xBRC Park Entry Model

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Description** |
| 1.0 | 4/8/2013 | Arkady Glabek | Initial version |
| 1.1 | 4/12/2013 | Michael Lampi | Minor edits |
| 1.7 | 7/22/2013 | Glenn Curtis | Adding changes related to offline (no Keystone) mode support. |
|  |  |  |  |

**Document Approvers & Sign-Off**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Approver** | **Role** | **Document Accept/Reject** |
| 4/14/13 | Glenn Curtis | Development Manager | Accept |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1 Introduction 5](#_Toc353879015)

[2 Referenced Documents 5](#_Toc353879016)

[3 Park Entry System Overview 6](#_Toc353879017)

[4 Omni Ticket and Reader Management 7](#_Toc353879018)

[5 Guest Entitlement Processing 8](#_Toc353879019)

[6 Cast Member State 11](#_Toc353879020)

[7 High Availability 12](#_Toc353879021)

[7.1 TOR failover 13](#_Toc353879022)

[7.2 Park Entry State 13](#_Toc353879023)

[8 xGreeter Messages 16](#_Toc353879024)

[8.1 The /model/greeter/message endpoint 16](#_Toc353879025)

[8.1.1 The locationStatus message to the xBRC 20](#_Toc353879026)

[8.1.2 The locationStatusResponse back to xGreeter 21](#_Toc353879027)

[8.1.3 The openLocation message to the xBRC 21](#_Toc353879028)

[8.1.4 The openLocationResponse message back to xGreeter 22](#_Toc353879029)

[8.1.5 The closeLocation message to the xBRC 22](#_Toc353879030)

[8.1.6 The closeLocationResponse message back to xGreeter 22](#_Toc353879031)

[8.1.7 The bumpLocation message to the xBRC 23](#_Toc353879032)

[8.1.8 The bumpLocationResponse message back to xGreeter 24](#_Toc353879033)

[8.1.9 The scheduleMaintenance message to the xBRC 24](#_Toc353879034)

[8.1.10 The scheduleMaintenanceResponse message back to xGreeter 25](#_Toc353879035)

[8.1.11 The scheduleBioMaintenance message to the xBRC 25](#_Toc353879036)

[8.1.12 The scheduleBioMaintenanceResponse message back to xGreeter 26](#_Toc353879037)

[8.1.13 The shutdownReader message to xBRC 26](#_Toc353879038)

[8.1.14 The shutdownReaderResponse back to xGreeter 27](#_Toc353879039)

[8.1.15 The flashLocation message to the xBRC 27](#_Toc353879040)

[8.1.16 The flashLocationResonse message back to xGreeter 27](#_Toc353879041)

[8.1.17 The getLocationMap message to the xBRC 28](#_Toc353879042)

[8.1.18 The redirectReader message to the xBRC 28](#_Toc353879043)

[8.1.19 The redirectReaderResponse back to xGreeter 28](#_Toc353879044)

[8.1.20 The readerRetry message to the xBRC 28](#_Toc353879045)

[8.1.21 The subscribeToLocation message to the xBRC 29](#_Toc353879046)

[8.1.22 The subscribeToLocationResponse back to xGreeter 29](#_Toc353879047)

[8.1.23 The unsubscribeFromLocation message to the xBRC 29](#_Toc353879048)

[8.1.24 The unsubscribeFromLocationResponse back to xGreeter 29](#_Toc353879049)

[8.1.25 The processingError back to xGreeter 30](#_Toc353879050)

[8.2 The /model/greeter/subscribe endpoint 30](#_Toc353879051)

[8.2.1 The requestNotifications message to the xBRC 30](#_Toc353879052)

[8.2.2 The locationStateChange notification 31](#_Toc353879053)

[8.2.3 The readerEvent notification 32](#_Toc353879054)

[9 Omni Ticket Messages 32](#_Toc353879055)

[9.1 Header 32](#_Toc353879056)

[9.2 Request 34](#_Toc353879057)

[9.2.1 Login 34](#_Toc353879058)

[9.2.2 Connect 34](#_Toc353879059)

[9.2.3 Watchdog 34](#_Toc353879060)

[9.2.4 Entitlement 34](#_Toc353879061)

[9.3 Answer 35](#_Toc353879062)

[9.3.1 Login 36](#_Toc353879063)

[9.3.2 Connect 36](#_Toc353879064)

[9.3.3 Watchdog 36](#_Toc353879065)

[9.3.4 Entitlement 36](#_Toc353879066)

[10 Park Entry xBRC MySQL Database Tables 38](#_Toc353879067)

[10.1 The xBRC Tables 38](#_Toc353879068)

[10.2 Park Entry Model Core Tables 42](#_Toc353879069)

[10.2.1 PETransaction Table 43](#_Toc353879070)

[11 Park Entry Configuration parameters 45](#_Toc353879071)

# Introduction

This document describes the internal workings of the xBRC Park Entry Model. This includes system overview, description of reader management with respect to the Omni Ticketing System, high availability support, messaging protocols, database tables, configuration parameters and state diagrams showing internal states of the xBRC.

Two park entry model-specific communication protocols are described: a messaging protocol between the xBRC and the Omni Ticketing System (via TOR) and the messaging protocol between the xBRC and the xGreeter mobile application.

Please refer to the xBRC Interface Control Document (ICD) (#900-0058) document for detailed description of the xBRC REST interface that is not park entry model specific. This includes REST endpoints used between the DAP readers and the xBRC. The xBRC Interface Control Document (ICD) also describes the informational messages generated by the xBRC that can be used for guest monitoring (sent via HTTP or JMS).

# Referenced Documents

| Document Name & Version | Issuance Date | Relationship |
| --- | --- | --- |
| xBRC Interface Control Document (ICD) 900-0058 | May 1, 2012 | Description of all xBRC communication via HTTP and JMS. |
| xConnect HA Troubleshooting Guide 900-0173 | Dec 5, 2012 | Describes xBRC high availability architecture |
| Rev 2 Park Greeter Application Design | Aug 24, 2012 | Business use case scenarios for the xGreeter mobile application. |
|  |  |  |

# Park Entry System Overview

The following diagram shows the components of the park entry guest entitlement validation system. A series of DAP (touch point) readers equipped with biometric reader are controlled by the xBRC. The DAP readers are grouped in locations two per location. A greeter cast member opens a location using xGreeter mobile application allowing the readers at that location to start processing guest TAP events. The TAP events are sent to the xBRC which forwards them to the TOR for entitlement processing. The xBRC receives entitlement responses from TOR and signals a “success” or “failure” sequence to be displayed on the DAP reader.



Note: In xConnect 1.7, the xGreeter application would first locate the park specific xBRMS service to attempt to perform the cast member authentication against. This xBRMS service would then perform an authentication in Keystone on behalf of the user logon from the Greeter application UI. It would only fall back to attempting authentication against the xBRC UI master if for some reason there was an error accessing the xBRMS instance. The xBRMS service implements an offline logon cache for situations where the backend Keystone service is unavailable. The xBRMS logon cache is stored in the xBRMS MSSQL database and is shared across the two HA service nodes. The user must succeed with one recent online Keystone logon in order for the cache to have a proper cached credential. The offline credential cache will remember the list of roles that the user was part of such that cast member or coordinator functionality is preserved.

Figure 1 : Park Entry System

# Omni Ticket and Reader Management

The Omni Ticketing System (Omni) requires that each DAP reader be registered with Omni using a unique device ID. Furthermore, Omni requires that each DAP reader is “opened” for guest entitlement processing by sending a LOGON request with a username and password. Lastly, Omni requires that each DAP reader sends periodic Watchdog messages allowing the Omni system to monitor the health of a reader.

The xBRC acts as a proxy between the readers and the Omni system in that it sends XML messages to the Omni system on behalf of each reader. First the “Connect” message, then the “Login” message and then periodically the “Watchdog” message. Before the “Login” message can be sent a cast member must open the location associated with the readers.



Figure 2: Reader State Diagram

Please refer to section 8.1 Omni Ticket Messages for the format of the XML messages sent in the Reader State Diagram above.

Also, it is worth mentioning at this point that the state of each reader as depicted by the Reader State Diagram above is maintained by the xBRC in a separate thread for each reader. The XML messages are sent via a TCP/IP connection to TOR, one connection per reader.

