xBRC Interface Control Document

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Description** |
| 1.0 | 5/1/2012 | Manny Vellon | Release Version |
| 1.4 | 9/30/2012 | Glenn Curtis | xConnect 1.4 refresh |
| 1.5 | 11/5/2012 | Glenn Curtis | xConnect 1.5 refresh |
| 1.5 | 11/14/2012 | Manny Vellon | xConnect 1.5 refresh (part 2) |
| 1.5 | 11/28/2012 | Michael Lampi | Update reader packages, fix typos |
| 1.5 | 01/16/2013 | Arkady Glabek | Added avmshello and heartbeat |
| 1.5 | 02/05/2013 | Manny Vellon | Add note regarding multiple xBRCs |
| 1.5 | 02/12/2013 | Glenn Curtis | Added LOCATIONABANDON message |
| 1.5 | 02/28/2013 | Arkady Glabek | Changed AVMS to VID |
| 1.6 | 03/01/2013 | Scott Salley | xConnect 1.6 refreshed with xTable 1.1 |
| 1.6 | 03/07/2013 | Glenn Curtis | Indicated plans to deprecate READEREVENT and Space model ABANDON messages |
| 1.6 | 03/07/2013 | Glenn Curtis | Added publicid to PE message formats |
| 1.7 | 05/15/2013 | Arkady Glabek | Added transmitreaderstatus REST endpoint |
| 1.7 | 05/17/2013 | Glenn Curtis | Fully deprecated READEREVENT and Space mode ABANDON messages.  Space model considerations now renamed to address HTTP Update Stream considerations.  LOCATIONEVENT and LOCATIONABANDON defined identically for both Space and Attraction models. |
| 1.7 | 05/17/2013 | Glenn Curtis | Clean up missing tags in LOCATIONEVENT message confidence and rfid. Removed readerlocationid and readerid tags from all messages. |
| 1.7 | 05/17/2013 | Glenn Curtis | Added PARKENTRYEVENT, PARKENTRYABANDONED, and BANDSTATUS messages for various LRR read scenarios. |
| 1.7 | 06/12/2013 | Glenn Curtis | Added clarification that use of readerlocation name values would enforce uniqueness across multiple defined locations within a given xBRC configuration. This is enforced during the creation of locations. |
| 1.7 | 07/18/2013 | Glenn Curtis | Added detail about how various link identifier types can be supported returned in xBRC messages, such as fidelio-link-id, gxp-link-id, and default xbms-link-id. |
| 1.7 | 7/18/2013 | Glenn Curtis | Added draft specification for WITHCHARACTER message format. This message would be implemented in future xBRC version with proposed Character-Band Association feature (ETA version 1.8.0). |
| 1.7 | 9/5/2013 | Arkady Glabek | Added specification for the /vidhello SYSTEMHELLO message. |
| 1.7.2 | 9/5/2013 | Michael Lampi | Clarify /vidhello specification, minor updates elsewhere |
| 1.7.2 | 9/9/2013 | Glenn Curtis | Removed readersection tag from xBRC messages and replaced with locationid tag. Added missing readername tag where it was occurring in actual xBRC messages in the solution implementation.  Removed from documentation of LOCATIONEVENT message the readerlocationid tag that was previously deprecated. |

**Document Approvers & Sign-Off**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Approver** | **Role** | **Document Accept/Reject** |
| 5/1/2012 | Ric Merrifield | Release Manager | Accept |
| 5/11/2012 | John Stiehl | Release Manager | Accept |
|  |  |  |  |

**Table of Contents**

[1 Introduction 7](#_Toc366144987)

[1.1 Purpose 7](#_Toc366144988)

[2 Referenced Documents 8](#_Toc366144989)

[3 xBRC Interfaces 9](#_Toc366144990)

[3.1 Reader Interface 9](#_Toc366144991)

[3.2 Control Interface 10](#_Toc366144992)

[3.3 Messaging Interface 10](#_Toc366144993)

[4 Control Interface 11](#_Toc366144994)

[4.1 GET status 13](#_Toc366144995)

[4.2 GET heartbeat 15](#_Toc366144996)

[4.3 GET messages 15](#_Toc366144997)

[4.4 PUT updatestream 16](#_Toc366144998)

[4.5 DELETE updatestream 17](#_Toc366144999)

[4.6 GET gueststatus 17](#_Toc366145000)

[4.7 DELETE gueststatus 18](#_Toc366145001)

[4.8 GET readerlocationinfo 18](#_Toc366145002)

[4.9 GET readerstats/reader/<reader name> 25](#_Toc366145003)

[4.10 DELETE readerstats 26](#_Toc366145004)

[4.11 GET bandevents/band/<long-range bandid> 26](#_Toc366145005)

[4.12 PUT storeconfiguration 27](#_Toc366145006)

[4.13 GET configurations 28](#_Toc366145007)

[4.14 GET currentconfiguration 29](#_Toc366145008)

[4.15 POST configuration?name=*name*&description=*description* 31](#_Toc366145009)

[4.16 DELETE configuration/name/<name> 31](#_Toc366145010)

[DELETE configuration/id/<id> 31](#_Toc366145011)

[4.17 PUT selectconfiguration/name/<name> 31](#_Toc366145012)

[4.18 GET ekgposition 32](#_Toc366145013)

[4.19 GET ekg 32](#_Toc366145014)

[4.20 PUT logcomment?text=<string> 32](#_Toc366145015)

[4.21 PUT updateconfig 32](#_Toc366145016)

[4.22 DELETE cache 33](#_Toc366145017)

[4.22.1 DELETE cache 33](#_Toc366145018)

[4.22.2 DELETE cache/guest/id/<guest-id> 33](#_Toc366145019)

[4.23 DELETE cache/band/secureid/<secure-id> 33](#_Toc366145020)

[DELETE cache/band/lrid/<long-range-id> 33](#_Toc366145021)

[DELETE cache/band/rfid/<rf-id> 33](#_Toc366145022)

[4.24 GET refreshpackages 33](#_Toc366145023)

[4.25 PUT videvent 33](#_Toc366145024)

[4.26 PUT vidhello 33](#_Toc366145025)

[4.27 GET perfmetricsmetadata 35](#_Toc366145026)

[4.28 PUT mediapackage 35](#_Toc366145027)

[4.29 DELETE mediapackage 36](#_Toc366145028)

[4.30 PUT playsequence/<readername>?sequencename=<sequencename>&timeout=<timeoutlength> 36](#_Toc366145029)

[4.31 GET sequences 36](#_Toc366145030)

[4.32 PUT scheduleoverrideon?minutes=<minutes> 37](#_Toc366145031)

[4.33 GET scheduleoverrideclear 37](#_Toc366145032)

[4.34 GET transmitreaderstatus 37](#_Toc366145033)

[5 Messaging Interface 39](#_Toc366145034)

[5.1 xBRC Models 39](#_Toc366145035)

[5.2 High Level Description of xBRC Messages 39](#_Toc366145036)

[5.3 General XML Payload Format 41](#_Toc366145037)

[5.4 Specific Message Payloads 41](#_Toc366145038)

[5.4.1 ENTRY Message 41](#_Toc366145039)

[5.4.2 MERGE Message 42](#_Toc366145040)

[5.4.3 LOAD Message 44](#_Toc366145041)

[5.4.4 INVEHICLE Message 45](#_Toc366145042)

[5.4.5 WITHCHARACTER Message (Draft – planned future feature in 1.8) 48](#_Toc366145043)

[5.4.6 EXIT Message 50](#_Toc366145044)

[5.4.7 ABANDON Message 52](#_Toc366145045)

[5.4.8 METRICS Message 53](#_Toc366145046)

[5.4.9 TAPPED Message 55](#_Toc366145047)

[5.4.10 ABANDONED Message 56](#_Toc366145048)

[5.4.11 BLUELANE Message 57](#_Toc366145049)

[5.4.12 HASENTERED Message 58](#_Toc366145050)

[5.4.13 PARKENTRYEVENT Message 58](#_Toc366145051)

[5.4.14 PARKENTRYABANDON Message 59](#_Toc366145052)

[5.4.15 BANDSTATUS Message 60](#_Toc366145053)

[5.4.16 LOCATIONEVENT Message 62](#_Toc366145054)

[5.4.17 LOCATIONABANDON Message 64](#_Toc366145055)

[5.4.18 DISCOVERY Message 66](#_Toc366145056)

[5.5 JMS Transport Details 67](#_Toc366145057)

[5.6 Using Multiple xBRCs at a Facility 69](#_Toc366145058)

[6 HTTP Update Stream use case scenarios 69](#_Toc366145059)

[6.1 Sequence Diagrams 70](#_Toc366145060)

[7 Park Entry 72](#_Toc366145061)

[7.1 GXP Interaction 73](#_Toc366145062)

[7.2 xBRC-to-GXP xPass Redemption Interaction 75](#_Toc366145063)

[7.3 GXP-to-xBRC Light Control 76](#_Toc366145064)

[7.4 Reconfiguring xTP Readers 77](#_Toc366145065)

[7.4.1 Get configuration 77](#_Toc366145066)

[7.4.1 PUT selectconfiguration/name/<name> 77](#_Toc366145067)

[8 Reader Registration 79](#_Toc366145068)

[8.1 Automatic Reader Software Updates 82](#_Toc366145069)

[8.1.1 How does the xBRC choose which version to install 82](#_Toc366145070)

[8.1.2 Minimum xBRC version 82](#_Toc366145071)

[8.1.3 How does the xBRC choose which install method to use 82](#_Toc366145072)

[8.1.4 Uploading new reader packages to the xBRC 83](#_Toc366145073)

[9 Scheduling 83](#_Toc366145074)

# Introduction

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

## Purpose

The xBRC Interface Control Document describes how the xBRC communicates with other component of the Disney Next Generation Experience (NGE). Three use cases are explored: “Attraction”, “Park Entry”, and “Space” (formerly called “Cruise Ship”).

From the perspective of other components that interact with the xBRC, this document serves as a documentation of the xBRC application programming interface (API).

