xTP Test Plan

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Description** |
| 1.0 | 5/1/12 | Joseph Ferrara | Release Version |
| 1.5 | 11/9/12 | Joseph Ferrara | Reflects changes through 11/1/2012 |
| 1.6 | 5/8/13 | Joseph Ferrara | Reflects changes to firmware through 5/8/2013 |

**Document Approvers & Sign-Off**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Approver** | **Role** | **Document Accept/Reject** |
| 5/1/12 | Mark Mecham | QA Manager | Accept |
| 5/8/13 | Mark Mecham | QA Manager |  |

**Table of Contents**

1 Introduction 6

1.1 Purpose 6

1.2 Definitions 6

1.3 Scope 8

1.4 Background 8

1.5 Assumptions and Constraints 11

1.6 Reference 12

1.7 Software Version 12

1.8 Contacts 13

2 Exit Criteria 13

3 Interfaces 14

3.1 RFID radio 14

3.2 Biometric Reader (Connected by USB) 15

3.3 REST API 15

Event Types 16

3.4 Media Interface 16

4 Areas Tested 18

4.1.1 Definition of Models 18

4.1.2 High Level Use Cases 19

4.2 Smoke 19

4.3 Build Verification 19

4.4 Integration 19

4.4.1 Finding an xBRMS on startup 19

4.4.1 Locking 20

4.4.2 Hardware 20

4.5 Builds 21

4.6 Security 21

4.6.1 Secure ID and Public ID 21

4.7 Setup and Upgrade 22

4.7.1 Make a Runnable SD Card 22

4.7.2 Make an SD Card for Flash 23

4.7.3 Updating Flash Image on an xTP 24

4.7.4 Upgrade an xTP 24

4.8 System Tests 25

4.8.1 Load Testing 25

4.8.2 Stress Testing 26

4.8.3 Performance Testing 26

4.9 Accessing Device File System 27

5 xTP Test Strategies 28

5.1.1 Resources Required 30

5.2 Test Tools 31

The automation written in Java is also located in the xConnect Test project. The files there can be imported into any IDE where tests can be run interactively in a test runner such as those included in IntelliJ or Eclipse. It is also possible to run the tests from the command line, though this is not documented here. 33

6 Testing an xTP Device After Installation 34

6.1 Purpose 34

6.2 Scope 34

6.3 Background 34

6.4 Assumptions and Constraints 34

7 Tests When Not Connected to xBRC 36

7.1 Manual Physical Tests 36

7.1.1 Touch RFID Card to xTP 36

7.1.2 Touch MagicBand Card to xTP 36

7.2 API Tests 37

7.2.1 Get Reader Info 37

7.2.2 Verify That Events Are Generated 38

8 Tests When Connected to xBRC 39

8.1 Manual Physical Tests 39

8.1.1 Touch RFID Card with Valid Entitlement 39

8.1.2 Touch RFID Card with Expired Entitlement 39

# Introduction

## Purpose

Define test plan and strategy for xTP type readers (short range) readers such as xTP, xTPra, and xTPrs.

## Definitions

| **Term** | **Definition** |
| --- | --- |
| Attraction Model | Scenario where guest uses FastPass+ for an attraction |
| DAP | Disney Access Portal, an xTP configured with a biometric reader for use at Park Entry locations |
| Entitlement | Any consumable product or service that a Guest considers a benefit, that can have a status of available, reserved (booked), redeemed (used) or expired |
| GxP | Guaranteed Experiences – Guests can pre-reserve attractions with a FastPass+ (next generation FastPass), schedule a Character Meet and Greet, and reserve dining through BOG (Be of Guest) restaurant. |
| HTTP | Hypertext Transfer Protocol |
| IDMS | Code and database storing Guest and MagicBand information |
| JSON | JavaScript Object Notation |
| Lumidigm Biometric Reader | Used as a finger print scanner in Park Entry model |
| MagicBand | RFID device worn by Guests |
| Park Entry Model | Scenario where xTP+xBio reader used at Park Entry |
| REST | Representational State Transfer |
| RFID | Radio-frequency identification |
| RPM | RPM Package Manager (RPM) is a package management system. |
| xBR | Long range RFID reader with uni- or omni-directional antennae |
| xBRC | xBand Reader Controller which manages xBRs, xTPs, and DAP devices |
| xBRMS | xBand Reader Management System code and database which stores operational data and manages xBRCs and unassigned readers |
| xConnect | Code, scripts, APIs, and database schemas that comprise the unifying messaging, management, and reporting software that ties the hardware together into a coherent solution |
| xTP | Experience TouchPoint, a Disney-themed short range RFID reader or “tap” device |
| xTPE (xFPE) | Internal box (black color) in reader containing reader’s circuit boards |
| xTPra | Remote access touch point |
| xTPrs | Remote sensor touch point |

## Scope

The xTP reader is tested in isolation from the end-to-end xConnect system. Focus is given to manual and automated test cases directly exercising the API for each reader’s RFID and biometric features. An xBRC is used for testing some APIs that depend on an xBRC being present.

In conjunction with the team responsible for other xConnect software, end to end testing is performed with a focus on monitoring the performance and reliability of hardware xTP readers. There are several test environments defined which are used to run end-to-end tests.