# Guest Entitlement Processing

At the core of the park entry system is processing of guest entitlements. If a guest has a valid entitlement he can enter the park. Otherwise, the guest is directed away from the DAP reader for further assistance. The Omni Ticketing System is the repository of guest entitlements. The xBRC is in communication with the Omni Ticketing System allowing validation of entitlements and decrementing of the entitlement counter for each guest entering the park. The Omni Ticketing System also receives biometric template sent by the xBRC for each adult guest for enrollment or validation.

The xBRC maintains a short-lived state for each guest as the guest interacts with the DAP reader at park entry. The state is short-lived because it lasts a short time, around 4 seconds in a successful scenario. The guest state is serialized to the GST Mayhem database table on the xBRC computer. Although, it would be possible to restore the guest state between xBRC restarts or during failover to a secondary xBRC, this is not done due to the short lived nature of the guest state.

The following diagram represents the guest state as maintained by the xBRC. Above or below each arc connecting the states is shown the trigger action causing a transition to the next state. Below each trigger action, in parenthesis is the side effect of the transition, usually an action that the xBRC performs while moving to the next state.



Figure 3: Guest State Diagram

The following sequence diagrams show the system interaction during processing of guest entitlement, first a successful entitlement scenario.



Figure 4: Successful Entitlement Sequence Diagram

The following diagram shows processing of a failed entitlement. In this scenario a guest is shown the “blue” failure sequence on the reader following which the Greeter uses the xGreeter application to reset the “blue” light and instructs the guest to try again.



Figure 5: Failed Entitlement Sequence Diagram

# Cast Member State

Just as the xBRC maintains a state for each guest, it also maintains a very simple state for each cast member (greeter) who opens a location. Greeters use the xGreeter mobile application to interact with the xBRC. One of the xGreeter functions is the ability to open and close locations. When greeter opens a location, the xBRC sends the LOGON request to TOR for each reader at the location. The location is deemed open upon reception of at least a single positive response to the LOGON request. So, at least one reader must be logged on to TOR for a location to be open.

The cast member state currently consists of two states: WAITINGLOGONRESP and LOGGEDIN. This state is maintained for each greeter per each open or pending location. The state is persisted to the CMST Mayhem database table on the xBRC machine. The ability to restore the cast member state is essential during xBRC restart and during High Availability failover.

The following state diagram illustrates the cast member state transitions.



Figure 6: Cast Member State Diagram

# High Availability

The Park Entry xBRC high availability (HA) solution is achieved by the presence of a secondary (slave) xBRC. The TOR high availability is achieved by specifying fallback TOR servers per each DAP reader. The XBRMS high availability is achieved by the presence of multiple XBRMS servers any of which capable of processing request from the xGreeter app. The DAP reader high availability is achieved by the presence of multiple readers at each park entry location.

This document is mainly focused on the xBRC so what follows is the description of the xBRC high availability as well as the xBRC configuration allowing TOR high availability. The XBRMS and the DAP high availability details are beyond the scope of this document.

Please refer to the *xConnect HA Troubleshooting Guide* (#900-0173) document which describes in detail the overall high availability architecture of the xBRC.

In a nutshell, there is couple of xBRCs, a master and a slave. The master is the xBRC currently receiving events from the readers. The events are routed via an F5 load balancer configured to send all events to the active xBRC, rather than load balance between the two. All xBRC configuration changes are synchronized from the master to the slave. Any state necessary for continuous operation following a failover to the slave is synchronized from the master to the slave. The method of configuration synchronization is by a REST call from the slave to the master xBRC triggered by a JMS “DISCOVERY” message. The method of state synchronization is via JMS messages sent from the master to the slave xBRC. A slave becomes a master as soon as it receives a single communication (HTTP REST call) from a reader.

The diagram below illustrates the high availability setup for a master and a slave xBRC.



Figure 7: xBRC High Availability

## TOR failover

The xBRC maintains a list of TOR servers in its configuration. This list is not limited in size except by the availability of TOR servers. Each DAP is configured with a sub list of the TOR servers. Typically this consists of a primary and a secondary TOR, but more TORs may be configured per each reader. The xBRC connects to the first TOR in the list and will failover to the next TOR if the current one is not reachable or does not respond in time to a request.

It is worth mentioning that a TOR is limited for performance reasons to around 10 DAP readers. Thus there are couple of TOR servers for each 10 DAP readers (primary and secondary). Also, each DAP reader is identified in TOR by a unique device ID. Care must be taken when configuring the TOR servers for each reader to only include TORs that know the device ID for that reader.

## Park Entry State

As mentioned earlier, the xBRC synchronizes the state between the master and slave xBRC allowing for minimal disruption in the event of failover from master to slave. The Park Entry xBRC maintains three states: 1) state of the guest, 2) state of the DAP readers with respect to the TOR, and 3) state of the cast member who has open locations.

It turns out that for practical reasons only the state of the cast member is important to synchronize between the slave and the master. Let’s examine the design considerations that led to this decision.

**Guest State**

The guest state is usually very short-lived. Under normal circumstances is lasts for about 4 seconds. For failed entitlements it persists until the greeter clears the “blue” light on the DAP reader.

The following table shows the expected system behavior during a failover from master to slave xBRC. This table specifically looks at the guest state. Remember that the guest state is not synchronized to the slave xBRC so the slave xBRC does not know anything about the guest when it becomes the new master.

|  |  |
| --- | --- |
| Guest State at failover time | Expected system behavior after failover to the slave xBRC |
| TAPPED. Guest tapped on the reader. The reader has not shown a blue or green light yet. The reader may or may not be waiting for BIO scan at this time. | There will be no response from the reader. If the reader was waiting for the scan then the reader is reset. The expectation is that the guest will tap again seeing that there was no response to his first tap. Also, greeter cast member may instruct the guest to tap again.  Note: there is a possibility that the tap was from a child and the xBRC managed to send the entitlement request to TOR before stopping to function. The entitlement was decremented even though no green light was shown. In this case the second tap would produce a “RECENTLY USED” “blue” event. The guest would be directed away from the DAP reader where he will be assisted further. |
| WAITINGFORENROLL or WAITINGFORMATCH. The guest has tapped the reader and the reader is waiting for a bio scan. | There will be no response from the reader. The BIO scan ring on the reader will be reset. The expectation is that the guest will tap again. Failing that, the guest will be instructed by the greeter to try gain. |
| HASENTERED. Guest tapped on the reader, performed a scan if required and the entitlement has been decremented in Omni. | The assumption here is that the xBRC failed just before it was able to issue the “green” light on the reader. The chances of that happening are very small given the short time between the Omni response and issuing the “green” light on the reader. If that does happen then there will be no response from the reader. The guest will either tap again or be instructed to do so by the greeter. This will result in a “RECENTLY USED” “blue” lane event. The guest would be directed away from the DAP reader where he will be assisted further. |
| BLUELANE. Guest tapped on the reader, performed a scan if required and the entitlement failed. | The assumption here is that the xBRC failed just before it was able to issue the “blue” light on the reader.  The expectation is that the guest will tap again seeing that there was no response to his first tap. Also, greeter cast member may instruct the guest to tap again. This will result in another failed entitlement, but this time the reader will show the “blue” light. |

Figure 8: Guest State at Failover

A couple of the cases described in the table above result in a less than ideal guest experience. These are the cases where the entitlement has been decremented, but no “green” light was shown to the guest. Given the timing of the xBRC communication between TOR and the DAP reader the chances of these failure cases are small. On a positive note, because of the “RECENTLY USED” functionality on the TOR, the entitlement is not decremented twice, so no financial burden occurs to the guest.

While the guest state is currently not synchronized between the master and the slave, if the guest experience needs to be improved during the failover, then the necessary guest state synchronization is possible.

**Reader State**

The xBRC holds an open TCP/IP connection to TOR for each DAP reader. Readers that are in open locations also are “logged in” on the TOR. This is the reader state. The reader state itself is not synchronized from master to slave. However, the reader state can be re-created on the slave xBRC following failover based on the cast member state which is synchronized.