The xBRC is the main conduit for xBand *events* to be communicated to downstream applications that need to act on those events. An xBand event begins with a park guest nearing an xBR long-range reader or touching an xTP reader. The xBRC processes events from multiple readers and then generates messages to downstream components listening for these events. Two message transport mechanisms are supported:

* The SonicMQ Java Messaging System (JMS) to communicate using a publish/subscribe model. Applications interested in messages from an xBRC can communicate with Sonic MQ and *subscribe* to the xBRCs message stream (the JMS *topic*).
* Alternatively, the xBRC can communicate messages using HTTP.

Each of the xBRC use cases has an xBRC “model” that is associated with it. The models are named:

*com.disney.xband.xbrc.attractionmodel.CEP*

*com.disney.xband.xbrc.parkentrymodel.CEP*

*com.disney.xband.xbrc.spacemodel.CEP*

A model is a plug-in “driver” that customizes the behavior of the xBRC for a particular use-case. For example, the Attraction model knows how to communicate with the GXP FastPass+ system and how to calculate various queue wait times as a guest proceeds from entry to exit, while the Park Entry model knows how to interact with the Omni ticketing system and how to handle the acquisition and verification of biometric data. The Space model is much simpler - for the most part, it simply reports the location of a guest every time the location changes.

The model in use by an xBRC is determined by the software package (RPM) that was used to install it. The package contains scripts to initialize the xBRC with the appropriate model-specific configuration parameters.

This document will present information as generically as possible. Where necessary, however, it will identify how the xBRC’s operation differs depending on the selected model.

# Referenced Documents

| Document Name & Version | Issuance Date | Relationship |
| --- | --- | --- |
| Reader Interface Control Document (ICD) | July 29, 2011 | Documents HTTP RESTful protocol between readers and XBRC |
| xBRC High-Level Technical Design | August 8, 2011 | Describes overall xBRC architecture |
| xBand Reader Management System System Architecture Document | July 21, 2011 | Describes overall xBRMS architecture |
| xBand\_RF\_ICD.doc | Feb 22, 2012 | Describes the wireless communications protocol to the xBand intended for use between the xBand and the long range reader or xBR. |

# xBRC Interfaces

The xBRC interfaces with various components

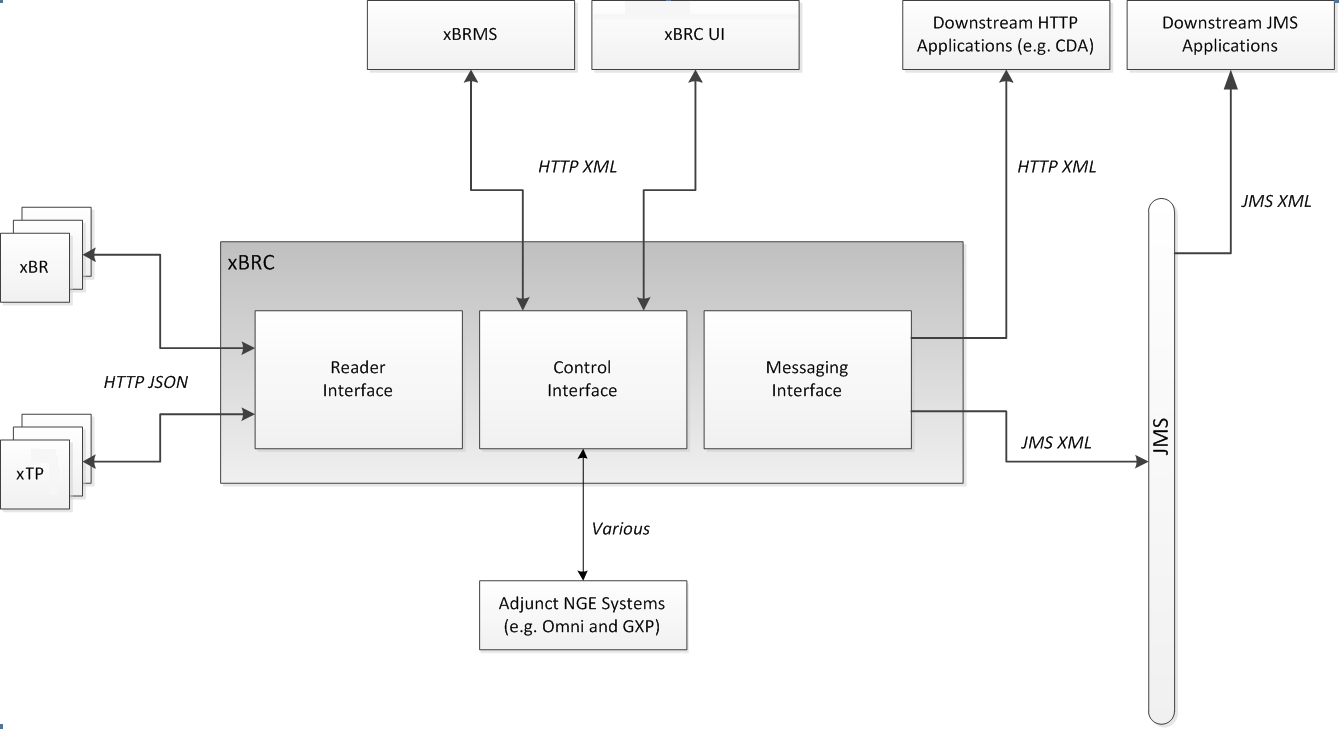


Figure . xBRC Interfaces

## Reader Interface

The xBRC reader interface supports communication with long-range and touch readers. These readers are configured to initiate communication with a specific xBRC when they start up. The initiation message consists of an HTTP PUT hello message; in other words, the readers perform an HTTP POST request to an xBRC’s web service port (8080) using the request path “hello”. The data payload of the hello message is a JSON object describing the reader. The details of this message are documented in the xBR and xTP Interface Control Documents.

Once a reader has initiated a dialog with an xBRC, the xBRC will typically perform various HTTP operations to properly configure the reader. These operations typically consist of HTTP POST operations to set the reader’s clock, to set its operating mode and to rename it if necessary.The xBRC may also inform the reader that it has new software available for it. Finally, the xBRC puts the reader in “push mode” telling it to send it events via a particular URL. When a reader is in push mode, it will periodically (as specified by parameters) send any band events to the xBRC. The details of these messages are contained in the respective reader interface control documents.

## Control Interface

The xBRC control interface consists of a set of HTTP requests (GET, PUT, POST and DELETE operations) that are used to query, configure and control the operation of an xBRC. These requests are detailed later in this document (section 4).

The xBRC control interface also includes communication with other related NGE systems. For example, in the Attraction use case, the xBRC needs to communicate with GXP in order to validate and redeem Fastpass+ entitlements. In the Park Entry use case, the xBRC needs to communicate with the Omni ticketing system to validate entry entitlements and biometric data. The details of these interfaces are not described in this document. For more information, refer to documentation for these adjunct systems.

## Messaging Interface

The primary role of the xBRC is to interpret high-traffic, low-level, reader events and to translate them into low-traffic, more useful, messages. Much of this document consists of descriptions of the types of messages generated by the xBRC.

# Control Interface

The xBRC implements the following HTTP requests (listed alphabetically, but described in a more logical order):

|  |  |
| --- | --- |
| Path | Operation |
| bandevents/band | GET |
| cache | DELETE |
| configuration | GET, POST, PUT, DELETE |
| configurations | GET |
| currentconfiguration | GET |
| ekg | GET |
| ekgposition | GET |
| gueststatus | GET, DELETE |
| heartbeat | GET |
| logcomment | PUT |
| messages | GET, DELETE |
| mediapackage | PUT, DELETE |
| playsequence | PUT |
| readerlocationinfo | GET |
| readerstats | GET, DELETE |
| refreshpackages | GET |
| scheduleoverrideon | PUT |
| Scheduleoverrideclear | GET |
| selectconfiguration | PUT |
| sequences | GET |
| status | GET |
| storeconfiguration | PUT |
| updateconfig | PUT |
| updatestream | PUT, DELETE |
| videvent | PUT |
| vidhello | PUT |

## GET status

Returns information describing the xBRC’s status:

<?xml version=”1.0” encoding=”UTF-8” standalone=”no” ?>  
<venue name=”*venue name*” time=”*timestamp*”>  
 <version>*string*</version>  
 <model>*model class*</model>  
 <status>*string*</status>  
 <statusMessage>*string*</statusMessage>  
 <JMSBroker>*server:port*</JMSBroker> <updateStreamUrl>*url*</updateStreamUrl>  
 <readerLocationsCount>*count*</readerLocationsCount>  
 <messageCount>*count*</messageCount>  
 <lastMessageSeq>*number*</lastMessageSeq> <lastMessageToJMS>*number*</lastMessageToJMS>  
 <lastMessageToUpdateStream>*number*</lastMessageToUpdateStream> <perfMetricsPeriod>*number*</perfMetricsPeriod>  
 <startPerfTime>*date*</startPerfTime>  
 <readerTestMode>*true/false*</readerTestMode>  
 <perfMETRIC>  
 <min>*double value*</min>  
 <max>*double value*</max>  
 <mean>*double value*</mean>  
 </perfMETRIC>  
 … more <perfMETRIC> tags …  
</venue>

| Status Element | Purpose |
| --- | --- |
| version | The xBRC software version. In the form: x.x.x.x.tag |
| model | The name of the Java class implementing the xBRC model. This is typically of the form *com.disney.xband.xbrc.MODEL.CEP* where MODEL is “attractionmodel”, “parkentry” or “space”. |
| status | “Red”, “Green” or “Yellow” |
| statusMessage | Short explanation if status is not “Green” |
| JMSBroker | The server and port number being used to communicate over JMS. Blank or a string prefixed with “#” if JMS communication is disabled. |
| updateStreamURL | If the xBRC is transporting messages over HTTP, this tag identifies the URL where it is PUT’ing its messages. |
| readerLocationsCount | The number of reader locations in the current xBRC reader configuration. |
| messageCount | The number of messages being queued by the xBRC that have yet to be sent to JMS or to an HTTP update stream URL. |
| lastMessageSeq | The last message sequence number used in the HTTP update stream. |
| lastMessageToJMS | The last message sequence number sent to the JMS. If this is not the same as *lastMessageSeq* there are messages awaiting to be sent via JMS. |
| lastMessageToUpdateStream | The last message sequence number sent via HTTP. If this is not the same as *lastMessageSeq* there are messages awaiting to be sent via HTTP. |
| perfMetricsPeriod | The sample size (in seconds) over which performance metrics are collected. Metrics are reset to zero at the end of each sample. |
| startPerfTime | The ISO 8601 date/time when the current performance metrics sample was started. |
| readerTestMode | Set to *true* if the readers connected to the xBRC are operating in “test mode”. |
| perf*METRIC* | min, max and mean values for *METRIC*. The currently available *METRIC*s are: *Events, EventAgeMsec, IDMSQueryMsec, EKGWriteMsec, SingulationMsec, PreModelingMsec, ModelingMsec, PostModelingMsec, ExternalMsec, WriteToReaderMsec, SaveGSTMsec, UpstreamMsec, MainLoopUtilization, Model1, Model2* and *Model3.* |

## GET heartbeat

This call that may be used to determine if the xBRC is up and running. This call may be called repeatedly as it incurs very minimal processing in the xBRC.

