – To manually create subway maps, then validate guests flowing through various subway map designs

# Data Needs

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

# Validators

* xBRC database validator
* JMS message validator
* Subway diagram validator
* Mysql record update validator
* Mysql schema validator tool
* Ability to validate some sort of event for all taps (logs, JMS, etc)
* AJAX validators for facility view

# Test Tools

* MySQL Admin tool with visual UI to enable quick ad hoc of database values
* 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 the 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
* A 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