To re-create the reader state, the slave xBRC which becomes the new master first establishes TCP/IP connections for each DAP reader to TOR. Next, the xBRC examines the cast member state. For all locations that are open the xBRC will make sure that the DAP readers at these locations are in the “logged in” state on TOR. This is accomplished by first sending the “STATUS” message to TOR for each reader. If the “STATUS” response indicates that the DAP reader is “logged in” then no further action is required by the xBRC. Otherwise, the xBRC sends the “LOGON” request.

**Cast Member State**

The cast member state, indicating which locations are open, is synchronized on the slave xBRC using JMS messages. Three types of JMS messages are sent: 1) CASTOPENLOCATION, 2) CASTBUMPLOCATION and 3) CASTCLOSELOCTION. These JMS messages are received on the slave xBRC allowing the slave to open, bump and close locations to keep the cast member state in sync with the master.

# xGreeter Messages

The xGreeter is a mobile application designed to run on Apple iPod Touch device. The primary function of the xGreeter application is opening and closing DAP readers for guest traffic as well as monitoring guest entitlements and clearing readers in “blue lane” state.

The xGreeter application business functionality is documented in detail by a separate document Rev 2 Park Greeter Application Design.

The xBRC and the xGreeter application communicate via JSON messages sent over HTTP REST calls. The xGreeter makes REST calls to the xBRC. The xBRC never makes REST calls to the xGreeter application so it is for the xGreeter application to find the xBRCs it needs to talk to. This is accomplished by the xGreeter application making a /rest/facilities REST call to the XBRMS system (located via a Service DNS record) to obtain a list of xBRCs to talk to.

The primary focus for this section is to document the JSON messages sent between the xGreeter application and the xBRC.

The following message types are sent to the xBRC from the xGreeter application: locationStatus, openLocation, closeLocation, bumpLocation, scheduleMaintenance, scheduleBioMaintenance, shutdownReader, flashLocation, getLocationMap, readerRedirect, readerRetry, subscribeToLocation, unsubscribeFromLocation, requestNotifications. These messages are documented below along with their responses.

There are two REST endpoints exposed by the xBRC to the xGreeter application: 1) /model/greeter/message and 2) /model/greeter/subscribe. These are documented below.

## The /model/greeter/message endpoint

The /model/greeter/message endpoint is used to send all xGreeter messages to the xBRC to perform some action or query for information. The specific messages and their format are listed in the sections below. There are some fields that are repeated in all of the messages. These fields are listed the tables below.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| version | The xGreeter protocol version | 1.0 |
| type | The type of the message | locationStatus |
| clientReference | A text field sent by the xGreeter app and returned back to the xGreeter app in the response from the xBRC. | radar003:30:F7:C5:F3:1E:54,A2:F3:3E:BB:BA:13,32:F7:C5:F3:1E:56 |
| sessionId | The unique, randomly generated session Id generated by the xGreeter application when connecting to the xBRC. This is a logical session Id that is used in multiple REST calls to the xBRC. | bdef11a-26af-ae48-6ebf-53dccd52a80 |

Table 1: Common Message Fields

Many of the messages serialize the reader object. The table below shows the fields of a reader object.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| name | The name of the reader | V7-TP2 |
| state | The current state of the reader. The possible values are open, closed, shutdown. | open |
| lights | The state of the reader lights. The possible values are blue, green, off. | blue |
| lightDuration | The duration in milliseconds of the light shown on the reader, 0 for infinity. | 2500 |
| response | The response code if the reader object is returned in response to a command from the xGreeter. The possible values are success and error. | success |
| errorCode | In case of error response this field contains a short code for the error. The possible values are invalidRequestParameters, torConnectionFailure, readerHardwareProblem,  badOmniCastCreds,  readerNotResponding, readerIsDisabled. | readerNotResponding |
| errorDescription | Text description of the error. | Reader is not responding |
| tor | The machine name or IP address of the TOR server to which this reader is connected. | 10.93.212.212 |
| torDescription | The description of the TOR server. This is stored in the Mayhem database on the xBRC computer. | Primary TOR for V1 |

Table 2: Reader Object Fields

The table below shows the fields of a cast object. The cast object represents a cast member interacting with the xGreeter application.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| portalId | The portal Id of the cast member | Ridek005 |
| omniId | The Omni Id used to open the DAP reader on TOR. This field is currently not used by the xBRC. The xBRC uses single Omni Id for all readers. | 3994 |

Table 3: Cast Member Object Fields

The table below shows the fields of a guest object. The guest object is returned in a reader notification message notifying the xGreeter application about the result of guest entitlement.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| firstName | First name of the guest. For seasonal tickets this information is returned in the entitlement response from the TOR. | JOHN |
| lastName | Last name of the guest. | BROWN |
| idCheckRequired | Whether ID check is required. Returned by TOR. | true/false |
| bioRequired | Whether biometric scan is required. Returned by TOR. | true/false |
| celebrations | List of celebrations for the guest. This information can optionally be returned if using IDMS. |  |
| flashAtTurnstile | Whether to flash at turnstile. Returned by TOR. | true/false |

Table 4: Guest Object Fields

The table below shows the fields of a location object. A single location usually contains couple of readers.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| name | The name of the location | V1 |
| readers | A list of reader object at this location as defined in Table 2: Reader Object Fields above. | [{"state":"closed","lights":"off","name":"V8-TP2","tor":"10.88.1.190:9920","torDescription":""} … |
| response | The response code if the location object is returned in response to a command from the xGreeter. The possible values are success and error. | success |
| errorCode | In case of error response this field contains a short code for the error. The possible values are invalidRequestParameters, noReadersAtLocation,  noReadersConnectedToOmniAtLocation,  locationOccupied, allFlashColorsAreBusy. | noReadersAtLocation |
| errorDescription | Text description of the error. | There are no readers at this location. |
| cast | The cast member who currently has opened the location. | {"portalId":"Ridek005","omniId":"3994"} |
| byWhom | The cast member who performed a location operation such as bumping another cast member or closing a location. | {"portalId":"Ridek005","omniId":"3994"} |
| flashColor | This is the RGB value for a color that has been flashed on all readers at a location to identify it. | 24,124,239 |

Table 5: Location Object Fields

### The locationStatus message to the xBRC

This message is sent by the xGreeter application to inquire about the status of a location or number of locations.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| locations | This optional field specifies a list of locations for which to return the status. Status for all locations is returned if this field is not specified or if it is set to a list containing a single \* element: [“\*”]. | [“\*”] for all locations or [“V1”, “V2”] for two locations, etc. |

Example:

{

"version":"1.0",

"type":"locationStatus",

"clientReference":"DaviR163:30:F7:C5:F2:73:91,A6:6B:82:76:E7:18,32:F7:C5:F2:73:93",

"sessionId":"bdef11a-26af-ae48-6ebf-53dccd52a80"

}

### The locationStatusResponse back to xGreeter

This is a response to the locationStatus message.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| locations | List of locations containing readers and for opened locations cast members. | [{"name":"HS-V-7-Left","cast":{"portalId":"Ridek005","omniId":"3994"},"readers":[{"state":"open","lights":"off","name": ... |

Table 6: locationStatusResponse Message

Example:

{"version":"1.0","type":"locationStatusResponse","clientReference":"KERNS009:30:F7:C5:F0:C1:A0,C2:89:9F:87:51:B2,32:F7:C5:F0:C1:A2","locations":[{"name":"HS-V-8-Left","readers":[{"state":"closed","lights":"off","name":"V8-TP2","tor":"10.88.1.190:9920","torDescription":""},{"state":"closed","lights":"off","name":"V8-TP1","tor":"10.88.1.190:9920","torDescription":""}]},{"name":"HS-V-7-Left","cast":{"portalId":"Ridek005","omniId":"3994"},"readers":[{"state":"open","lights":"off","name":"V7-TP2","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"},{"state":"open","lights":"off","name":"V7-TP1","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"}]}

### The openLocation message to the xBRC

This message is sent to the xBRC to open a location.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| locationName | The name of the location to open. | HS-V-7-Left |
| cast | The cast member opening the location. | {"portalId":"Ridek005","omniId":"unknown","omniPassword":"unknown"} |

Table 7: openLocation Message

Example:

{"version":"1.0","type":"openLocation","clientReference":"Ridek005:8C:2D:AA:66:3F:E6,12:A9:BB:85:0A:C6,8E:2D:AA:66:3F:E8","sessionId":"1aff73d4-f8b6-bfa2-47ba-626d2bf6b8c0","cast":{"portalId":"Ridek005","omniId":"unknown","omniPassword":"unknown"},"locationName":"HS-V-7-Left"}

### The openLocationResponse message back to xGreeter

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| response | Response code: success or error. | success |
| cast | The cast member object that opened the location. | {"portalId":"Ridek005","omniId":"3994"} |
| readers | The list of reader objects at this location. | [{"state":"open","lights":"off","name":"V7-TP2","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"} … |

Table 8: openLocationResponse Message

Example:

{"version":"1.0","type":"openLocationResponse","clientReference":"Ridek005:8C:2D:AA:66:3F:E6,12:A9:BB:85:0A:C6,8E:2D:AA:66:3F:E8","location":{"name":"HS-V-7-Left","response":"success","cast":{"portalId":"Ridek005","omniId":"3994"},"readers":[{"state":"open","lights":"off","name":"V7-TP2","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"},{"state":"open","lights":"off","name":"V7-TP1","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"}]}}

### The closeLocation message to the xBRC

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| locationName | The name of the location to close. | HS-V-7-Left |
| cast | The cast member closing the location. | {"portalId":"Ridek005","omniId":"unknown","omniPassword":"unknown"} |

Table 9: closeLocation Message

Example:

{"version":"1.0","type":"closeLocation","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90","sessionId":"46f5d465-8eff-a338-7e0f-c4f9e247e14b","cast":{"portalId":"STASJ004","omniId":"unknown"},"locationName":"HS-V-8-Right"}

### The closeLocationResponse message back to xGreeter

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| response | Response code: success or error. | success |
| readers | The list of reader objects at this location. | [{"state":"open","lights":"off","name":"V7-TP2","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"} … |

Table 10: closeLocationResponse Message

Example:

{"version":"1.0","type":"closeLocationResponse","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90","location":{"name":"HS-V-8-Right","response":"success","readers":[{"state":"closed","lights":"off","name":"V8-TP3","tor":"10.88.1.190:9920","torDescription":""},{"state":"closed","lights":"off","name":"V8-TP4","tor":"10.88.1.190:9920","torDescription":""}]}}

### The bumpLocation message to the xBRC

This message is sent to the xBRC to open a location that is already opened by another cast member.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| locationName | The name of the location to bump. | HS-V-7-Left |
| cast | The cast member bumping the location. | {"portalId":"Ridek005","omniId":"unknown","omniPassword":"unknown"} |

Table 11: bumpLocation Message

Example:

{"version":"1.0","type":"bumpLocation","clientReference":"KERNS009:30:F7:C5:F0:C1:A0,0A:73:1C:75:A0:5F,32:F7:C5:F0:C1:A2","sessionId":"cb7dbb0-ce2a-ff60-471e-8db230da8b71","cast":{"portalId":"KERNS009","omniId":"unknown","omniPassword":"unknown"},"locationName":"HS-V-7-Left"}

### The bumpLocationResponse message back to xGreeter

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| response | Response code: success or error. | success |
| cast | The cast member object that bumped the location. | {"portalId":"Ridek005","omniId":"3994"} |
| readers | The list of reader objects at this location. | [{"state":"open","lights":"off","name":"V7-TP2","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"} … |

Table 12: bumpLocationResponse Message

Example:

{"version":"1.0","type":"bumpLocationResponse","clientReference":"KERNS009:30:F7:C5:F0:C1:A0,0A:73:1C:75:A0:5F,32:F7:C5:F0:C1:A2","location":{"name":"HS-V-7-Left","response":"success","cast":{"portalId":"KERNS009","omniId":"3994"},"readers":[{"state":"open","lights":"off","name":"V7-TP2","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"},{"state":"open","lights":"off","name":"V7-TP1","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"}]}}

### The scheduleMaintenance message to the xBRC

The scheduleMaintenance message is sent to the xBRC to schedule maintenance for a reader. Currently, the xBRC only logs this event in the log file and returns success back to the caller.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| cast | The cast member scheduling the maintenance. | {"portalId":"Ridek005","omniId":"3994"} |
| reader | The reader object for which to schedule maintenance. | {"name":"V7-TP2","maintenance":{“reason”: “Need to repaint the reader”}} |
| reader : name | The name of the reader. | V7-TP2 |
| reader : maintenance : reason | The maintenance reason. | Need to repaint the reader. |

Table 13: scheduleMaintenance message

Example:

{"version":"1.0","type":"scheduleMaintenance","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90","sessionId":"46f5d465-8eff-a338-7e0f-c4f9e247e14b","cast":{"portalId":"STASJ004","omniId":"unknown"}, “reader”: {"name":"V7-TP2","maintenance":{“reason”: “Need to repaint the reader”}}}

### The scheduleMaintenanceResponse message back to xGreeter

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| response | The response code: success or error. | success |
| reader | The reader for which maintenance was scheduled. | {"state":"open","lights":"off","name":"V7-TP2","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"} |

Table 14: scheduleMaintenanceResponse Message

Example:

{"version":"1.0","type":"scheduleMaintenanceResponse","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90, "response":"success","reader":{"state":"closed","lights":"off","name":"V8-TP3","tor":"10.88.1.190:9920","torDescription":""}}

### The scheduleBioMaintenance message to the xBRC

The scheduleBioMaintenance message is sent to the xBRC to schedule biometric maintenance for a reader. Currently, the xBRC only logs this event in the log file and returns success back to the caller.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| cast | The cast member scheduling the maintenance. | {"portalId":"Ridek005","omniId":"3994"} |
| reader | The reader object for which to schedule maintenance. | {"name":"V7-TP2","maintenance":{“reason”: “Need to repaint the reader”}} |
| reader : name | The name of the reader. | V7-TP2 |
| reader : maintenance : reason | The maintenance reason. | Need to repaint the reader. |

Table 15: scheduleBioMaintenance Message

Example:

{"version":"1.0","type":"scheduleBioMaintenance","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90","sessionId":"46f5d465-8eff-a338-7e0f-c4f9e247e14b","cast":{"portalId":"STASJ004","omniId":"unknown"}, “reader”: {"name":"V7-TP2","maintenance":{“reason”: “Need to repaint the reader”}}}

### The scheduleBioMaintenanceResponse message back to xGreeter

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| response | The response code: success or error. | success |
| reader | The reader for which maintenance was scheduled. | {"state":"open","lights":"off","name":"V7-TP2","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"} |

Table 16: scheduleBioMaintenance Message

Example:

{"version":"1.0","type":"scheduleBioMaintenanceResponse","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90, "response":"success","reader":{"state":"closed","lights":"off","name":"V8-TP3","tor":"10.88.1.190:9920","torDescription":""}}

### The shutdownReader message to xBRC

The shutdownReader message is used to disable a reader. This effectively takes the reader out of service. No events are processed for the reader while the reader is disabled. The xBRC UI may be used to enable the reader back in service.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| cast | The cast member shutting down the reader. | {"portalId":"Ridek005","omniId":"3994"} |
| reader | The reader object to shut down. | {"name":"V7-TP2","maintenance":{“reason”: “Need to repaint the reader”}} |
| reader : name | The name of the reader. | V7-TP2 |
| reader : shutdownReason | The maintenance reason. | Need to repaint the reader. |

Table 17: shutdownReader Message

Example:

{"version":"1.0","type":"shutdownReader","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90","sessionId":"46f5d465-8eff-a338-7e0f-c4f9e247e14b","cast":{"portalId":"STASJ004","omniId":"unknown"}, “reader”: {"name":"V7-TP2","shutdownReason": “Broken led”}}

### The shutdownReaderResponse back to xGreeter

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| response | The response code: success or error. | success |
| reader | The reader which was shutdown. | {"state":"shutdown","lights":"off","name":"V7-TP2”} |

Table 18: shutdownReaderResponse Message

Example:

{"version":"1.0","type":"shutdownReaderResponse","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90, "response":"success","reader":{"state":"shutdown","lights":"off","name":"V8-TP3"}}