<heartbeat>  
<lastProcessingDuration>20</lastProcessingDuration>  
<mainThreadLoopCount>314661</mainThreadLoopCount>  
</heartbeat>

The information returned in this call is serves as an indicator that the xBRC is working. The lastProcessingDuration is the number of milliseconds that it took to process all events during the last processing cycle. The mainThreadLoopCount shows how many processing cycles xBRC performed since start. If this value is not increasing the main processing thread may be dead locked or temporarily locked.

## GET messages

**GET messages/guestid/<guestid>**

This operation returns an XML structure.

<?xml version=”1.0” encoding=”UTF-8” standalone=”no” ?>  
<venue name=”*venue name*” time=”*timestamp*” >  
 <message type=”*message type”* time=”*timestamp*”>  
 <sequence>*number*</sequence>  
… additional elements depending on message type …  
 </message>  
… additional messages …  
</venue>

Note the addition of the <sequence> element. This element contains a monotonically increasing number for each xBRC message. Its purpose is for use in the after parameter described below.

In the first form of the path, “GET messages”, the operation returns messages for all guests. In the second form, the operation returns messages only for the identified <guestid>.

Note, too, that a timestamp is included with the venue element as well as with each message element. The first timestamp reflects the time when the batch of events is retrieved while the timestamps in each message identify when the message was originally generated.

The operation will support two optional parameters:

| Parameter | Purpose |
| --- | --- |
| after=*number* | Requests only messages whose sequence numbers are greater than the indicated value. If missing, the xBRC will return all of the messages that it has cached (subject to the *max* parameter). |
| max=# | Requests that a maximum number of messages be included in the reply |

Generally, downstream applications will not use GET messages to retrieve information from an xBRC. Instead, they will read messages from the JMS bus or will use “PUT updatestream” to place the xBRC in “push mode”. When in push mode, the xBRC will periodically send queued messages to a configured URL using HTTP PUT. This request is described in the following section.

## PUT updatestream

This operation requests that the xBRC operate in “push” mode. In this mode, the xBRC will use HTTP PUT to communicate messages with a given URL on a designated, periodic, basis. Push mode can only operate with a single destination. If this message is repeated, the latest updatestream message overrides the settings of any earlier message.

The payload for the operation will can provide values for four elements:

<updatestream>  
 <url>url for update stream</url>  
 <after>number</after>  
 <interval>milliseconds</interval>  
 <max>count</max> <preferredGuestIdType>type</preferredGuestIdType>  
 <messageTypes>message types</messageTypes>  
</updatestream>

|  |  |
| --- | --- |
| Element | Purpose |
| url | Identifies the URL to which the xBRC should post messages |
| after | Requests only messages whose sequence numbers are greater than the indicated value |
| interval | Requests that messages be posted using the designated period (in milliseconds). It is suggested that this value not be smaller than 100. |
| max | Requests that a maximum number of messages be included in each posted |
| preferredGuestIdType | Requests that a particular *type*  of guest id (e.g. “Fidelio”) be provided for guests, if present |
| message types | Space separated list of message types to receive. Use \* to receive all types. Example: “HASENTERED EXITED” |

When messages are posted to the designated URL, the message body is identical to that described in section 7.6.1.

Note that an xBRC that is operating in push mode will still respond to pull requests and will still publish JMS messages if configured to do so.

## DELETE updatestream

Disables “push mode”.

## GET gueststatus

**GET gueststatus/guestid/<guestid>**

This operation returns information about the Guests known to the xBRC. It is primarily intended for use when using the Space model but can also be used with other models.

In the first form, the operation returns information about all of the Guests known to the xBRC. In the second form, the operation returns only information about the specified <guestid>.

The response from this message will be XML data of the form:

<?xml version=”1.0” encoding=”UTF-8” standalone=”no” ?>  
<venue name=”*venue name*” time=”*timestamp*”>  
 <guest>  
 <id>*guestid*</id>  
 <xpass>*true or false*</xpass>  
 <state>*string*</state>  
 <location>  
 <name>*location name*</name>  
 <id>*location id*</id>  
 <arrived>*timestamp*</arrived>  
 <latest>*timestamp*</latest>  
 </location>  
 </guest>  
… additional guest information messages …  
</venue>

For each guest, the xBRC reports his/her current location within the venue. The “arrived” element identifies when the Guest first appeared at the location while the “latest” element identifies the time of the most recent band transmission received at the location. Note that Guests may be considered to be at an old location for a few seconds after their departure (unless they’re picked up at another reader). This latency in the xBRCs operation is to account for occasional band transmissions that might be missed by a long range reader.

Note that GET gueststatus also returns information about the guest (whether he or she is an xPass guest and in what model “state” the guest is). State strings are model dependent, but here are some typical values: HASENTERED, HASMERGED, LOADING, RIDING, EXITED.

## DELETE gueststatus

This clears all guest status from the xBRC. This request can be used to “reset” an xBRC by telling it to clear its internal guest status table.

## GET readerlocationinfo

**GET readerlocationinfo/index/<location index>**

**GET readerlocationinfo/name/<location name>**

This operation returns information about the reader locations known to the xBRC. In the first form, the xBRC returns information about all known locations. In the second form, it returns information about the location designated by <location index>. In the last form, the location is identified by name. The data returned looks like:

<?xml version=”1.0” encoding=”UTF-8” standalone=”no” ?>  
<venue name=”*venue name*” time=”*timestamp*”>  
<readerlocationinfo>  
 <readerlocation>  
 <locationid>*editable location identifier*</locationid>  
 <name>*location name*</name>

<id>*auto generated location id*</id>  
 <type>*type code*</type>  
 <typename>*type name*</typename>

<x>*value*</x>  
 <y>*value*</y>

<useSecureId>true/false</useSecureId>

<successSequence>sequence name</successSequence>

<successTimeout>milliseconds value</successTimeout>

<failureSequence>sequence name</failureSequence>

<failureTimeout>milliseconds value</failureTimeout>

<errorSequence>sequence name</errorSequence>

<errorTimeout>milliseconds value</errorTimeout>

<idleSequence>sequence name</idleSequence>  
<readers>

<reader>

<name>*reader name*</name>

<id>*reader id*</id>

<deviceid>*number*</deviceid>

<type>*reader type*</type>

<macaddress>*address*</macaddress>

<ipaddress>*address*</ipaddress>

<port>*port number*</port>

<gain>*double from -25.0-25.0*</gain>

<lane>*number*</lane>

<threshold>*integer from 0-63*</threshold>

<timelasthello>*date*</timelasthello>

<status>*Red/Yellow/Green*</status>

<statusMessage>*string*</statusMessage>

<version>*string*</version>

<minXbrcVersion>*string*</minXbrcVersion>

<xbioSerialNumber>*number*</xbioSerialNumber>

<istransmitter>*true/false*</istransmitter>

<battery>

<transmitPayload>*use only by simulators*</transmitPayload>

<useSecureId>*true/false*</useSecureId>

<transmitCommands>

<transmitCommand>

<command>*FAST\_RX\_ONLY/SLOW\_PING/FAST\_PING*

</command>

<interval>*integer*</interval>

<mode>*REPLY/BROADCAST*</mode>

<recipients>

<recipientLocationId>*integer*

</recipientLocation>

… -- other recipient location IDs

</recipients>

<timeout>*integer*</timeout>

</transmitCommand>

… -- other transmit commands

</transmitCommands>

<hardwareType>*xTP1/xTP2/…*</hardwareType>

<signalStrengthTransitThreshold>*integer from -127 to -40</*signalStrengthTransitThreshold>

*<batteryLevel>battery charge percent – only present if a battery is present </batteryLevel>*

*<batteryTime>estimated battery life left in minutes – only present if an estimate can be generated</batteryTime>*

*<temperature>temperature in Celsius</temperature>*

*<antennas>true/false – should xBRv4 antenna be powered and will have as many as there are antennas</antennas>*