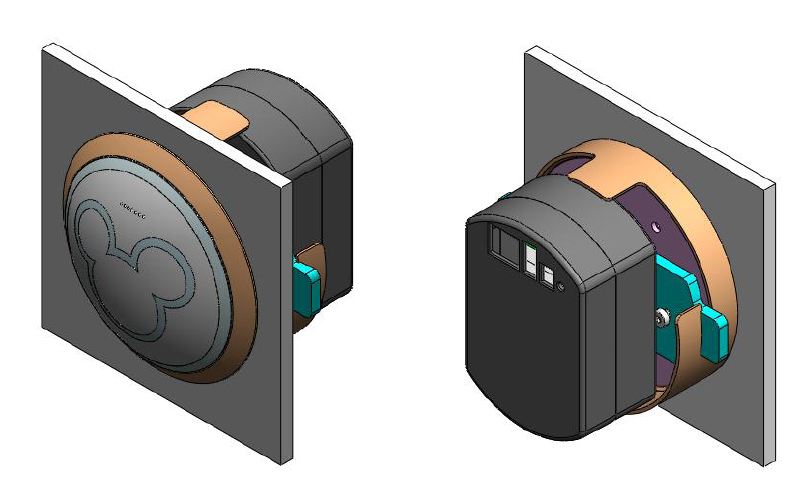
Hardware and software other than the xTP+xBio are not under test with the exception that some xTP features require testing with xBRC and xBRMS software.

Low level testing involving mechanical and electrical changes to the xTP is out of scope.

## Background

The xTP reader is used in several cases including park entry, FastPass+ entitlements, sensor based reads and objects that contain multiple touch points employing remote antennas. When used for park entry, the reader has a biometric reader attached via a USB connection. When an adult pass holder enters, they are required to tap with an RFID card or MagicBand and are also required to provide a fingerprint. Guests who are children (under 10) do not need to provide a fingerprint.

**TouchPoint (Short Range Reader)**



Composed of two components:

xTP (customer-facing circular domed touch plate with LED and sound feedback to customer actions)

xTPE (interior controlling component)

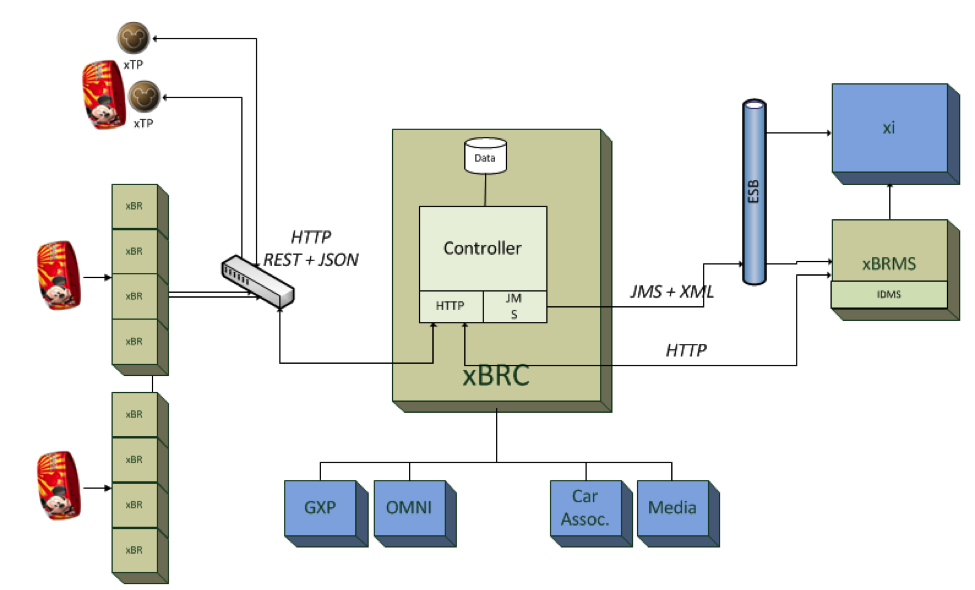
Ethernet connectivity & Direct current required

As used in the parks, the readers are associated with one or more xBRC installations. These are used to determine entitlement status for a guest by placing messages on an Enterprise System Bus (ESB). This allows for back end systems to evaluate entitlements and track guest data for a collection of back end applications. The xBRC tells the xTP reader to light up with appropriate media in response to evaluation of entitlements or error conditions.

xTP readers, also known as short-range readers, are used for guest identification using either an RFID card or a MagicBand. Models for this device are park entry, attraction FastPass+ lines, and in restaurants among others. The readers will detect an RFID card or MagicBand from several inches away in order to read ID information. For the park entry model, the xTP also includes a Lumidigm biometric reader so that fingerprints can be enrolled and scanned for guests.

Variations on the xTP have been developed. One employs a remote antennae allowing for the physical touch point to be located away from the electronics. One use case is where a statue could have multiple touch points and the electronics could just be in the base of the statue. A second variation, xTPrs, allow for use of a remote sensor rather than an RFID based trigger. For example, an xTPrs could be used where a roller coaster attraction breaks a laser beam allowing for timing data to be used for locating guests in one of the cars.

An xTP reader is one component of the overall xConnect project. Below is a diagram showing components for xConnect.