### The flashLocation message to the xBRC

The flashLocation message instructs the xBRC to flash all readers at a location. The readers are flashed one of pre-configured colors. The chosen color is returned to the xGreeter application so that the greeter knows what color was flashed. This helps with more than one greeter flashing different locations at the same time.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| locationName | The name of the location to flash. | HS-V-7-Left |

Table 19: flashLocation Message

Example:

{"version":"1.0","type":"flashLocation","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90","sessionId":"46f5d465-8eff-a338-7e0f-c4f9e247e14b","locationName":"HS-V-8-Right"}

### The flashLocationResonse message back to xGreeter

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| response | Response code: success or error. | success |
| readers | The list of reader objects at this location. | [{"state":"closed","lights":"off","name":"V7-TP2","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"} … |
| flashColor | The RGB color that was flashed. | 30,55,39 |

Table 20: flashLocationResponse Message

Example:

{"version":"1.0","type":"flashLocationResponse","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90","location":{"name":"HS-V-8-Right","response":"success",”flashColor”:”44,33,44”,"readers":[{"state":"closed","lights":"off","name":"V8-TP3","tor":"10.88.1.190:9920","torDescription":""},{"state":"closed","lights":"off","name":"V8-TP4","tor":"10.88.1.190:9920","torDescription":""}]}}

### The getLocationMap message to the xBRC

This message is currently not implemented by the xBRC.

### The redirectReader message to the xBRC

The redirectReader message instructs the xBRC to clear the guest state as well as the reader state for a particular reader and redirect the guest away from the reader for further assistance. If a reader was showing a “blue” light then the light would be turned off.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| reader | The reader to reset. | {"name":"V7-TP2"} |

Table 21: redirectReader Message

Example:

{"version":"1.0","type":"redirectReader","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90","sessionId":"46f5d465-8eff-a338-7e0f-c4f9e247e14b, “reader”: {"name":"V7-TP2"}

### The redirectReaderResponse back to xGreeter

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| response | The response code: success or error. | success |
| reader | The reader which was reset. | {"state":"open","lights":"off","name":"V7-TP2”,"tor":"10.88.1.190:9920","torDescription":""} |

Table 22: redirectReaderResponse Message

Example:

{"version":"1.0","type":"redirectReaderResponse","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90, "response":"success","reader":{"state":"open","lights":"off","name":"V8-TP3","tor":"10.88.1.190:9920","torDescription":""}}

### The readerRetry message to the xBRC

The readerRetry message is identical to the redirectReader message except that the intent is for the guest to retry tapping on the reader rather than being redirected to away from the reader. The readerRetry message format is identical to the redirectReader message as is the readerRetryResponse message with the redirectReaderResponse message.

### The subscribeToLocation message to the xBRC

The subscribeToLocation message is sent by xGreeter app asking the xBRC to add the specified location to the list of locations for which the xBRC is sending reader notification messages (documented later on) to the xGreeter application.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| locationName | The name of the location for which to received reader notifications. | HS-V-8-Right |

Table 23: subscribeToLocation Message

Example:

{"version":"1.0","type":"subscribeToLocation","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90","sessionId":"46f5d465-8eff-a338-7e0f-c4f9e247e14b","locationName":"HS-V-8-Right"}

### The subscribeToLocationResponse back to xGreeter

The response only contains the common fields.

Example:

{"version":"1.0","type":"subscribeToLoctionResponse","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90”}

### The unsubscribeFromLocation message to the xBRC

The unsubscribeFromLocation message is sent by xGreeter app asking the xBRC to remove the specified location from the list of locations for which the xBRC is sending reader notification messages (documented later on) to the xGreeter application.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| locationName | The name of the location to remove from reader notifications. | HS-V-8-Right |

Table 24: unsubscribeFromLocation Message

Example:

{"version":"1.0","type":"unsubscribeFromLocation","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90","sessionId":"46f5d465-8eff-a338-7e0f-c4f9e247e14b","locationName":"HS-V-8-Right"}

### The unsubscribeFromLocationResponse back to xGreeter

The response only contains the common fields.

Example:

{"version":"1.0","type":"unsubscribeFromLoctionResponse","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90”}

### The processingError back to xGreeter

The processingError response message may be sent back to xGreeter in response to any of the requests. This message indicates that the xBRC was unable to process the request for some reason. For example, this error is returned when a request is received for a reader name that does exist.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| errorDescription | The description of the processing error. | Reader V2-3 not found. |

Table 25: processingError Message

Example:

{"version":"1.0","type":"processingError","clientReference":"STASJ004:F4:1B:A1:CD:C0:8E,D6:84:6A:A1:F9:C7,F6:1B:A1:CD:C0:90","sessionId":"46f5d465-8eff-a338-7e0f-c4f9e247e14b","errorDescription":"Reader V2-3 not found"}

## The /model/greeter/subscribe endpoint

The /model/greeter/subscribe HTTP endpoint is only used by xGreeter to subscribe to reader notification messages sent from the xBRC to the xGreeter application. The xGreeter can subscribe to receive notification messages for a single or multiple locations. The method of communication used is called “long polling” where the xGreeter makes a HTTP call to the xBRC (/model/greeter/subscribe) and the xBRC does not return a response back to the xGreeter until there is a notification or a list of notifications to be sent back. The xGreeter application makes another HTTP call to the xBRC after processing the notification.

### The requestNotifications message to the xBRC

The requestNotifications message is sent to the xBRC to establish a long poll connection to receive reader event notifications and location state change notifications.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| sessionId | The xGreeter session id for which to create a subscription. | bdef11a-26af-ae48-6ebf-53dccd52a80 |
| locations | A list of locations for which to receive notifications. Status for all locations is returned if set to a list containing a single \* element: [“\*”]. | [“\*”] or [“V1”, “V2”] |

Table 26: requestNotifications Message

Example:

{"version":"1.0","type":"requestNotifications","clientReference":"unknown:F4:1B:A1:CF:0E:F4,96:69:BA:7D:57:38,F6:1B:A1:CF:0E:F6","locations":["\*"],"sessionId":"b83cf646-4f47-d3cc-2d0e-908b62af52cc"}

### The locationStateChange notification

The locationStateChange notification is sent to the xGreeter informing the xGreeter that there was some state change for one or more readers at a location. For example, the location may have been opened and all readers changed state from closed to open.

The locationStateChange notification returns an array of messages rather than a single message. Each message contains the fields described in the following table.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| location | The location object with all of its readers for the location that has changed state. |  |

Table 27: locationStateChange Notification

Example:

[

{"version":"1.0","type":"locationStateChange","location":{"name":"HS-V-8-Left","readers":[{"state":"closed","lights":"off","name":"V8-TP2","tor":"10.88.1.190:9920","torDescription":""},{"state":"closed","lights":"off","name":"V8-TP1","tor":"10.88.1.190:9920","torDescription":""}]}},

{"version":"1.0","type":"locationStateChange","location":{"name":"HS-V-7-Left","readers":[{"state":"closed","lights":"off","name":"V7-TP2","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"},{"state":"closed","lights":"off","name":"V7-TP1","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"}]}},

{"version":"1.0","type":"locationStateChange","location":{"name":"HS-Solo-10-Right","readers":[{"state":"closed","lights":"off","name":"S10-TP2","tor":"10.88.1.176:9920","torDescription":""}]}},

{"version":"1.0","type":"locationStateChange","location":{"name":"HS-Solo-10-Left","readers":[{"state":"closed","lights":"off","name":"S10-TP1","tor":"10.88.1.190:9920","torDescription":""}]}},

{"version":"1.0","type":"locationStateChange","location":{"name":"HS-V-7-Right","readers":[{"state":"closed","lights":"off","name":"V7-TP4","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"},{"state":"closed","lights":"off","name":"V7-TP3","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"}]}},

{"version":"1.0","type":"locationStateChange","location":{"name":"HS-V-8-Right","readers":[{"state":"closed","lights":"off","name":"V8-TP3","tor":"10.88.1.190:9920","torDescription":""},{"state":"closed","lights":"off","name":"V8-TP4","tor":"10.88.1.190:9920","torDescription":""}]}}

]

### The readerEvent notification

The readerEvent notification is sent to xGreeter application whenever a guest entitlement is processed or the reader is reset. Essentially, it is sent every time the reader lights change from off to green or blue, or back to off.