</reader>

</readers>

</readerlocation> … other reader locations …

</readerlocationinfo>

</venue>

|  |  |  |
| --- | --- | --- |
| Element | Purpose | |
| Name | Name of the reader location which appears in the <readerlocation> elements in other get requests and messages. | |
| Id | Numeric id of the reader location which appears in the <readerlocation> elements in other get requests and messages. | |
| type | Integer identifying the operational type of the location. These values are defined as: | |
|  | 1 (ENTRY) | A reader location used to indicate entry to a venue (in the Attraction and Controller Space model). |
|  | 2 (WAYPOINT) | An intermediate reader location used to trigger an external action or simply to keep track of a guest’s location. |
|  | 3 (EXIT) | A reader location used to detect when a Guest has exited an Attraction or controlled Space. |
|  | 4 (LOAD) | A reader location placed immediately before a Guest boards an attraction “car”. The readers at the location will detect bands attached to attraction cars in addition to those worn by Guests. |
|  | 5 (INCAR) | A reader location placed so that it detects only a particular attraction car and the guests within it. |
|  | 6 (MERGE) | A reader location placed where the xPass and standby queues merge. |
|  | 7 (XPASSENTRY) | A reader location placed at the beginning of the xPass queue. Readers at this location are involved with xPass entitlement checking. |
|  | 8 (COMBO) | A reader location that serves as both XPASSENTRY and MERGE. |
| x | Used by simulators only. Please disregard. | |
| Y | Used by simulators only. Please disregard. | |
| useSecureId | If not provided a global configuration setting is used. If *true*, xBRC instructs xTPs to read Secure ID off of the rfid media used to touch and uses the Secure ID read to verify that media against the IDMS database. Otherwise, Public ID is used. | |
| successSequence | Name of a file containing a combination of lights and sounds sequences to be played by a xTP reader in the event of a successful touch. Definition of a successful touch is model and location dependent. When not explicitly provided on a location a global configuration setting is used. | |
| successTimeout | Stop playing the success sequence after this many milliseconds. When not explicitly provided on a location a global configuration setting is used. | |
| failureSequence | Name of a file containing a combination of lights and sounds sequences to be played by a xTP reader in the event of a rejected touch. Definition of a rejected touch is model and location dependent. When not explicitly provided on a location a global configuration setting is used. | |
| failureTimeout | Stop playing the failure sequence after this many milliseconds. When not explicitly provided on a location a global configuration setting is used. | |
| errorSequence | Name of a file containing a combination of lights and sounds sequences to be played by a xTP reader in the event of an error on touch. Definition of an error touch is model and location dependent. When not explicitly provided on a location a global configuration setting is used. | |
| errorTimeout | Stop playing the error sequence after this many milliseconds. When not explicitly provided on a location a global configuration setting is used. | |
| idleSequence | Name of a file containing a combination of lights and sounds sequences to be played by a xTP reader continuously. When not explicitly provided on a location a global configuration setting is used. | |

The <readers> element enumerates all of the readers (xBR or xTP) present at the reader location. Long range xBR readers are typically deployed in “gangs” of 4 while touch readers are typically deployed 2-8 per location (two in the Attraction model, eight at park entry). For each reader, the following information is provided:

| Reader Element | Purpose |
| --- | --- |
| name | The name of the reader. Note that reader names are generally chosen to reflect the location. For example, the Entry location may consist of four xBRs named “entry-1”, “entry-2”, “entry-3” and “entry-4”. |
| id | A numeric id associated with the reader. This id remains constant even if the reader is renamed. |
| deviceid | Similar to *id*, but assigned by the OneSource system. |
| lane | In the Attraction model, this is typically 0 or 1 for the left and right-side reader at a location. In Park Entry, however, this is typically 0-7. The value is N/A for long-range readers. |
| type | “lrr” (an xBR), “xTP” (touch) or “xTP+xbio” (touch with biometrics) |
| macaddress | The network MAC address of the reader. |
| ipaddress | The ip address of the reader. This is expected to be an address on a physical or virtual private network of the form 192.168.0.x. Note that the xBRC operates as a DHCP server for devices in the private network. |
| port | The TCP port on which the reader listens for configuration and monitoring requests (typically, 8080). |
| gain | A floating point (double) value from 0-63.0 that is used to attenuate signal strength data received by the reader. An incoming signal strength value from the reader will be multiplied by its gain value. |
| threshold | An integer value from -90 to -40 used to filter out low signal strengths. A band message whose signal strength (after being multiplied by the reader’s *gain*) is less than the threshold value will be ignored. |
| timelasthello | Long integer - milliseconds since the start of the epoch. Identifies when the reader has sent a PUT hello to the xBRC. |
| status | “Red”, “Yellow” or “Green” |
| statusMessage | Explanatory text if status is not “Green” |
| version | The installed version of the reader software |
| minXbrcVersion | The minimal version of the xBRC software with which the reader is designed to work. |
| xbioSerialNumber | The serial number of the biometric hardware, if *type* is *xfp+xbio* |
| x | The x coordinate of the reader. The reader may be assigned an (x,y) location for user interface visualization purposes. This value serves no other functional role in the xBRC. |
| y | They coordinate of the reader. The reader may be assigned an (x,y) location for user interface visualization purposes. This value serves no other functional role in the xBRC. |
| istransmitter | Is this reader capable and configured to transmit commands to xbands. Applies only to xBR readers. |
| transmitPayload | Used by simulators only. Please disregard. |
| useSecureId | Inherited from the location. |
| hardwareType | *xTP1* represents the first version of xTP without the RGB capabilities. *xTP2* represents xTPs with RGB LED lights. |
| signalStrengthTransitThreshold | Transmitting xBR transmits applicable commands in response to a ping from bands transmitting with a signal strength specified in this element. Valid range is -127 to -40. |

The <transmitCommands> element enumerates band commands that a transmitting long range reader is configured to either broadcast or send on reply. For transmit command, the following information is provided:

|  |  |
| --- | --- |
| Element | Purpose |
| command | Command to transmit. Possible choices are *FAST\_PING*, *SLOW\_PING*, and *FAST\_RX\_ONLY*. |
| interval | Band commands will be transmitted as often as specified by the interval. Specified in milliseconds. |
| mode | Either *REPLY* of *BROADCAST*.  *REPLY* mode means that a transmitting xBR will reply to a ping from a band by sending it a command. This means that a command will be sent on a second ping, since the first ping is used for detection and the second ping is used to respond to.  *BROADCAST* modes translates to a transmitting xBR sending a band command continuously to be picked up by a band as soon as its first ping is detected.  A single transmitting xBR can be configured to send either a single *BROADCAST* command or a list of *REPLY* commands, but not a combination of the two. |
| recipients | One too many location IDs. The list of bands that are to receive this command is comprised of bands seen at these locations since the last time the GST table has been cleared. Applicable to *REPLY* mode commands only. |
| timeout | Number of milliseconds after which a band will revert to the next most energy efficient mode. FAST\_PING reverts to SLOW\_PING, which reverts to FAST\_RX\_ONLY, which in turn reverts to SLOW\_RX\_ONLY. A band may not be set to the SLOW\_RX\_ONLY mode, it can only revert to it. |

Refer to xBand\_RF\_ICD.doc for details on communication protocol between the xBand and the xBR.

## GET readerstats/reader/<reader name>

Returns statistical information regarding the performance of a particular reader (identified by <reader name>:

<?xml version=”1.0” encoding=”UTF-8” standalone=”no” ?>  
<venue name=”*venue name*” time=”*timestamp*”>  
 <readerstats>  
 <reader>  
 <name>*name*</name>  
 <stats>  
 <stat>  
 <channel>*number*</channel>  
 <frequency>*number*</frequency>  
 <ss>*number*</ss>  
 <count>*number*</count>  
 </stat>  
 … more stat elements …  
 </stats>  
 </reader>  
 … more reader elements …>  
 </readerstats>  
</venue>

The name element value should match the name in the URL. The stats elements provide information about the 8 radios present in the reader. These radios are organized into two different channels (0 or 1). Each radio is programmed to one of four frequencies. This request returns a signal strength histogram for the data received by each of the 8 radios. Each stat element identifies a radio (channel, frequency) and identifies how many events (count) were received with a particular signal strength (ss). The statistical data for the reader is cleared after the response is generated.

The elements in *stat* are defined as per the following table.

|  |  |
| --- | --- |
| Stat Element | Purpose |
| channel | The channel number (0 or 1) of the radio |
| frequency | The frequency of the radio (2401, 2424, 2450 or 2476) |
| ss | A particular signal strength (-90 to -40). |
| count | The number of events received with the indicated signal strength. Note that if *count* is zero, the entire *stat* element is omitted. |

## DELETE readerstats

**DELETE readerstats/reader/<reader name>**

This clears reader performance statistics for a particular reader (second form) or for all readers.

## GET bandevents/band/<long-range bandid>

This URL is used to retrieve data to support the “power level” UI functionality in the xBRC. The operation retrieves the latest (most recent packet number) low-level xBR (long-range) events for the band with the indicated long-range id:

<?xml version=”1.0” encoding=”UTF-8” standalone=”no” ?>  
<venue name=”*venue name*” time=”*timestamp*”>  
 <events>  
 <event>  
 <type>*LRR*</type>  
 <xlrid>*id*</xlrid>  
 <time>*timestamp*</time>  
 <readerName>*name*</readerName>  
 <eno>*number*</eno>  
 <pno>*number*</pno>  
 <freq>*number*</freq>  
 <chan>*number*</chan>  
 <ss>*number*</ss>  
 </event>  
 … more event elements …  
 </events>  
</venue>

|  |  |
| --- | --- |
| Event Element | Purpose |
| type | Identifies the type of event. Always “LRR”. |
| xlrid | The long-range id that originated this message. Should match the id specified in the URL. |
| time | The timestamp associated with the event (in ISO 8601 form). |
| readerName | The name of xBR reader reporting the event. |
| eno | The event number of the event. |
| pno | The packet number (0-255) of the event. All the events in the xml message will have the same packet number value. |
| freq | The frequency (2401, 2424, 2450 or 2476) of the radio that received the event. |
| chan | The channel (0 or 1) of the radio bank that received the event. |
| ss | The signal strength (0-64) associated with the event. |

## PUT storeconfiguration

This request stores current configuration information in the “stored configurations” table of the xBRC. An xBRC can store multiple configurations and can switch between them on request. A configuration consists of:

This request stores current configuration information in the “stored configurations” table of the xBRC. An xBRC can store multiple configurations and can switch between them on request. A configuration consists of:

- All the configuration parameter values

- The reader location and reader tables

- The “subway diagram” for the configuration

- Additional, model-specific, information

PUT storeconfiguration supports two parameters, name and description. As parameters are specified as part of the URL, remember to escape these as necessary to conform to URL encoding rules.

|  |  |
| --- | --- |
| Parameter | Purpose |
| name=*string* | Specifies the name to be given to the stored configuration. Note that names must be unique. If a configuration already exists with the given name, a 500 error will be returned. |
| description=*string* | A description of the stored configuration. Note that if this parameter is omitted, the xBRC will assign a description based on the time when the configuration was stored. |

PUT storeconfiguration returns a text-encoded number (mime type text/plain) that is numeric id assigned to the newly stored configuration. This number may be used later in the PUT selectconfiguration request.