Figure 1- xConnect components and system architecture

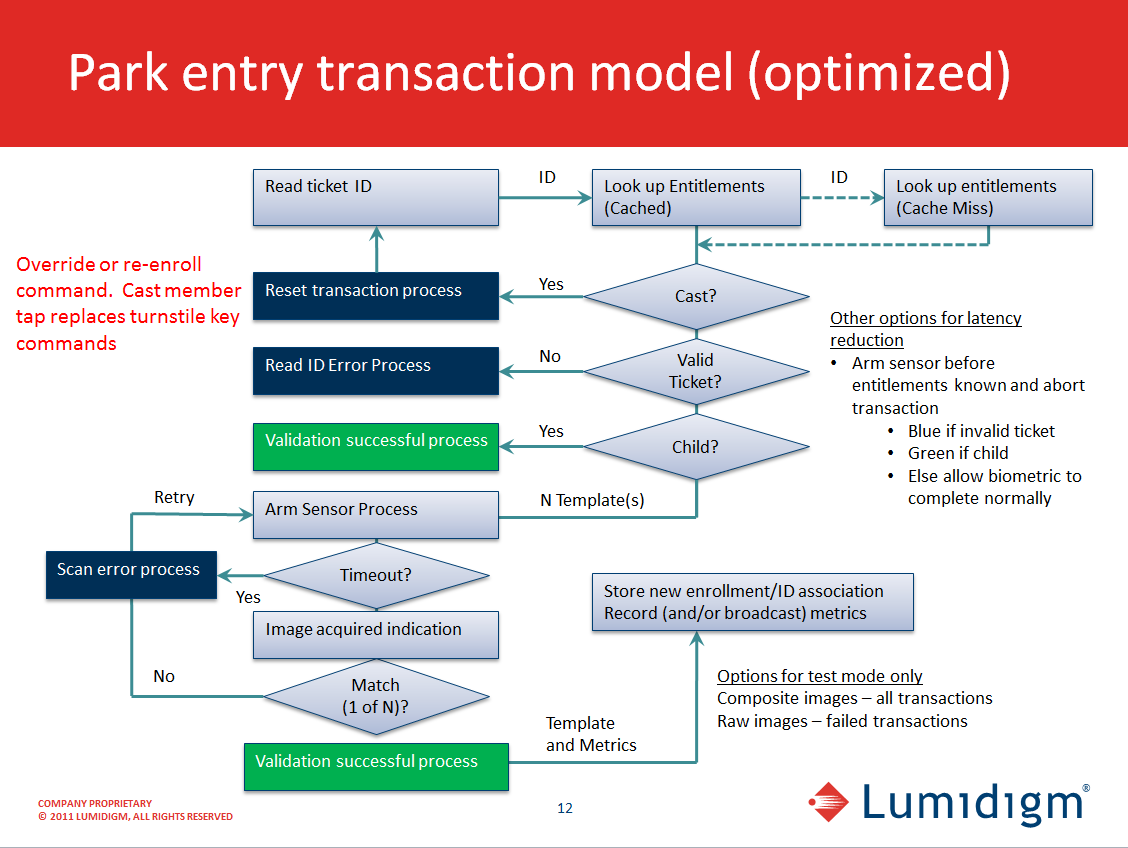


Figure 2 Park entry model

## Assumptions and Constraints

Availability of xTP hardware during the testing phase is assumed. Testing xTP on the component level requires a standalone xTP accessible from a test computer. Testing of the xTP connected to an xBRC requires test benches. Manual tests can be run when connected to the environments. The load tests run by the xConnect team also exercise the xTP software.

## Reference

| **Title** | **Relationship** |
| --- | --- |
| xTP Test Cases – API.xlsx | Documents test cases for xTP |
| 900-026 xTP Interface Control Document.pdf | Documents interface between xTP and xBRC |
| 900-0149 xTP Configuration File Description.pdf | Documents configuration file settings. |
| 900-0058 xBRC Interface Control Document.docx | Documents API for the xBRC |
| 900-0067 xConnect Test Plan.docx | Test Plan for xConnect software modules |
| 900-0071 xBRC Test Plan.docx | Test Plan for xBRC |
| Synapse – WDW - Lumidigm.specs.v0.4.pptx | Details for Lumidigm biometric reader |
| xTP Build 2.5.7 Release Notes.pdf | <Mark: Please Insert document here> |

## Software Version

This document applies to or involves subject matter that requires or is optimized for the following version of software. Details about changes introduced for earlier versions are documented in the xTP ICD. The park only has V2 versions of the xTP installed. The primary differences between the V1 (Engineering Proof of Concept) device and the current V2 device is that the new version has an improved antennae and features faceplate and reader surface that can be scripted to light up using LED arrays.

|  |  |  |
| --- | --- | --- |
| **Software** | **Version** | **Notes** |
| Reader | 2.5.7 | Software internal to xTP |

## Contacts

|  |  |  |
| --- | --- | --- |
| **Role** | **Name** | **Email** |
| SDET | Joe Ferrara | [joef@synapse.com](mailto:joef@synapse.com) |
| SDET | John McLean | [johnmc@synapse.com](mailto:johnmc@synapse.com) |
| Developer | Greg Strange | [greg@synapse.com](mailto:greg@synapse.com) |
| Developer | Corey Wharton | [coreyw@synapse.com](mailto:coreyw@synapse.com) |
| Project Manager | Patrick Stanko | [patrick.stanko@synapse.com](mailto:patrick.stanko@synapse.com) |
| ME Manager | Brian Piquette | [brian@synapse.com](mailto:brian@synapse.com) |
| QA Manager | Mark Mecham | [mark.mecham@synapse.com](mailto:mark.mecham@synapse.com) |
| Lumidigm | Paul Butler | [pbutler@lumidigm.com](mailto:pbutler@lumidigm.com) |

# Exit Criteria

* 100% of planned test cases have been executed and test results have been recorded with a pass or fail.
* Overall test pass rate of 90% is accomplished once Triage decides which bugs can be postponed, if any.
* All Test Cases have been successfully executed and individual interfaces tested and functioning according to design (ICD).
* All the defects have been appropriately documented in FogBugz with correct description, status, and are appropriately assigned to relevant team / individual.
* All test cases are updated as necessary and all expected and actual results related to each test case are documented within FogBugz.
* All known “critical” (Pri 1) and “high” (Pri 2) defects have been resolved and closed.
* All known defects have either been Resolved and Closed during Integration testing.
* All accepted defects have been reviewed, and signed-off by the Program Lead, Program Manager, Development Manager, QA Test Manager.
* Workarounds have been identified and documented for all defects that are accepted.