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Sample Value |
| reader | The reader object that changed its state. | ":{"state":"open","lights":"green","lightDuration":2500,"name":"V7-TP1","errorDescription":"","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"} |
| guest | The guest object currently at the reader. Note that the guest object is only present if the readerEvent is sent in response to guest entitlement. If the readerEvent is sent because the reader was reset then no guest object is included. | {"firstName":"","lastName":"","idCheckRequired":false,"bioRequired":false,"celebrations":[],"flashAtTurnstile":false} |

Table 28: readerEvent Notification

Example:

[{"version":"1.0","type":"readerEvent","reader":{"state":"open","lights":"green","lightDuration":2500,"name":"V7-TP1","errorDescription":"","tor":"10.88.1.134:9920","torDescription":"WST-TOR002"},"guest":{"firstName":"","lastName":"","idCheckRequired":false,"bioRequired":false,"celebrations":[],"flashAtTurnstile":false}}]

# Omni Ticket Messages

The Omni Ticket messages are exchanged between the xBRC and TOR. They are used to connect and login readers to the TOR system. They are also used to check and decrement guest entitlements. These messages are exchanged using a TCP/IP connection. Each message is an XML document terminated by a 0 (NULL) character.

## Header

<Header>

<ReferenceNumber>reference number</ReferenceNumber>

<TransactionId>transaction id</TransactionId>

<RetryCounter>retries</RetryCounter>

<RequestType>request type</RequestType>

<RequestSubType/>

<InterfaceVersionID>version number</InterfaceVersionID>

<DeviceID/>

</Header>

|  |  |
| --- | --- |
| Element / Attribute | Purpose |
| ReferenceNumber | This is an xBRC generated number indicating the overall transaction. It should stay constant throughout the Guest Gate interaction. |
| TransactionNumber | This is the incremental transaction number. It should start at 1 with each new Guest Gate interaction and increment with each OmniTicket communication. This is used to match up requests and responses/ |
| RetryCounter | This starts with 0 and increments each time a transaction is repeated (i.e. when moving to the second TOR). |
| RequestType | Enumerated value since the Request Answer message set is coarse grained, this is used to distinguish transactions. The values are:  Login: Used to login the cast member  Connect: Used to introduce each xTP to the OmniTicket TOR. An xTP that is not Connected will not be allowed to admit guests.  Watchdog: Once connected, this message must be transmitted from each xTP periodically (once per minute, by default, set in the Connect Answer).  Entitlement: Used to decrement entitlements as bands are touched. This is the main transaction for the xBRC / OmniTicket interface. |
| RequestSubType | Used only with the Entitlement RequestType it indicates the type of interaction. The values are:  Update: This is the primary subtype, used whenever the xBRC is validating an entitlement and the OmniTicket system is matching Biometrics.  NoUpdate: This is an inquiry transaction and is not used by the xBRC.  NoMatch: Replaces the Update when the xBio is doing the Biometric matching. |
| InterfaceVersion | Set to V1.0 In future releases, this can be incremented to guarantee that the OmniTicket system can successfully process the messages being sent. The OmniTicket will reject any message version it cannot process. |
| DeviceID | 3 digit (000-254) identifying the xTP the transaction is coming from. |

Table 29: Omni message header

## Request

### Login

<Login>

<UserInfo>

<User>

<NumericId>BandID</NumericId>

</UserInfo>

</Login>

|  |  |
| --- | --- |
| Element / Attribute | Purpose |
| NumericId | This is the band id of the cast member logging in. It can come from any of the xTP in the location once the xBRC has been put into “login” mode. |

Table 30: Login request message

### Connect

<WorkRules>

<Tags>All</Tags>

</WorkRules>

<Connect>

<DeviceType>30</DeviceType>

<Status>1</Status>

</Connect>

|  |  |
| --- | --- |
| Element / Attribute | Purpose |
| Tags | The level of granularity in the response. Must be set to All |
| DeviceType | The type of turnstile. The xTP is type 30. |
| Status | The operational status of the device. 1 is operational. |

Table 31: Connect request message

### Watchdog

<Watchdog/>

There is no value; this tag with the Device ID in the header is enough.

### Entitlement

<WorkRules>

<Tags>All</Tags>

</WorkRules>

<Entitlement>

<Redeem>1</Redeem>

<BioValidation>1</BioValidation>

<MediaInfo>

<MediaSearchMode>

<xBandID>BandID</xBandID>

</MediaSearchMode>

<BioDeviceFilter>3</BioDeviceFilter>

<BiometricInfo>

<Item>1</Item>

<BioDeviceType>3</BioDeviceType>

<BiometricTemplate>template</BiometricTemplate>

</BiometricInfo>

</MediaInfo>

</Entitlement>

|  |  |
| --- | --- |
| Element / Attribute | Purpose |
| Tags | The level of granularity in the response. Must be set to All |
| Redeem | Request to redeem the entitlement if it exists, should always be 1 |
| BioValidation | Request to have OmniTicket perform Bio validation. Should be 1 when OmniTicket is doing validation, 0 when the xBio is doing the validation. |
| xBandID | The RFID that was read. This should NOT be the Guest ID used elsewhere in this document. |
| Item | The sequential number of the template being sent. This is only used on the second request when a biometric read is required, should always start with 1. There will only ever be 1 template, |
| BioDeviceFilter | This will always be set to 3 for the xBRC. |
| BiometricTemplate | The template data collected from the Bio device. This is only used on the second request when a biometric read is required |

Table 32: Entitlement request message

## Answer

<Error>

<ErrorCode>error code</ErrorCode>

<ErrorShortDescription>Short Desc.</ErrorShortDescription>

<ErrorDescription>Description</ErrorDescription>

</Error>

|  |  |
| --- | --- |
| Element / Attribute | Purpose |
| ErrorCode | This is the error code generated from the OmniTicket system. This will be 0 if there is no error. |
| ErrorShortDescription | This is the 100 character description of the error. |
| ErrorDescription | This is the full text of the error generated by OmniTicket. This tag will only be sent if there is an error. |

Table 33: Entitlement response message

### Login

The request data is echoed back in the Answer.

### Connect

<Connect>

<DeviceIP>IP address</DeviceIP>

<TorId>tor id</TorId>

<AccessAreaId>1</AccessAreaId>

<GroupID>1</GroupID>

<WatchDogTimeout>watchdog</WatchDogTimeout>

</Connect>

|  |  |
| --- | --- |
| Element / Attribute | Purpose |
| DeviceIP | The IP address of the TOR. |
| TorId | The identifier of the currently connected TOR |
| AccessArea | Always set to 1 for xBRC |
| GroupID | This is always set to 1 for xBRC. |
| WatchdogTimer | The amount of time in milliseconds the TOR is expecting to receive WatchDog messages. |

Table 34: Connect response message

### Watchdog

The Watchdog tag is simply echoed.