## GET configurations

**Note: returns JSON!**

This request returns information about the configurations stored in an xBRC. The result is a JSON encoded object of the form:

{  
 “configurations”:  
 [  
 {  
 “configurationId” : *number*,  
 “name” : *string*,  
 “description” : *string*  
 },  
 … other configurations …  
 ]  
}

“configurationId” identifies the number assigned to the stored configuration. “name” identifies the name provided when the configuration was stored. “description” is either the description provided when the configuration was stored or an automatically generated description in none was provided.

This request returns JSON as the primary customer for this request is a GXP application that can more easily parse JSON than XML

## GET currentconfiguration

**GET configuration/name/<name>**

**GET configuration/id/<id>**

This request returns an XML document that describes a configuration. The configuration can be requested by name (including “current” for the current configuration) or by id. The “currentconfiguration” path yields the same results as “configuration/name/current”. The general format of the returned document (of mime type application/xml) is:

<?xml version=”1.0” encoding=”UTF-8” standalone=”no” ?>  
<venue name=”*venue name*” time=”*timestamp*”>  
 <configuration name=”current” type=”full”>  
 <description/>   
 <properties>  
 <property class=”*class”* name=”*name”*>*value*</property>  
 … more properties …  
 </properties>  
 <readerlocationinfo>  
 ……  
 </readerlocationinfo>  
 <griditems>  
 <griditem id=”*number”* type=”*type”* xgrid=”*number”* ygrid=”*number”*>  
 <state>*string*</state> <label>*string*</label><description>*string*</description>  
 <image>*string*</image>  
 <sequence>*number*</sequence>  
 <gueststoshow>*0,1 or 2*</gueststoshow>  
 <locationid>*number*</locationid>  
 </griditem>  
 … more grid item entries…  
 </griditems>  
 <images>  
 … image information …  
 </images>  
 <model>*model specific tags*</model>  
 </configuration>  
</venue>

The name attribute in the configuration tag will always be “current” and the type attribute will always be “full”. The description tag will always be empty. If the retrieved XML is used as a template for uploaded configurations (see section 4.13), these values should be changed as necessary. Note that a “full” configuration completely replaces an existing one (when selected) whereas a “partial” configuration is “additive” (it only replaces existing data with data provided in the new configuration).

The properties section contains all of the name-value pairs that are used to control and fine-tune the operation of the xBRC. xBRCs use dozens of properties to identify adjunct systems (for example, to locate a GXP or Omni system), to provide credentials and to tune various operations (for example, abandonment timeouts or xTP light durations.

The readerlocationinfo section is exactly as described earlier in section 4.6 - it describes all of the reader locations configured in the xBRC and the readers assigned to those locations.

The griditems section describes the “subway diagram” for the xBRC. It consists of a series of griditem tags each of which defines an entry in a subway map square. Each grid item is identified by an id number and is of a provided type and occupies the x, y location in the grid identified by xgrid and ygrid. These and other tags/attributes are described in the following table:

|  |  |
| --- | --- |
| <griditem> tag or attribute | Description |
| id | A numeric id assigned to the item. |
| type | A string identifying the type of grid item: *Gate, HPath, VPath, TNorth, TSouth, TWest, TEast, Cross, ESTurn, WSTurn, ENTurn, WNTurn.* |
| xgrid, ygrid | The x and y coordinate of the item. |
| label | A string to display in a *Gate* type item. |
| image | A URL (local to the xBRC) identifying the bitmap to display for a *Gate.* |
| description | A description (used for popup text) to associate with the item. |
| state | The model state for which the grid item should display guest counts. This tag is mutually exclusive with <locationid>. |
| locationid | The location id for which the grid item should display guest counts. This tag is mutually exclusive with <state>. |
| sequence | A number assigning the priority of this item relative to other items displaying similar guest counts |
| gueststoshow | 0 if all guests, 1 if xPass only, 2 if standby only |

The images section describes any images stored in the xBRC (typically, for use by the subway diagram) while the model section can contain additional, model-specific, information.

## POST configuration?name=*name*&description=*description*

**POST newconfiguration?name=*name*&description=*description***

**PUT configuration?name=*name*&description=*description***

This request allows a configuration to be uploaded. If the POST verb is used, a new configuration is created. If the PUT verb is used, the configuration with the given name is replaced by the uploaded version. The information for the uploaded configuration is provided as they payload of the request. The form of this payload is identical to the form described in GET configuration (section 4.13).

The name and description for the configuration can be provided by URL parameters or in the supplied XML.

Either “full” or “partial” type configurations may be uploaded. Partial configurations need not have all sections of the configuration specified - only the sections where they specify changes.

The newconfiguration endpoint is identical to POST configuration behavior.

## DELETE configuration/name/<name>

## DELETE configuration/id/<id>

These requests can be used to delete stored configurations by specifying their name or id.

## PUT selectconfiguration/name/<name>

**PUT selectconfiguration/id/<id>**

These requests replace the current configuration with a stored configuration identified by its name or by its assigned id. If the stored configuration is a full configuration, the operation begins by removing all current properties, reader locations, readers and grid items. The configuration must fully specify all the information needed to repopulate these items. If the stored configuration is a partial configuration, however, it need only contains the items that it wants to change. In this case, PUT selectconfiguration does not delete items en masse - it only deletes an item when it detects that a value is being provided in the selected configuration. As a result of this algorithm, note that partial configurations are always “additive”. A partial configuration can never “delete” a current setting; it can only overwrite it with a new value.

## GET ekgposition

This request returns the current “cursor” position of the low-level “ekg” event log file. The purpose of this call is to help provide a value for the position parameter in the GET ekg request (section 4.19).

The ekg file is used, mostly, for diagnosing the low-level operation of the xBRC; it is of little interest once an xBRC has been deployed and configured for a particular facility.

GET ekgposition returns a “long” text encoded number (mime type text/plain).

## GET ekg

This request returns the contents of the low-level ekg event log file. The request supports two parameters:

|  |  |
| --- | --- |
| Parameter | Purpose |
| position=*long int* | Specifies the initial position within the ekg file from which the xBRC should read. |
| max=*int* | Specifies the maximum number of lines that the request should return. |

The response to the request is of mime type text/plain. It consists of the requested contents of the ekg file appended with a text encoded number that identifies the next value that can be used as the position value if another chunk of data is to be read.In general, the algorithm for reading the ekg file through the HTTP interface is:

1. Use GET ekgposition to determine the current end of file

2. Loop

a. Call GET EKG?position=value where value is the number returned in step 1.

b. Read the response but treat the last line as a number providing a new value for position.

## PUT logcomment?text=<string>

Inserts a string into the ekg file. This call can be used by diagnostic tools to insert a recognizable marker into the low -level event file.

## PUT updateconfig

This request is used by external programs (for example, the xBRC UI or xbrcconfig) that modify the xBRCs database directly then need to inform the xBRC that it should re-read its configuration from the database.

## DELETE cache

Removes cached guest and band information from the xBRC. This call should be used if changes have been made to the xBMS or other band service and there is the possibility that the xBRC is holding out-of-date information. Various forms of this path are available as documented below.

### DELETE cache

This form removes all cached information from the xBRC.

### DELETE cache/guest/id/<guest-id>

Removes cached guest information for the guest identified by the given guest id.

## DELETE cache/band/secureid/<secure-id>

### DELETE cache/band/lrid/<long-range-id>

### DELETE cache/band/rfid/<rf-id>

Removes cached band information for the band identified by the given secure ID, long range ID or RFID (tap id).

## GET refreshpackages

Request the xBRC to refresh the reader packages from the disk

## PUT videvent

Notify the xBRC that a vehicle just passed a detection point. The payload of the request is shown below.

NOTE: PUT avmsevent is deprecated in xBRC version 1.6.

<message type="VEHICLE" time="2012-02-02T22:07:52.5000000-05:00">

<vehicleid>123456</vehicleid>

<attractionid>WMKHAMA</attractionid>

<sceneid>13B</sceneid>

<locationid>01</locationid>

<confidence>99</confidence>

</message>

## PUT vidhello

The VID reader needs to call this endpoint once per minute.

NOTE: PUT avmshello is deprecated in xBRC version 1.6.

<message type="VEHICLEHELLO" time="2012-02-02T22:07:52.5000000-05:00">

<attractionid>WMKHAMA</attractionid>

<sceneid>13B</sceneid>

<locationid>01</locationid>

</message>

All the fields in the message are echoed back in the INVEHICLE event published when a guest is associated to a vehicle.

Version 1.7.2 of the xBRC introduced support for the VID *software system* hello message. This message is also sent to the xBRC using the /vidhello endpoint. The VID software system may send this message once per minute. Once the first message is received by the xBRC, the xBRC will start monitoring for subsequent messages. If no messages are received within the *readerhellotimeout* period (an xBRC configuration setting) then the xBRC will report the VID system as not responding.

Here is an example of a system hello message:

<message type="SYSTEMHELLO" time="2012-12-21T10:20:43.9697632-05:00">

<vehicleid>HEARTBEAT</vehicleid>

<attractionid>26068</attractionid>

<sceneid>SOFTWARE</sceneid>

<locationid>SOFTWARE</locationid>

<confidence>99</confidence>

<status>Yellow</status>

<statusmessage>System low on memory</statusmessage>

</message>

|  |  |
| --- | --- |
| Parameter | Purpose |
| type | Must be “HEARTBEAT” (case sensitive) |
| time | The time when the message was sent by the VID software system. |
| vehicleid | This field is left for compatibility with the videvent message. It is ignored. |
| attractionid | This field is left for compatibility with the videvent message. It is ignored. |
| sceneid | This filed is left for compatibility with the videvent message. It is ignored. |
| locationid | This field is left for compatibility with the videvent message. It is ignored. |
| confidence | This field is left for compatibility with the videvent message. It is ignored. |
| status | Optional field. It is either “Green”, “Yellow”, or “Red” (case sensitive) indicating the severity of the statusmessage field. |
| statusmessage | The VID system may notify the xBRC of any severe problems that impact system functionality. This message is the description of the problem. If the status is Green, then this field is ignored. |

## GET perfmetricsmetadata

This request returns the metadata describing the performance metrics that xBRC collects. The following information about each metric is returned:

|  |  |
| --- | --- |
| Parameter | Purpose |
| name | Name under which the xBRC stores this metric’s data structure on its status object. |
| displayName | Human friendly name to use as a user facing title for this metric. |
| description | Description of performance data collected by this metric. |
| version | Version of this meta data. This is used to preserve historical meaning of this metric. |
| type | Units of performance data collected by this metric. |

## PUT mediapackage

This request allows a file containing media content (color sequences, sound sequences, WAV files) to be uploaded to the xBRC. In turn, the media package is pushed to all of the readers that the xBRC is associated with. The xBRC can be configured through the xBRC user interface to play individual media files or media sequences in response to various xBRC events. In addition, media files and sequences can be queued to a reader using the playsequence API.