# Interfaces

## RFID radio

The radio is indirectly tested in all manual tests which employ the reader to process taps. If taps are missed then messages are not sent to the xBRC or reported through direct REST calls.

Tap events are reported in a response to the GET /events API call. Data is contained in an RFID event. This event contains data for uid, pid, sid, iin, and time stamp. The uid is a manufacturing ID set by the RFID chipmaker. When an RFID card is provisioned then the pid, sid and iin are written. The pid is the public ID. The sid is a secure ID and the iid is a portion of the secureid. The secure ID is encrypted and is described in more details in the Security section of this document.

## Biometric Reader (Connected by USB)

This configuration is only present in a Park Entry reader. As part of the entrance procedure for youths and adults, the guest must present their fingerprint in order to validate their park entry entitlement.

This is indirectly tested in integration tests with overall xConnect system. The xBio unit is designed and implemented by an outside vendor, Lumidigm. Bugs can be introduced by physical defect in connection such as a crimped wire of a partial connection. The xTP system software interfaces to the xBio unit via a REST interface.

The API is documented in the xTP ICD document. xBio data is passed back to a listener such as the xBRC through GET /events API or when streamed to an xBRC through PUT /events call. The xBRC can tell the reader to enroll or match a guest using the xBio unit. The REST call includes a bio template to allow for matching the guest’s fingerprint.

An xBio Diagnostic Event is sent from the xBio unit through xTP’s events stream every 5 minutes.

## REST API

The xBRC controller software is the primary client of the REST API. It communicates to the reader via HTTP. An xBRC is usually associated to a reader when the reader is first connected to the network. It is also possible for another system to get streamed events from the device by using the /update\_stream REST end point. As described in another section of this document, the reader will find its xBRMS server where the reader is placed in an unassociated readers set. The administrator of the xBRMS is then responsible for adding that reader to an xBRC controller.

The reader uses the PUT /hello message to its connected xBRC periodically. This allows the xBRC to know that the reader is connected.

Once the xBRC is listening for messages from the reader, the reader sends events as they are generated. The event responses contain different types of events to describe tap ID information, errors, xBio data and several others.

The commands available in the API include those for events, reader status, reset, setup, options, media, and biometrics calls.

## Event Types

* Reboot event
* RFID event
* Biometric read event
* Biometric Scan error event
* Biometric Image event
* xTP Diagnostics event
* xBio Diagnostic Event
* xBio FW Upgrade Event

These events are read through the GET /events API end point. Events can also be pushed from a test program by an “echo” command, PUT /events. The “echo” command is included to facilitate integration testing with the overall xConnect system.

## Media Interface

Defines the API for installing and playing media on the xTP and xBio. Media is sound, colors, sequences of sound and color and LED scripts which allow for programmable sequences.

#### Media Included on default xTP Configuration

{  
 "defaults" : {  
 "ledscripts" : [  
 "login\_ok",  
 "entry\_timeout",  
 "success",  
 "idle",  
 "entry\_start\_scan",  
 "entry\_xbio\_thinking",  
 "thinking",  
 "identify",  
 "exception",  
 "entry\_retry"  
 ],   
 "sounds" : [  
 "prompt",  
 "bioprocess",  
 "biocue",  
 "tap",  
 "sweep1",  
 "sweep2",  
 "retry",  
 "success",  
 "thinking",  
 "tone",  
 "exception"  
 ],   
 "xbio" : [ "scripts" ]   
 },  
 "package" : {  
 "ledscripts" : [  
 "HM\_success\_spooky",  
 "HM\_success\_xylo",  
 "HM\_exception",  
 "GXP\_tap",  
 "HM\_success\_bells",  
 "GXP\_success"  
 ],   
 "media hash" : "bcaceadfb8dcf572d6e220f4dd1f7302",  
 "sounds" : [  
 "HM\_exception\_strings",  
 "HM\_success\_skeletonXylo",  
 "HM\_exception\_skeletonXylo",  
 "HM\_success\_spooky",  
 "GXP\_tap",  
 "HM\_success\_bells"  
 ],   
 "xbio" : null  
 }  
}

#### Media Installation

Media are installed onto a device using the REST endpoint, PUT /media/package. See ICD for details. The body of the API call is in a gzip/tar format. When disconnected from an xBRC (or even if connected), you can use a CURL command or REST client to push the gzip/tar file type. You would verify the call succeeds by calling GET /media/inventory.

# Areas Tested

### Definition of Models

**Park Entry** – When xTP includes both tap reader and biometric reader to support park entry.

**Attraction** – When xTP is used within an attraction’s queue. The xTP can be found at start of queue, merge point between FastPass+ guests and regular queue as well at attraction exit.

**Space** – When xTP is used as a tap device that records user interaction without requiring secure ID information. The reads from a reader in the Space model to not include secure ID. An example is if you have a game that contains choices that the guest taps to choose.

### High Level Use Cases

Guest taps with an RFID card  
Guest taps with a MagicBand   
Guest taps with an RFID card for FastPass+  
Guest taps with a MagicBand for FastPass+  
Guest taps with a secure ID in park entry and other models  
Guest taps where secure ID is not used, as in space model

## Smoke

Development tests their changes before passing it on to QA. QA is then responsible for running all of the tests.