### Entitlement

<Entitlement>

<MediaInfo>

<xBandID>BandID</xBandID>

<VisualxBandID>printed band id</VisualxBandID>

</MediaInfo>

<EntitlementInfo>

<Decremented>0/1</Decremented>

<BioRequired>0/1</BioRequired>

<SimulateBio>0/1</SimulateBio>

<NewEnrollment>0/1</NewEnrollment>

<EntitlementConfiguration>

<InternalID/>

<ExternalID/>

<Name/>

<ExtendedDescriptions/>

<Text/>

<Biometric>

<BiometricLevel>level</BiometricLevel>

<IdCheck/>

</Biometric>

<FlashAtTurnstile/>

<CodeAtTurnstile/>

</EntitlementConfiguration>

<RemainingValue/>

<Validity/>

<Usages/>

<BiometricInfo>

<Item>item number</Item>

<DeviceType>3</DeviceType>

<MatchThreshold>resolution0</MatchThreshold>

<BiometricTemplate>template</BiometricTemplate>

</BiometricInfo>

<SeasonPassInfo/>

<ShowData/>

<TicketNote/>

<TicketAttribute/>

<AccessInfoList/>

</EntitlementInfo>

</Entitlement>

|  |  |
| --- | --- |
| Element / Attribute | Purpose |
| Decremented | Indicates whether the entitlement has been decremented. When this value is 1, the “green” light should be lit. |
| BioRequired | Indicates that a biometric template must be collected. When this value is 1 and Decremented is 0, the xBRC should ask the xTP for a biometric template. |
| SimulatedBio | Indicates whether a full biometric scan (0) or a beam break (1) is required. This data is passed to the xTP as part of the biometric request. |
| NewEnrollment | Indicates whether a template exists or not. When this value is 1, a new enrollment is required. This information should be passed to the xTP. |
| InternalID | This value is not used. |
| ExternalID | This value is not used. |
| Name | This value is not used. |
| ExtendedDescriptions | This value is not used. |
| Text | This value is not used. |
| BiometricLevel | The threshold for passing a matched template. This is only used when the xBio is doing the match. |
| IdCheck | This value is not used. |
| FlashAtTurnstile | This value is not used. |
| CodeAtTurmstile | This value is not used. |
| RemainingValue | This value is not used. |
| Validity | This value is not used. |
| Usages | This value is not used. |
| Item | This is the index to the list of Biometric templates. This value is only used when the xBio is performing the match. |
| DeviceType | This value is always 3 on the xBRC |
| MatchThreshold | The minimum match value. This is only used when the xBRC is performing the match. |
| BiometricTemplate | The template used for the match. This value is only used when the xBio is performing the match. |
| ShowData | This value is not used. |
| TicketNote | This value is not used. |
| AccessInfoList | This value is not used. |

Table 35: Watchdog message to TOR

# Park Entry xBRC MySQL Database Tables

The xBRC stores its configuration and state in the MySQL server Mayhem database running on the xBRC computer. There are number of tables dedicated to the configuration of the xBRC. These include the tables to store locations, readers, configuration parameters. There are also model specific tables that differ between the three models: park entry, attraction and space.

## The xBRC Tables

The tables described in this section are shared across all models. Since they are not park entry specific, only a brief explanation is given for each table.

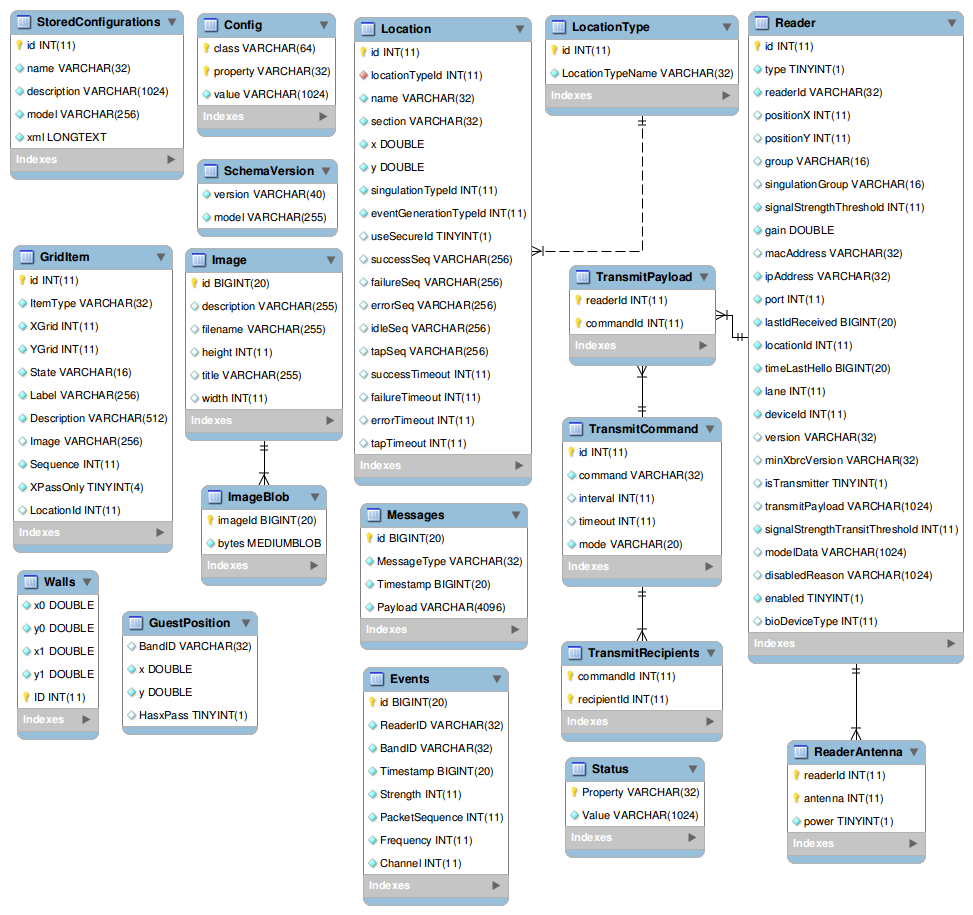


Figure : Core xBRC Database Tables

|  |  |
| --- | --- |
| Table Name | Description |
| General Configuration Tables | |
| Config | Various configuration parameters controlling many aspects of the xBRC operation, some model specific. |
| StoredConfigurations | Repository of XML formatted complete xBRC configurations. The stored configurations can be applied to a running xBRC to restore previously stored configuration including readers, locations and model specific configuration. The guest state is not stored in these configurations. |
| SchemaVersion | This table contains a single row with the current schema version of the Mayhem database. |
| Locations and Readers | |
| Location | The location of a given type and allowing a grouping of readers. |
| LocationType | The type of a location, for example: ENTRY, EXIT, WAYPOINT. |
| Reader | The reader of a given type. Contains various configuration parameters maintained per each reader. |
| ReaderAntenna | The power settings for each reader antenna. Only applies to long range readers. |
| TransmitPayload | Payload of the messages sent from the long range reader to the xBand. |
| TransmitCommand | Transmit command sent from the long range reader to the xBand. |
| TransmitRecipients | Recipient list for xBand commands. |
| Message Queue | |
| Messages | XML messages that are to be sent or were recently sent from the xBRC on the JMS bus or over HTTP. |
| Status | Status information for message queue. This includes the id of the last message that was sent over JMS as well as over HTTP. |
| Events | Not used at this time. |
| Guest Simulation | |
| GuestPosition | Position of the guest as he moves through a simulated attraction. |
| Walls | The walls of a simulated attraction. |
| GridItem | Represents a single grid item in a subway diagram representation of the attraction. |
| Image | Graphical images used in rendering of Facility View showing the subway diagram representation of the attraction. |
| ImageBlob | The actual image data of the graphical image. |