A call to mediapackage should include a binary encoded file as the message body of an HTTP POST request. This media package file should consist of a collection of directories and files that are compressed into a single file using TAR and GZIP. The naming convention of the resulting file is not important (an example might be “media.tar.gz”).

The media package folder and file structure is flexible but does require two directories for successful operation. The first is LEDSCRIPTS, which contain scripts to drive a combination of light and sound effects. These scripts can refer to any other file within any arbitrary directory structure within the file. The final required directory is SOUNDS, which contains any number of WAV files that could be referenced by LED scripts or played individually.

## DELETE mediapackage

This allows any existing media package to be deleted from the xBRC. In turn, any associated readers will have their media packages deleted. Given that an xBRC can have one and only one media package installed at any given time, a package name does not need to be specified and any existing package will be deleted in response to the request.

## PUT playsequence/<readername>?sequencename=<sequencename>&timeout=<timeoutlength>

This request allows a media sequence (or media resource) to be queued through a specific reader. The specific reader to receive the request must be associated with the xBRC and is identified by the reader’s name through <readername>. The sequence to queue, as identified by <sequencename>, can be a script name, color sequence, or sound file.

The timeout is an optional parameter, as specified by <timeoutlength> in milliseconds, that allows the request to override a sequence’s default sequence length. If no timeout is specified, a sequence script will run according to its own default values and LED sequences and WAV files will play in its entirety. If 0 is specified as the timeout, the specified media sequence will loop indefinitely. To terminate a looping sequence, a playsequence request can be made to the same reader with a reserved sequence name of “off”.

## GET sequences

This request allows the retrieval of all supported sequences within the xBRC’s current media package. The retrieved names can be used as the <sequencename> parameter to the playsequence API.

The request returns the following example payload (and includes supported default sequence names):

<sequences time="2012-11-29T22:47:58.597">

<sequence name="blue" readerdefault="true"/>

<sequence name="exception" readerdefault="true"/>

<sequence name="green" readerdefault="true"/>

<sequence name="inner\_blue" readerdefault="true"/>

<sequence name="inner\_green" readerdefault="true"/>

<sequence name="inner\_red" readerdefault="true"/>

<sequence name="inner\_white" readerdefault="true"/>

<sequence name="inner\_yellow" readerdefault="true"/>

<sequence name="off" readerdefault="true"/>

<sequence name="outer\_blue" readerdefault="true"/>

<sequence name="outer\_green" readerdefault="true"/>

<sequence name="outer\_red" readerdefault="true"/>

<sequence name="outer\_white" readerdefault="true"/>

<sequence name="outer\_yellow" readerdefault="true"/>

<sequence name="red" readerdefault="true"/>

<sequence name="success" readerdefault="true"/>

<sequence name="thinking" readerdefault="true"/>

<sequence name="white" readerdefault="true"/>

<sequence name="yellow" readerdefault="true"/>

</sequences>

## PUT scheduleoverrideon?minutes=<minutes>

Inform battery powered devices (like an xBRv4) that they should remain ‘on’ for some duration given by the ‘minutes’ parameter.

This interface is only of use if a schedule is set (see properties) and battery powered devices are in use. Using this interface under other circumstances will not cause problems.

If the interface is used during a scheduled ‘on’ period, the ‘on’ period will be extended by the ‘minutes’ parameter.

If the interface is used during a schedule ‘off’ period, the devices will be turned on for at least the time specified by the ‘minutes’ parameter.

## GET scheduleoverrideclear

Clear any existing schedule override set with the scheduleoverrideon interface. This is useful if the override was scheduled in error or the requested length of time was not needed.

## GET transmitreaderstatus

The xBRC allows for configuring a prioritized list of long range readers capable of transmitting band commands at a location. Only one reader is told to transmit at a time. When the current transmitter stops saying Hello to the xBRC, the xBRC picks the next priority reader as the transmitter.

The transtmitreaderstatus can be used to find out which of the long range transmit readers are currently designated to transmit band commands for each location configured with transmit commands.

Below is an example output of the transmitreaderstatus REST call.

<transmitReaderStatus facilityId="26068" name="Expedition Everest" time="2013-05-15T17:31:49.178">

<locations>

<location>

<currentTransmitter>lrr-2</currentTransmitter>

<name>Entry</name>

<readers>

<reader>

<alive>true</alive>

<currentTransmitter>true</currentTransmitter>

<ipAddress>127.0.0.1</ipAddress>

<name>lrr-2</name>

<transmitterHaPriority>1</transmitterHaPriority>

</reader>

<reader>

<alive>true</alive>

<currentTransmitter>false</currentTransmitter>

<ipAddress>127.0.0.1</ipAddress>

<name>lrr-1</name>

<transmitterHaPriority>2</transmitterHaPriority>

</reader>

</readers>

<statusMessage>There are currently no transmit readers responding at this location.</statusMessage>

</location>

</locations>

</transmitReaderStatus>

# Messaging Interface

This section describes the format of the XML messages sent via JMS or HTTP to downstream applications.

## xBRC Models

Not all of the messages described in this document are generated in all of the xBRC operating models. The following table identifies which messages are used by each model.

|  |  |
| --- | --- |
| Model | Message Types Used |
| Attraction | ENTRY, MERGE, LOAD, INVEHICLE, EXIT, ABANDON, METRICS, BANDSTATUS, LOCATIONEVENT, LOCATIONABANDON, DISCOVERY |
| Space | METRICS, LOCATIONEVENT, LOCATIONABANDON, DISCOVERY |
| Park Entry | METRICS, TAPPED, ABANDONED, BLUELANE, HASENTERED, PARKENTRYEVENT, PARKENTRYABANDON, BANDSTATUS, DISCOVERY |

## High Level Description of xBRC Messages

Before describing how xBRC messages may be read, let us consider what data xBRC messages might contain. In general, an xBRC datum reflects some “event” that has occurred due to an xBand transmitter communicating with a reader (touch or long-range). Many xBRC messages correspond one-to-one with xBR events as documented in the Reader Interface Control Document. The following table describes the different types of messages/events that an xBRC can report. The table identifies a message type with a string. These same strings appear later in message payloads identifying the nature of the message.

|  |  |
| --- | --- |
| Message Type | Purpose |
| ENTRY | A Guest has entered an “venue” |
| MERGE | A Guest has touched a merge-point reader (in a FP+ queue) |
| LOAD | A Guest has been “loaded” onto a venue vehicle (typically, on some type of “car”) |
| INVEHICLE | Similar to LOAD event, but generated in response to the VEHICLE event received from the AGC system. |
| EXIT | A Guest has exited a venue |
| ABANDON | A Guest has abandoned a venue |
| METRICS | A venue is reporting aggregate metrics |
| TAPPED | A Guest has touched a Park Entry DAP reader |
| ABANDONED | A Guest that tapped on a Park Entry DAP reader abandoned the entrance. Reason is supplied in message (timed out, tapped elsewhere, incomplete bio-scan) |
| BLUELANED | A Guest that tapped on a Park Entry DAP reader was denied entry due to a ticketing problem. |
| HASENTERED | A Guest that tapped on a Park Entry DAP reader was successfully verified against ticketing and was granted access. |
| PARKENTRYEVENT | A Guest was seen near the Park Entry long range reader location(s) of a given xBRC. |
| PARKENTRYABANDON | A Guest that was seen near the Park Entry long range reader location(s) of a given xBRC appears to have left and is no longer detectable. |
| BANDSTATUS | A Guest tapped on a reader at a location that has a known long range reader associated to it, the guest however was not detected by the long range reader which suggests that the band may be having problems (waking or out of battery power). |
| LOCATIONEVENT | A reader has sensed an xBand (by a tap or long range read) |
| LOCATIONABANDON | A Guest has left a specific venue location (no longer detected by a long range reader) |
| DISCOVERY | Message used to describe an available service to the xBRMS xConnect Management System. Identifies information such as name, address, port, type, role, and current status of service. |

Whether these messages are communicated using JMS or RESTful HTTP, data payloads describing these messages will be formatted in XML.

## General XML Payload Format

The general format for an xBRC message payload will be:

<venue name=”*venue name*” time=”*timestamp*”>  
 <message type=”*message type”* time=”*timestamp*”>  
… additional elements depending on message type …

<sequence>*sequence number (See section 4.3)*</sequence>  
</message>  
… additional messages …  
</venue>

The venue name identifies the attraction, dining experience or other application in which the xBRC is installed. Message type is a string taken from Table 1 while time is an ISO 8601 formatted time stamp, for example: “2011-06-20T13:41:00:891” (note the inclusion of milliseconds). Times are always stored in UTC form although they may be displayed as local time for usability purposes. Note that the timestamp in the venue element represents the time when the “batch” of messages are delivered whereas the timestamp in the message element represents the time when the message was generated. Each message includes an identifying sequence number which is explained in chapter 4.3 of this document.