## Build Verification

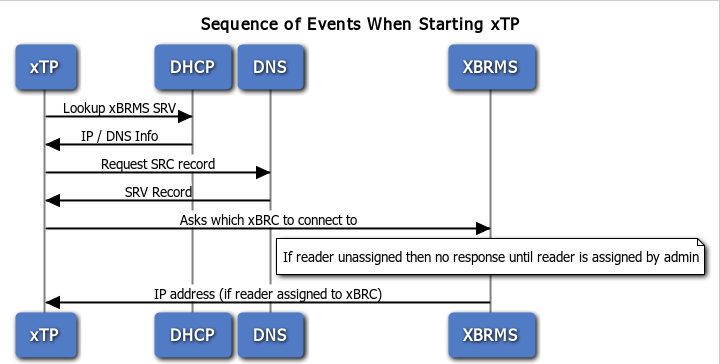
QA runs an automated script as the BVT. It touches on important Restful endpoints and checks returned values in several cases. The script, written in python, is described in the section named “Test Tools”.

## Integration

### Finding an xBRMS on startup

When the xTP is initializing, it attempt to find the xBRMS on its network using its DHCP server. If one is found, then the reader can go into the xBRMS unassigned readers classification if it is not already assigned. If it cannot find an xBRMS, then it will fall back to whatever is set in the “config.json” file.

**Communication Sequence when xTP is Starting Up**



### Locking

When an xTP is in the Blue state, then xTP ignores additional taps. This Ignore state is cleared when a cast member taps to clear the Blue state. A blue state is when the user of a MagicBand does not get an entitlement from the backend and it becomes the operator’s responsibility to deal with the issue.

When an xTP is not yet in a determined state, then it keeps accepting taps until is hears back from xBRC.

### Hardware

The xTP v2 hardware has a faceplate that contains 48 LEDs that can be individually addressed in a LED script. It is possible to light up any of the LEDs with red, green, or blue in 255 intensities. Depending on the installation, the faceplate can be replaced. The one used for park entry and attraction queues is a mickey face.

The xTP v1 hardware (engineering proof of concept) does not have individually addressable LEDs, so it can only be told to turn off, or to change between green and blue colors.

## Builds

xTP builds can be installed via RESTful API calls issued from an xBRC. The xBRC can either do a fresh install or upgrade an existing installation. These API are documented in the xTP IDC document. The xBRC stores xTP builds using Yum. A brief outline of what happens when a new build is placed on the xBRC follows.

1. Build an RPM file  
   Do this by placing the ipk files in git under the appropriate directory for each reader type.  
     
   disney.MagicBand Controller/build/repos/linux/armv7  
     
   Make sure to follow the naming convention of xfp-a.b.c-d.ipk, xbr-a.b.c-d.ipk, xfpxbio-a.b.c-d.ipk
2. Copy it to a directory that the xBRC know to look for new installation packages. Syntax is “sudo yum install <filename>”
3. Make a restful call to the xBRC http://<xbrc ip>:8080/refreshpackages after installing a new version of the reader rpm.
4. When configuring the xBRC, set a location for the RPM files and its IP address.

## Security

### Secure ID and Public ID

A secure ID is read from an RFID device and then decoded by the redemption system in order to match it to a guest. The secure ID is used for redemption to ensure that the Disney ticketing system that the band or card is presented …The secure ID is also important for cases where the RFID device is used to make financial transactions.

When testing in Space model, the reader is configured to not return the secure ID data. A public ID is useful for cases where authentication is not necessary such as for in park attractions that invite guest participation. It is also useful since the read is very fast, only needing to get the ID from the card and not needing to communicate with the xBRC in order to do authentication. For example, it is used in Test Track for a game where guests tap many different locations to play. This attraction uses a variant on the xTP, an xTPra (xTP remote antennae). This allows for faster taps. For this model, only the Public ID and Manufacture’s unique ID are read.

The following four types of ID strings are read from an RFID card or MagicBand:

|  |  |
| --- | --- |
| **ID Name** | **Description** |
| Pid | Public ID (also read by long range readers) |
| uid | Manufacturer’s Unique ID (MUID) or serial number from the RFID chip |
| sid | Account number portion of the secure ID. A 16 digit decimal number. Not included if the RFID option secure\_id is turned off. See **PUT rfid/options** API. |
| Iin | The IIN portion of the secure ID – a six digit decimal number. Not included if the RFID option secure\_id is turned off. |

## Setup and Upgrade

### Make a Runnable SD Card

This form of SD card can be used to run the xTP software directly from the SD card. It does not update the software on the xTP itself. It could be useful in a case where you want to test new changes, but not update the reader software yet.

The following commands will copy an image of a completed xTP software build and copy it to an SD card. The commands assume that a build was successfully completed for the xTP software.

Tools/card.sh images/dap-reader-image/<image>

OR

make dap-reader-image card

In order to get the xTP to use the firmware on a runnable SD card, insert it into the device’s SD card slot and restart the device. It will then be running on the firmware you have on the SD card. If you restart the device without the SD card, then it revertd back to whichever firmware has been installed on it.

### Make an SD Card for Flash

This form of an SD card is used to update the software found on the reader. Section titled “Make a runnable SD Card” describes how to take the prepared SD card and update the reader itself.

1. From the disney.reader top directory
2. tools/update-card.sh images/dap-reader-image/<image>  
     
   The <image> is an image file.
3. Follow these steps when xTP is in a state where SD card slow is exposed. In the V1 xTP readers this first requires removing the top circuit board fastened with screws so that the bottom circuit board is accessible.
4. Insert SD card
5. Turn on power
6. Wait for boot to finish and you’re all set

### Updating Flash Image on an xTP

First create the SD card image using steps at start of section titled “Make an SD Card for Flash”.

Follow these steps when xTP is in a state where SD card slow is exposed. In the V1 xTP readers this first requires removing the top circuit board fastened with screws so that the bottom circuit board is accessible.

1. Insert SD card
2. Turn on power
3. Wait 20 seconds
4. Turn off power
5. Remove SD card
6. Turn on power and test

### Upgrade an xTP

Use the REST endpoint, “PUT /upgrade” to upgrade a device from an existing build to a new one. This endpoint can also be used to kick off a downgrade. This is covered in section 2.7.