Table 36: xBRC general database tables

## Park Entry Model Core Tables

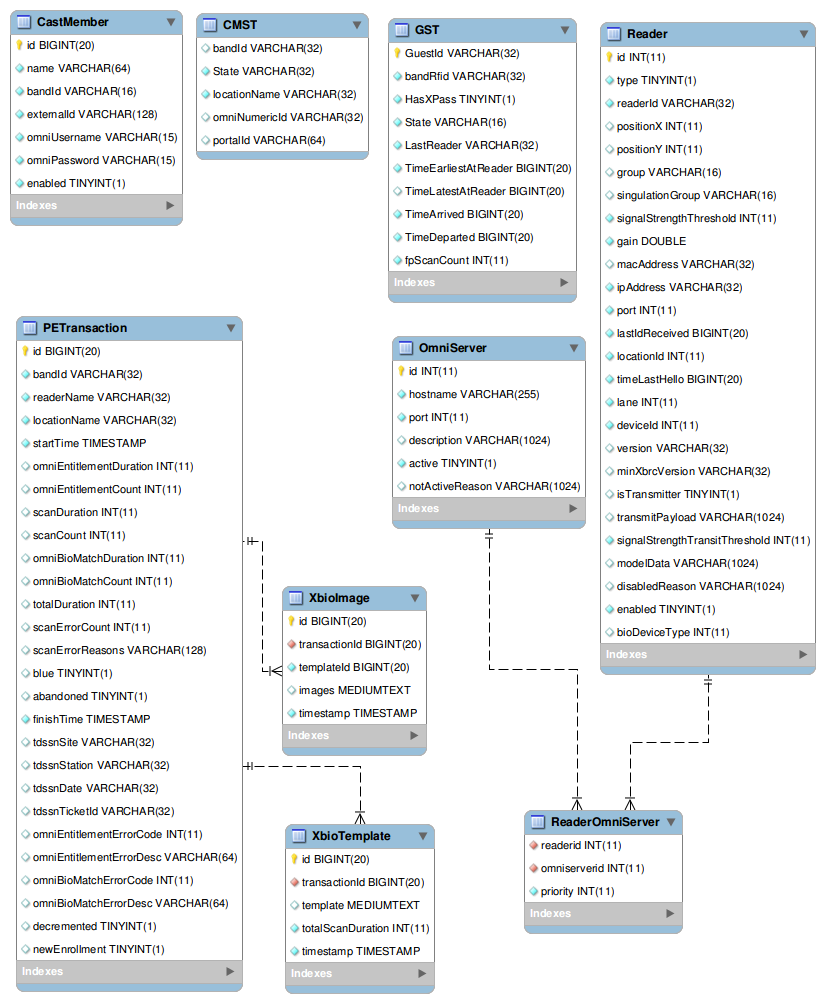


Figure 10: Park Entry Core Tables

|  |  |
| --- | --- |
| Table Name | Description |
| Reader Configuration | |
| Reader | Table storing DAP and long range readers. |
| OmniServer | Omni Server (TOR) |
| ReaderOmniServer | Join table connecting Omni Servers and Readers |
| Guest Status | |
| GST | Guest status for each guest at park entry. |
| Cast Member Management | |
| CastMember | Table storing a card or band for a cast member that can be used to clear readers by tapping it on the DAP reader. |
| CMST | Cast member state. This table keeps the state of the cast member as he opens a location. |
| Reporting | |
| PETransaction | Stores information about a single guest transaction from the tap to green or blue light shown on the reader. |
| XbioTemplate | List of short biometric templates generated by the guest during a single guest transaction. The same template is sent to TOR for enrollment or bio match. |
| XbioImage | Large binary data (around 3M each record) generated for each guest during biometric scan. This data is stored for diagnostic purposes. |

Table 37: Park entry model database tables

### PETransaction Table

The following table describes in detail the PETransaction table.

|  |  |
| --- | --- |
| **PETransaction Table** | |
| **Column** | **Description** |
| id | Database generated serial record id |
| bandId | The public id of the guest band or ticket tapped at the reader |
| readerName | The reader name tapped by the guest |
| locationName | The location name where the reader is located |
| startTime | The time when the reader was tapped for the first time by the guest |
| omniEntitlementDuration | Time in milliseconds from the time guest entitlement was queued to be sent to TOR and the time it was received from TOR. This roughly corresponds to the time it takes to transmit the entitlement to TOR, TOR processing time, and time it takes to receive the entitlement from TOR. Some xBRC parsing overhead is also included in this time. |
| omniEntitlementCount | If for any reason more than one entitlement request was sent to TOR during single guest transaction then this field will contain the number of entitlements sent. In this case the omniEntitlementDuration will be the sum of the time it took to send and receive all entitlements. |
| scanDuration | Time in milliseconds from the time the xBRC asked the reader for a biometric scan and the time the biometric event was received by the xBRC from the reader. |
| scanCount | In the event of multiple biometric scan attempts (requested by the xBRC, not the reader) this will contain the number of biometric scans requested. In this case the scanDuration will be the sum of all biometric scan times. |
| omniBioMatchDuration | The time in milliseconds from the time guest biometric match request was queued to be sent to TOR and the time it was received from TOR. As with omniEntitlementDuration some xBRC parsing overhead is included in this time. |
| omniBioMatchCount | The number of biometric match requests sent to TOR. If more than one then omniBioMatchDuration will contain the sum time of all biometric match requests. |
| totalDuration | The time in milliseconds from the guest tap to the time the guest was shown a green or blue light on the reader. |
| scanErrorCount | The DAP reader notifies the xBRC of every failed biometric scan. This is the count of failed biometric scans for a guest for a single guest transaction. |
| scanErrorReasons | Comma separated list of scan error reasons. |
| blue | Boolean field. Set to true if the guest was shown a “blue” light. |
| abandoned | Boolean field. Set to true if the guest abandoned the reader. |
| finishTime | The time when the guest was shown blue or green light on the reader. |
| tdssnSite | Omni specific field. Received from TOR as part of entitlement response. |
| tdssnStation | Omni specific field. Received from TOR as part of entitlement response. |
| tdssnDate | Omni specific field. Received from TOR as part of entitlement response. |
| tdssnTicketId | Omni specific field. Received from TOR as part of entitlement response. |
| omniEntitlementErrorCode | In case of failed entitlement this field contains the textual error code returned from TOR. |
| omniEntitlementErrorDesc | In case of failed entitlement this field contains the error description returned from TOR. |
| omniBioMatchErrorCode | In case of failed biometric match this field contains the textual error code returned from TOR. |
| omniBioMatchErrorDesc | In case of failed biometric match this field contains the error description returned from TOR. |
| decremented | Boolean field. True if the entitlement count was decremented on TOR for a guest. |
| newEnrollment | Boolean field. True if the guest did a new enrollment on TOR. |

Table 38: PETransaction table columns

# Park Entry Configuration parameters

The following table describes the park entry specific configuration parameters that are stored in the Mayhem Config table. Each property stored in the Config table has the class column set to “ParkEntryModelConfig”.

|  |  |  |
| --- | --- | --- |
| Property | Description | Sample Value |
| abandonmenttimesec | Number of seconds to wait for biometric scan since guest tap on the reader. If biometric scan is not received then the xBRC will mark the guest as having abandoned the reader. | 25 |
| castappcorethreadpoolsize | The starting number of threads to reserve for sending notification messages to xGreeter application. | 10 |
| castappmaxthreadpoolsize | The maximum number of threads to use for sending notification messages to xGreeter applications. | 50 |
| castappnotifytimeoutsec | Time in seconds to queue notification messages to xGreeter applications that are about to make another long polling notification request. | 5 |
| castloginokdurationms | Not used. To be removed. |  |
| castloginoklight | Not used. To be removed. |  |
| castlogontimeoutsec | Not used. To be removed. |  |
| flashcolors | Pipe separated list of RGB color values for possible choices of reader flash lights. | #ba2525|#c39c00|#009f50|#00509f|#a128a1 |
| greenlighttimeoutms | The time in milliseconds to display the green light icon on the xGreeter application in response to positive guest entitlement. | 5000 |
| guestretaptimeoutms | The time in millisecond to ignore guest tap on the reader following a green light shown to the guest. | 1000 |
| maxfpscanretry | Not used. To be removed. |  |
| omniconnecttimeoutms | The time in milliseconds between subsequent connection attempts to a TOR server. | 4000 |
| omniid | The Omni user ID to use to logon all readers to TOR. | 020303030 |
| omnipassword | The Omni user password to logon all readers to TOR. |  |
| omnirequesttimeoutms | The time in milliseconds after which the xBRC times out any message sent to TOR for which there was no response. The xBRC will try twice to send a timed out message. Following this, it will try switching to a secondary TOR. | 15000 |
| omniticketport | The TCP/IP port on which to connect to TOR. | 9920 |
| readerconnecttimeoutms | Reader HTTP connection time out in milliseconds when talking to readers. | 2000 |
| readerflashtimems | The duration in milliseconds to flash the reader light when requested by the xGreeter application. | 3000 |
| savebioimages | Whether to save the 3 MB biometric diagnostic images for each guest. Possible values are: all – all images, none – no images saved, failed – only when the guest was shown a blue light. | none (or all or failed) |
| savebioimagesfrequency | Whether to (1) save every biometric image, or (2) every other one, or (3) every third one, etc. This is done to allow for saving sample biometric images thus reducing storage requirements. | 1 |
| startscandurationms | The duration in milliseconds to show the biometric light on the reader to prompt the user for biometric scan. The value of 0 instructs to reader to use the time set in the sequence stored on the reader. | 0 |
| startscanlight | The sequence name sent to the reader using the /media/sequence REST endpoint to initiate the biometric scan. | entry\_scan\_start |
| testmode | Boolean property. When set to true the guest TAP is processed even though the location is not open. This works only when talking to a simulated TOR since if the location is not open then readers are not logged in to TOR and the TOR will reject all entitlement requests. | true/false |

Table 39: Park entry configuration parameters