## Specific Message Payloads

### ENTRY Message

<message type=”ENTRY*”* time=”*timestamp*”>

<guestid>*guest id*</guestid>

<publicid>*public id of card or band*</publicid>

<linkid>*link id value*</linkid>

<linkidtype>[*xbms-link-id | gxp-link-id | fidelio-link-id | other*]</linkidtype>

<bandtype>[*Guest | Attraction Card | Cast Member*]</bandtype>

<xpass>[*true | false*]</xpass>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*unique reader id*</readerdeviceid>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| guestid | xConnect IDMS system identifier of the guest that triggered the entry message. |
| publicid | Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| linkid | Global identifier the guest. Typically the xBMS link identifier GUID of a guest. If the link identifier type order setting of the xBRC specifies an alternate preference, then the first type found and matched will be returned |
| linkidtype | Indicates the type of link identifier used in the linkid. Typically “xbms-link-id”, or “fidelio-link-id”, or “gxp-link-id” depending on link identifier type preference order match. |
| bandtype | Value used to indicate the role of the card or band which was used to identify the owner. Typically “Guest”, other forms may include “Attraction Card”, “TEST”, “Cast Member” |
| xpass\* | “true” if guest entered the venue through an xPass queue. “false” if through standby queue |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “entry”, “load”, etc.). The names are unique per defined location. |
| readername | Identifies the individual reader (by name) originating the event. |
| readerdeviceid | unique reader device id |

\*Note that if an xBRC is operating with the Space model (as opposed to the Attraction model) that the *xpass* value will always be set to *true* indicating that the guest entered the controlled space through an xTP touch reader.

### MERGE Message

<message type=”MERGE*”* time=”*timestamp*”>

<guestid>*guest id*</guestid>

<publicid>*public id of card or band*</publicid>

<linkid>*link id value*</linkid>

<linkidtype>[*xbms-link-id | gxp-link-id | fidelio-link-id | other*]</linkidtype>

<bandtype>[*Guest | Attraction Card | Cast Member*]</bandtype>

<xpass>*true*</xpass>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*unique reader id*</readerdeviceid>

<sequence>*sequence number*</sequence>

</message>

| Element | | Purpose |
| --- | --- | --- |
| guestid | | xConnect IDMS system identifier of the guest that triggered the merge message. |
| publicid | | Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| linkid | | Global identifier the guest. Typically the xBMS link identifier GUID of a guest. If the link identifier type order setting of the xBRC specifies an alternate preference, then the first type found and matched will be returned |
| linkidtype | | Indicates the type of link identifier used in the linkid. Typically “xbms-link-id”, or “fidelio-link-id”, or “gxp-link-id” depending on link identifier type preference order match. |
| bandtype | | Value used to indicate the role of the card or band which was used to identify the owner. Typically “Guest”, other forms may include “Attraction Card”, “TEST”, “Cast Member” |
| xpass | | Currently, only xPass guests generate MERGE messages so this value is always *true* |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “entry”, “load”, etc.). The names are unique per defined location. |
| readername | | Identifies the individual reader (by name) originating the event. |
| readerdeviceid | | unique reader device id |

### LOAD Message

<message type=”LOAD*”* time=”*timestamp*”>

<guestid>*guest id*</guestid>

<publicid>*public id of card or band*</publicid>

<linkid>*link id value*</linkid>

<linkidtype>[*xbms-link-id | gxp-link-id | fidelio-link-id | other*]</linkidtype>

<bandtype>[*Guest | Attraction Card | Cast Member*]</bandtype>

<xpass>[*true | false*]</xpass>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*unique reader id*</readerdeviceid>

<carid>*car id*</carid>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| guestid | xConnect IDMS system identifier of the guest that triggered the load message. |
| publicid | Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| linkid | Global identifier the guest. Typically the xBMS link identifier GUID of a guest. If the link identifier type order setting of the xBRC specifies an alternate preference, then the first type found and matched will be returned |
| linkidtype | Indicates the type of link identifier used in the linkid. Typically “xbms-link-id”, or “fidelio-link-id”, or “gxp-link-id” depending on link identifier type preference order match. |
| bandtype | Value used to indicate the role of the card or band which was used to identify the owner. Typically “Guest”, other forms may include “Attraction Card”, “TEST”, “Cast Member” |
| xpass | “true” if guest entered the venue through an xPass queue. “false” if through standby queue |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “entry”, “load”, etc.). The names are unique per defined location. |
| readername | Identifies the individual reader (by name) originating the event. |
| readerdeviceid | unique reader device id |
| carid | Identifies the “car” associated with the guestid at the load location |
| readerdeviceid | unique reader device id |

### INVEHICLE Message

<message type="INVEHICLE" time=”timestamp”>

<guestid>*guest id*</guestid>

<publicid>*public id of card or band*</publicid>

<linkid>*link id value*</linkid>

<linkidtype>[*xbms-link-id | gxp-link-id | fidelio-link-id | other*]</linkidtype>

<bandtype>[*Guest | Attraction Card | Cast Member*]</bandtype>

<xpass>*true/false*</xpass>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*unique reader id*</readerdeviceid>

<vehicle>*vehicle tag id*</vehicle>

<car>*car number of train*</car>

<row>*row number of car or vehicle*</row>

<seat>*seat number or identifier*</seat>

<attractionid>*attraction id*</attractionid>

<sceneid>*scene id*</sceneid>

<vidlocationid>*location id as received from the VID reader*</vidlocationid>

<confidence>*confidence*</confidence>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| guestid | xConnect IDMS system identifier of the guest that triggered the in-vehicle message. |
| publicid | Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| linkid | Global identifier the guest. Typically the xBMS link identifier GUID of a guest. If the link identifier type order setting of the xBRC specifies an alternate preference, then the first type found and matched will be returned |
| linkidtype | Indicates the type of link identifier used in the linkid. Typically “xbms-link-id”, or “fidelio-link-id”, or “gxp-link-id” depending on link identifier type preference order match. |
| bandtype | Value used to indicate the role of the card or band which was used to identify the owner. Typically “Guest”, other forms may include “Attraction Card”, “TEST”, “Cast Member” |
| xpass | “true” if guest entered the venue through an xPass queue. “false” if through standby queue |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “entry”, “load”, etc.). The names are unique per defined location. |
| readername | Identifies the individual reader (by name) originating the event. |
| readerdeviceid | unique reader device id |
| vehicle | Identifies the “vehicle” associated with the guestid. This id is received in the AGC VEHICLE event. Value is a string. |
| car | Identifies the “car” associated with the guestid. This id is based on the count of cars that make up a train for a rollercoaster style attraction. This is a sequence number that counts starting with 1. Value is in decimal. This is optional in the message format. |
| row | Identifies the “row” associated with the guestid. This id is based on the count of rows that make up a car (in train scenarios) or vehicle (boat or single car attraction). This is a sequence number that counts starting with 1. Value is in decimal. This is optional in the message format. |
| seat | Seat number of theater or car/vehicle associated with the guestid. Value is a string. 13A, 2, 4-L, etc. This is optional in the message format. |
| attractionid | Attraction id received in the AGC VEHICLE event. |
| sceneid | Scene id received in the AGC VEHICLE event. |
| locationid | Location Id received in the AGC VEHICLE event. |
| confidence | Confidence received in the AGC VEHICLE event. |
| sequence | Sequence received in the AGC VEHICLE event. |
| readerdeviceid | unique reader device id |

### WITHCHARACTER Message (Draft – planned future feature in 1.8)

This message type would be sent in scenarios where a given long range reader location is enabled to associate designated Character bands with regular Guest bands. The association event would occur when the xBRC is able to read multiple bands for both guests and characters from a common reader location. Character bands are those that are defined in xBMS (bandrole = “Character Band”) resulting in IDMS as bandtype = “Character Band”. If more than on Character Band is present at the same guest location, then the xBRC will send a WITHCHARACTER message per Character.

<message type="WITHCHARACTER" time=”timestamp”>

<guestid>*guest id*</guestid>

<publicid>*guest band public id*</publicid>

<linkid>*link id value*</linkid>

<linkidtype>[*xbms-link-id | gxp-link-id | fidelio-link-id | other*]</linkidtype>

<bandtype>[*Guest | Attraction Card | Cast Member*]</bandtype>

<xpass>*true/false*</xpass>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*unique reader id*</readerdeviceid>

<characterid>*character id*</characterid>

<characterpublicid>*character band* *public id*</characterpublicid>

<characterlinkid>*character link id value*</characterlinkid>

<characterlinkidtype>*character link id type* [*xbms-link-id | gxp-link-id | fidelio-link-id | other*]</characterlinkidtype>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| guestid | xConnect IDMS system identifier of the guest that triggered the with-character message. |
| publicid | Identifies the guest band used at the location of the long range read event for this location. With long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| linkid | Global identifier the guest. Typically the xBMS link identifier GUID of a guest. If the link identifier type order setting of the xBRC specifies an alternate preference, then the first type found and matched will be returned. |
| linkidtype | Indicates the type of link identifier used in the linkid for the guest. Typically “xbms-link-id”, or “fidelio-link-id”, or “gxp-link-id” depending on link identifier type preference order match. |
| bandtype | Value used to indicate the role of the band which was used to identify the owner. Typically “Guest”, other forms may include “Attraction Card”, “TEST”, “Cast Member” |
| xpass | “true” if guest entered the venue through an xPass queue. “false” if through standby queue |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “entry”, “load”, etc.). The names are unique per defined location. |
| readername | Identifies the individual reader (by name) originating the event. |
| readerdeviceid | Unique reader device id |
| characterid | xConnect IDMS system identifier of the character that was associated to the guest in this with-character message. |
| characterpublicid | Identifies the character band used at the location of the long range read event for this location. Wtih long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| characterlinkid | Global identifier the character. Typically the xBMS link identifier GUID of a character. If the link identifier type order setting of the xBRC specifies an alternate preference, then the first type found and matched will be returned. |
| characterlinkidtype | Indicates the type of link identifier used in the linkid for the character. Typically “xbms-link-id”, or “fidelio-link-id”, or “gxp-link-id” depending on link identifier type preference order match. |

### EXIT Message

<message type=”EXIT*”* time=”*timestamp*”>

<guestid>*guest id*</guestid>

<publicid>*public id of card or band*</publicid>

<linkid>*link id value*</linkid>

<linkidtype>[*xbms-link-id | gxp-link-id | fidelio-link-id | other*]</linkidtype>

<bandtype>[*Guest | Attraction Card | Cast Member*]</bandtype>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*unique reader id*</readerdeviceid>