Build files from the packages folder should be stored on a web server accessible from an http address. The “PUTR /upgrade” API requires a Body that contains information about how to upgrade system components as well as the reader application.

Example body:

{  
 "repos": [  
 {  
 "path": "http://10.1.201.244/repos/repositories/base-0.0.6-0/all",  
 "name": "all",   
 "weight": "10"  
 },  
 {  
 "path": "http://10.1.201.244/repos/repositories/base-0.0.6-0/armv7a",   
 "name": "armv7a",   
 "weight": "30"  
 },  
 {  
 "path": "http://10.1.201.244/repos/repositories/base-0.0.6-0/overo",   
 "name": "overo",   
 "weight": "50"  
 },   
 {  
 "path": "http://10.1.201.244/repos/repositories/xtp-reader-2.2.0-2542",   
 "name": "xTP",   
 "weight": "90"  
 }  
 ],   
 "downgrade": false  
}

## System Tests

### Load Testing

The xTP is exercised when the end-to-end testing is performed for the end-to-end test. Load tests are executed for park entry and attraction scenarios that include hardware readers. The hardware reader supports an ‘echo’ command which allows for telling the hardware that a tap has occurred and interacting with the connected xBRC.

The end-to-end redemption system is testable in several environments. The following table describes environments used to test the end-to-end system that includes the xTP.

|  |  |
| --- | --- |
| **Environment** | **Explanation** |
| BVT | Build verification tests for latest xConnnect builds |
| SIT | Functional xConnect testing with stable builds |
| HowVille | Test facility that mirrors Disney’s Alpha Lab. Used with stable xConnect builds. |
| Alpha Lab | Disney’s pre-production test lab |
| Park LDU | Logical Data Unit housed at a park, production environment |

### Stress Testing

Use built in utilities to process tap events over and over again. Stress for the overall system can also be demonstrated by driving an xConnect system level stress test.

Rfidtest –l This will exercise the RFID functionality

Use rfidtest –help for complete list of commands

Another way to make the device perform lots of reads then set the test\_loop mode to on by using REST command /PUT /rfid/options?test\_loop=on

### Performance Testing

Performance target is to be as fast as possible with respect to reading data from an RFID card or a MagicBand. The current expectation is that a tap with secure ID should get processed within 350 milliseconds. Ultralight C is currently 250ms.

Performance is measured in two ways.

1) To manually time the process of getting a card or MagicBand close to the reader and measuring how long it takes for the first light to appear.

2) To look at the reader.log file found in /var/logs/ on the device OS. Entries showing the first read of the card to the communication to the xBRC will reveal timings.

The performance target does not include time taken for the tap to be processed by the xBRC and GxP. This is addressed in end-to-end system testing. For more information about system load testing, see section named “Load Testing”.

## Accessing Device File System

It may be useful to access the device file system to check logs or directly verify that settings have your expected values.

1. Determine IP address for device

2. Enter ‘ssh root@<IP address>

3. Enter <password>

Once you are in the Linux shell, you can enter “reader –v” to see the currently installed version of the reader. Some useful directories are described below.

|  |  |
| --- | --- |
| **Location** | **Contains** |
| /mayhem | Main reader application and associated tools are here |
| /mayhem/firmware | xBio firmware |
| /mayhem/ledscripts | Media scripts for xTP reader |
| /mayhem/sounds | Sound files for xTP |
| /mayhem/www | Web server root. Device documentation is available in the docs folder:  ICD.docx, config.docx, release\_notes.docx |
| /mayhem/xbio | Media scripts for xBio reader |
| /var/mayhem | Configuration files, config.json and reader-cache |
| /var/log | reader.log (also accessible via GET /reader/log endpoint) |

As shown in the table above, the xTP contains a reader.log file in /var/log. One way this could be used directly on the device it to look at real time updates to the log using “tail –f” on the log file from the command line. Events show up in the log as well as any errors or warnings within the hardware.

The device contains a Gumstix (<http://www.gumstix.com>) distribution of Linux Overo. It offers a limited set of Linux features specifically needed to support our application.

Its RAM is about 512 MB and its file system has about 128 MB available.

# xTP Test Strategies

* Automation – Automate API calls using a test harness, node.js. Subset of these cases will constitute a BVT. Later integrate automated tests into test benches used for xBRC and other system components.
* Manual Testing – Manually test API calls and use cases that require physical verification such as lights changing or sounds made.
* Reliability – Use the PUT /tap command to run hardware for several hours and monitor effect on component.
* Error Handling – Develop and run sets of test cases which exercise error conditions. The possible error conditions can be derived from source code. The list of error conditions is documented in the source code. The xTP software will pass through an error message so that the xBRMS will expose to the user.
* Configuration Testing - Configuration testing (for example: park entry build, varying values in configuration file; varying physical variables such as light level and temperature.
* Integration Testing – Perform integration testing with biometric reader. Test for use case where adult is using reader. When a child enters the park, they are not asked to provide a fingerprint for admission.
* Longevity Testing – The device is set to its test\_loop for hours or days at a time in order to reveal issues that could arise.
* The xTP is unaffected by architecture in the overall xConnect infrastructure such as HA and F5.

Breakdown of xTP Test Cases - The following are itemized in the xTP test cases spreadsheet:

|  |  |
| --- | --- |
| **Component** | **How tested** |
| RFID reader | API Tests & Manual |
| Media Interface | API Tests & Manual |
| Temperature | API Tests & Manual |
| Install / Upgrade | API Tests & Coordinated testing with xBRC |
| Events | API Tests & Manual – Check for taps & diagnostic messages |
| Software Tap | API Tests & Coordinated testing with xBRC |
| Configuration | API Tests where settings changed by API; restarting device and verifying results |
| Lumidigm Reader | API Tests using xBio calls |

### Resources Required

|  |  |
| --- | --- |
| **Resource** | **Description** |
| RFID – Desfire type | RFID card that contains a chip used to process RF read events. |
| RFID – Ultralight C type | RFID card that contains a chip used to process read events. |
| Provisioning tool | A hardware device used to assign IDs for RFID cards. |
| MagicBand | Wristband worn by guest that contains an RFID chip as well as radio hardware that supports long range reads. May be referred to as an xBand in older documents. These contain a Desfire RFID chip. |
| xTP reader | Required for Attraction model |
| xTP+xBio reader | Required for Park Entry model |

## Test Tools

|  |  |  |
| --- | --- | --- |
| **Tool** | **Description** | **Version** |
| Web Browser | Adhoc API testing | Any |
| RESTClient Firefox plugin | Full interactive API testing | 1.3.4 |
| Part\_1.py (Python script) | API Automation written using Python programming language. The script to run is named part\_1.py. It is included with other xConnect automation projects. | Requires python version 3 to run. |
| xTestReader (Java tests) | A project containing extended xTP automated tests. | Java 7 |
| FogBugz | Defect Tracking | Most recent |
| Excel | Test Scenarios & Cases | Most recent |

The automation written in Python is available in the xConnect software repositories.

The scripts exercise some of the REST interface APIs documented in the ICD. The set of cases represent Build Verification type cases. One thing that is checked is that each of the endpoints is reachable and returns http status 200 – Success. Other cases vary parameter combinations and check values returned for correctness. For REST calls that PUT data, then the test does a GET corresponding to the PUT to compare the data input to the data returned from the GET call.

|  |
| --- |
| **REST Endpoint** |
| GET /events |
| PUT /events |
| PUT /reader/name |
| GET /reader/info |
| GET /reader/loglevel |
| PUT /reader/loglevel |
| PUT /time |
| GET /time |
| GET /rfid/options |
| PUT /tap/options |
| GET /tap/options |
| GET /diagnostics |
| PUT /media/idle |
| GET /media/idle |
| PUT /media/sequence |
| GET /media/sequence |
| GET /xbrc |
| PUT /xbrc |

To run the automation script, enter “python part\_1.py” from the command line. An xTP reader should be reachable via the network. When prompted by the script, enter the IP address for the device. PASS and FAILURE status are shown in addition to the full rest call made for each test.

Sample output from reader automation (partial)

|  |
| --- |
| Enter IP Address of reader -->10.1.201.232  Testing common GET endpoints on ig88  PASS http://10.1.201.232:8080/reader/info  PASS http://10.1.201.232:8080/reader/name  PASS http://10.1.201.232:8080/reader/loglevel  PASS http://10.1.201.232:8080/diagnostics  PASS http://10.1.201.232:8080/events  PASS http://10.1.201.232:8080/time |

# The automation written in Java is also located in the xConnect Test project. The files there can be imported into any IDE where tests can be run interactively in a test runner such as those included in IntelliJ or Eclipse. It is also possible to run the tests from the command line, though this is not documented here.

To build from sources, maven is required to build the project. Maven is available from the Apache web site. You would run “mvn validate” from the command line in order to have maven pull in all of its dependencies. You could then run “mvn compile” from the command line to build the project outside of the IDE.

The following tests end points and tests are implemented now.

|  |  |
| --- | --- |
| API end point | Test Case |
| GET /diagnostics | getDiagnosticsSimple |
| GET /events | getEventsVerification |
| PUT /media/brightness | testSetMediaBrightness |
| GET /reader/info | testReaderInfoData |
| GET /reader/log | getReaderLogVerification |
| PUT /reader/name | setReaderNameToString |
|  | setReaderNameToNumber |
|  | setReaderNameToBlankString |
|  | setReaderNameToMultiWordString |
|  | setReaderNameToVeryLongString |
|  | setReaderNameNoParameters |
| PUT /rfid/options | testSetRfidOptions |
|  | testSetTapOptionsSequence |
|  | testSetTapOptionsSound |
|  | testSetTapOptionsTimeout |
| GET /update\_stream | getUpdateStreamSimple |

# Testing an xTP Device After Installation

## Purpose

Provide basic tests to verify that xTP reader is healthy upon initial installation.

This is a guide for a simple test to run when the device is installed in a park and not meant for testing builds of the software.

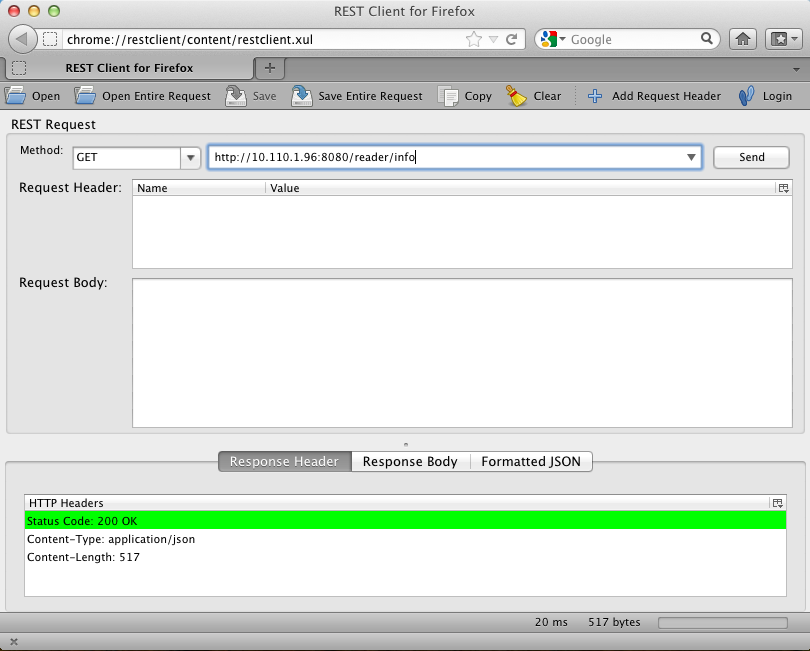
## Scope

This document covers basic manual physical tests as well as a set of API tests that can be run in a browser environment.

## Background

These tests are useful when a quick assessment of the xTP device’s functionality is desired.

## Assumptions and Constraints

* xTP hardware has been set up and accessible from a networked computer
* Firefox browser is available for use
* Plug-in named “REST Client for Firefox” installed for Firefox. Available at <https://addons.mozilla.org/en-US/firefox/addon/restclient/>  
  **Figure 5 REST Client for Firefox**  
    
  **How to use the REST client**
  1. Specify type of REST request in the Method field. Most tests use a **GET** request.
  2. Enter the URL in the field to right of Method field.
  3. Click **Send**.

The Response Header tab below should show “Status Code: 200 OK”. The Response Body tab will show output from API where applicable.

* This document is written from the perspective of someone running either Mac OX X or Linux. If you use Windows, then the ‘putty’ utility is used instead of ‘ssh’ to access the reader. More information about ‘putty’ is at: <http://www.chiark.greenend.org.uk/~sgtatham/putty/>

# Tests When Not Connected to xBRC

## Manual Physical Tests

Required items are a provisioned RFID card and (optionally) a provisioned MagicBand.

### Touch RFID Card to xTP

1. **Steps to** Turn on xTP.
2. Wait for boot up (LIGHTS and sound will occur when xTP is READY).
3. Touch an RFID for a second on the xTP.

**Expected result**

Lights and sounds occur.

### Touch MagicBand Card to xTP

1. **Steps to** Turn on xTP.
2. Wait for boot up (LIGHTS and sound will occur when xTP is READY).
3. Touch a MagicBand for a second on the xTP.

**Expected result**

Lights and sounds occur.

## API Tests

### Get Reader Info

1. **Steps to** Open Firefox and choose **REST Client** tool from Tools menu.
2. Specify **GET** method in Method field.
3. In next field enter “http://<xTP IP address>:<xTP port>/reader/info”
4. Click **Send**.

**Expected result**

1. HTTP Headers contain “Status Code: 200 OK”
2. Response Body shows output similar to below. Some of the values you see in your output may be blank.

{  
 "HW type" : "xTP2",  
 "RFID" : {  
 "SAM FW version" : "4.1",  
 "SAM mfg date" : "5/24/12",  
 "SAM sn" : "041C3BF2F92580",  
 "TDA8029 FW version" : "07 Release 1.0"  
 },  
 "ext hw" : "xTPX rev 1",  
 "ext sn" : "",  
 "linux version" : "Linux overo 2.6.39 #1 Fri Mar 8 11:47:58 PST 2013 armv7l unknown\n",  
 "mac" : "02:0A:80:00:02:3B",  
 "media hash" : "bcaceadfb8dcf572d6e220f4dd1f7302",  
 "min xbrc version" : "0.0.0.0",  
 "next eno" : 418,   
 "port" : 8080,   
 "reader name" : "ig88",  
 "reader type" : "xFP",  
 "reader version" : "2.5.7-3097 2013 Apr 05-09:22:28 - xfp-release (f49f8c92c91f532d86b5194a61597baf486a9e3a)",   
 "time" : "2013-04-17T00:07:59.547",  
 "update stream" : [  
 {  
 "url" : "http://10.110.1.58:8080/stream"  
 }  
 ],   
 "xbrc url" : "http://10.110.1.58:8080"  
}

### Verify That Events Are Generated

**Procedure**

1. Manually touch a provisioned card to xTP. You must know its PID and UID values.
2. Open Firefox and choose **REST Client** tool from Tools menu.
3. Specify **GET** method in Method field.
4. In next field enter “http://<xTP IP address>:<xTP port>/events”
5. Click **Send**.

**Expected result**

1. HTTP Headers contain “Status Code: 200 OK”
2. Response Body shows output similar to below, where pid and uid fields match actual values.

{  
 "events" : [  
 {  
 "eno" : 4668,   
 "pid" : "123456",  
 "time" : "2012-04-05T21:55:05.600",  
 "type" : "RFID",  
 "uid" : "F0F2C3C3F6B500"  
 }  
}

# Tests When Connected to xBRC

Resources required for these tests are an RFID card with a valid entitlement and a RFID card with an expired entitlement. In order to set these up, it will be necessary to know the PID, UID and SID for the RFID cards used. Then it will be possible to provision the cards in the IDMS and GxP systems. Once the cards are provisioned, then the following tests can be executed

## Manual Physical Tests

Required items are a provisioned RFID card and (optionally) a provisioned MagicBand.

### Touch RFID Card with Valid Entitlement

**Procedure**

1. Turn on xTP.
2. Wait for boot up (LIGHTS and sound will occur when xTP is READY).
3. Touch an RFID for a second on the xTP.

**Expected result**

Green light.

### Touch RFID Card with Expired Entitlement

**Procedure**

1. Turn on xTP.
2. Wait for boot up (LIGHTS and sound will occur when xTP is READY).
3. Touch an RFID for a second on the xTP.

**Expected result**

Blue light.

**Instructions for examining xTP reader log and looking for events generated by manual taps**

1. Call GET/reader/log from REST Client
2. Successfully search for F0F2C3C3F6B500 (a UID for an RFID device) in response