<xpass>*true* | *false*</xpass>

<carid>*car id*</carid>

<statistics>

<waittime>*wait time*</waittime>

<mergetime>*merge time*</mergetime>

<totaltime>*total time*</totaltime>

</statistics>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| guestid | xConnect IDMS system identifier of the guest that triggered the exit message. |
| publicid | I Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| linkid | Global identifier the guest. Typically the xBMS link identifier GUID of a guest. If the link identifier type order setting of the xBRC specifies an alternate preference, then the first type found and matched will be returned |
| linkidtype | Indicates the type of link identifier used in the linkid. Typically “xbms-link-id”, or “fidelio-link-id”, or “gxp-link-id” depending on link identifier type preference order match. |
| bandtype | Value used to indicate the role of the card or band which was used to identify the owner. Typically “Guest”, other forms may include “Attraction Card”, “TEST”, “Cast Member” |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “entry”, “load”, etc.). The names are unique per defined location. |
| readername | Identifies the individual reader (by name) originating the event. |
| readerdeviceid | Unique reader device id |
| xpass\* | “true” if guest entered the venue through an xPass queue. “false” if through standby queue |
| carid\* | Identifies the “car” associated with the guestid at the load location |
| waittime\* | Specifies, in seconds, the time the Guest spent waiting to load the venue’s “ride” (from “entry”) |
| mergetime\* | Specifies, in seconds, the time the Guest spent waiting to reach the merge point. This element is only present if the Guest entered through the xPass queue. |
| totaltime | Specifies, in seconds, the total time the Guest spent in the venue |

\*Note that if an xBRC is operating with the Controlled Space model (as opposed to the Attraction model) that the *xpass* value will always be set to *true* indicating that the guest exited the controlled space through an xTP touch reader. Additionally, *waittime* and *mergetime* values will not be set. The *carid* value may or may not be set, depending on the actual venue. Restaurants, for example, will populate the *carid* value with some reference to Guests’ table locations.

### ABANDON Message

The ABANDON message is used in the Attraction model. The ABANDON message is used to identify guests that have not been reported at any reader in a venue for a configurable amount of time. Typically, this occurs when a guest has left a venue prematurely, perhaps ducking under a queue rope.

<message type=”ABANDON*”* time=”*timestamp*”>

<guestid>*guest id*</guestid>

<publicid>*public id of card or band*</publicid>

<linkid>*link id value*</linkid>

<linkidtype>[*xbms-link-id | gxp-link-id | fidelio-link-id | other*]</linkidtype>

<bandtype>[*Guest | Attraction Card | Cast Member*]</bandtype>

<xpass*>true | false*</xpass>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*unique reader id*</readerdeviceid>

<lastxmit>*time*</lastxmit>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| guestid | xConnect IDMS system identifier of the guest that triggered the abandon message. |
| publicid | Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| linkid | Global identifier the guest. Typically the xBMS link identifier GUID of a guest. If the link identifier type order setting of the xBRC specifies an alternate preference, then the first type found and matched will be returned |
| linkidtype | Indicates the type of link identifier used in the linkid. Typically “xbms-link-id”, or “fidelio-link-id”, or “gxp-link-id” depending on link identifier type preference order match. |
| bandtype | Value used to indicate the role of the card or band which was used to identify the owner. Typically “Guest”, other forms may include “Attraction Card”, “TEST”, “Cast Member” |
| xpass | “true” if guest entered the venue through an xPass queue. “false” if through standby queue |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “entry”, “load”, etc.). The names are unique per defined location. |
| readername | Identifies the individual reader (by name) originating the event. |
| lastxmit | Specifies the time (in ISO 8601 format) when the last activity was noted from the band |
| readerdeviceid | Unique reader device id |

### METRICS Message

<message type=”METRICS*”* time=”*timestamp*”>

<starttime>*time*</starttime>

<endtime>*time*</endtime>

<standby>

<guests>*guest count*</guests>

<abandonments>*abandonment count*</abandonments>

<waittime>*wait time*</waittime>

<totaltime>*total time*</totaltime>

</standby>

<xpass>

<guests>*guest count*</guests>

<abandonments>*abandonment count*</abandonments>

<mergetime>*merge time*</mergetime>

<waittime>*wait time*</waittime>

<totaltime>*total time*</totaltime>

</xpass>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| starttime | Time (ISO 8061) format when the statistics started to be collected. |
| endtime | Time when statistics stopped being collected |
| guests | Count (integer) of Guests that entered the venue during the period. |
| abandonments | Count (integer) of Guests that abandoned the venue during the period. |
| waittime\* | Specifies, in seconds, the time the Guest spent waiting to load the venue’s “ride” (from “entry”) |
| mergetime\* | Specifies, in seconds, the average time Guests spent waiting to reach the merge point. Note that this data is only relevant (and present) for xPass data. |
| totaltime | Specifies, in seconds, the average total time Guests spent in the venue |

\*The format of the METRICS message varies depending on whether the xBRC is configured to use the Attraction or Controlled Space model. In the latter case, only the *xpass* section is populated and only a subset of the data is present in that section. Specifically, neither *waittime* nor *mergetime* is present.

The METRICS message is sent out on a periodic basis. By default, this period will be every 10 minutes but a mechanism will be provided to change this time value. The METRICS message provides xBRC-calculated statistics summarizing the operational performance of an attraction. The *startime* and *endtime* elements identify the time period summarized by the message. The *standby* and *xpass* elements contain elements summarizing the operational performance of the standby and xPass queues. Data are provided for guest counts and abandonments as well as for wait times, total times and merge times (xPass queue only).

### TAPPED Message

The TAPPED message is used in the Park Entry model. The TAPPED message is used to identify guests that have reported to a reader in one of the locations of a given park entry venue.

<message type=”TAPPED*”* time=”*timestamp*”>

<reason>*reason code*</reason>

<xbrcreferenceno>*reference id of card or band*</xbrcreferenceno>

<publicid>*public id of card or band*</publicid>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*unique reader id*</readerdeviceid>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| reason | Reason code for message. One of <blank>, TIMEOUT, SWITCHEDREADER, SCANFAILED, ENTITLEMENTFAILED, IDCHECKREQ |
| xbrcreferenceno | xConnect xBRC system identifier of the media that triggered the abandon message. Do not use this internal field used by the xBRC.  Ex. <xbrcreferenceno>?RFID=MD2C81364312657654</xbrcreferenceno> |
| publicid | I Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “MK V 1”, “HS V 5”, etc.). The names are unique per defined location. |
| readername | Identifies the individual touch reader (by name) originating the event. |
| readerdeviceid | Unique reader device id |

### ABANDONED Message

The ABANDONED message is used in the Park Entry model. The ABANDONED message is used to identify guests that have reported to a reader in one of the locations of a given park entry venue.

<message type=”ABANDONED*”* time=”*timestamp*”>

<reason>*reason code*</reason>

<xbrcreferenceno>*reference id of card or band*</xbrcreferenceno>

<publicid>*public id of card or band*</publicid>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*unique reader id*</readerdeviceid>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| reason | Reason code for message. One of <blank>, TIMEOUT, SWITCHEDREADER, SCANFAILED, ENTITLEMENTFAILED, IDCHECKREQ |
| xbrcreferenceno | xConnect xBRC system identifier of the media that triggered the abandon message. Do not use this internal field used by the xBRC.  Ex. <xbrcreferenceno>?RFID=MD2C81364312657654</xbrcreferenceno> |
| publicid | I Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “MK V 1”, “HS V 5”, etc.). The names are unique per defined location. |
| readername | Identifies the individual reader (by name) originating the event. |
| readerdeviceid | Unique reader device id |

### BLUELANE Message

The BLUELANE message is used in the Park Entry model. The BLUELANE message is used to identify guests that have reported to a reader in one of the locations of a given park entry venue.

<message type=”BLUELANE*”* time=”*timestamp*”>

<reason>*reason code*</reason>

<xbrcreferenceno>*reference id of card or band*</xbrcreferenceno>

<publicid>*public id of card or band*</publicid>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>reader name</readername>

<readerdeviceid>*unique reader id*</readerdeviceid>

<sequence>*sequence number*</sequence>

<enterrorcode>error code as returned by TOR</enterrorcode>

<enterrordescription>error description as returned by TOR </enterrordescription>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| reason | Reason code for message. One of <blank>, TIMEOUT, SWITCHEDREADER, SCANFAILED, ENTITLEMENTFAILED, IDCHECKREQ |
| xbrcreferenceno | xConnect xBRC system identifier of the media that triggered the abandon message. Do not use this internal field used by the xBRC.  Ex. <xbrcreferenceno>?RFID=MD2C81364312657654</xbrcreferenceno> |
| publicid | I Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “MK V 1”, “HS V 5”, etc.). The names are unique per defined location. |
| readername | Identifies the individual touch reader (by name) originating the event. |
| readerdeviceid | Unique reader device id |
| enterrorcode | Entitlement error code as returned from TOR |
| enterrordescription | Entitlement error description as returned from TOR |

### HASENTERED Message

The HASENTERED message is used in the Park Entry model. The HASENTERED message is used to identify guests that have reported to a reader in one of the locations of a given park entry venue.

<message type=”HASENTERED*”* time=”*timestamp*”>

<reason>*reason code*</reason>

<xbrcreferenceno>*reference id of card or band*</xbrcreferenceno>

<publicid>*public id of card or band*</publicid>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>reader name</readername>

<readerdeviceid>*unique reader id*</readerdeviceid>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| reason | Reason code for message. One of <blank>, TIMEOUT, SWITCHEDREADER, SCANFAILED, ENTITLEMENTFAILED, IDCHECKREQ |
| xbrcreferenceno | xConnect xBRC system identifier of the media that triggered the abandon message. Do not use this internal field used by the xBRC.  Ex. <xbrcreferenceno>?RFID=MD2C81364312657654</xbrcreferenceno> |
| publicid | I Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “MK V 1”, “HS V 5”, etc.). The names are unique per defined location. |
| readername | Identifies the individual touch reader (by name) originating the event. |
| readerdeviceid | Unique reader device id |

### PARKENTRYEVENT Message

The PARKENTRYEVENT message is used in the Park Entry model. The PARKENTRYEVENT message is used to identify a guest who is being sensed by one of the entry reader locations for a given xBRC. For example: any one of several long range readers at park entry will trigger a PARKENTRYEVENT as they approach the readers, when they walk away from this set of entry locations the PARKENTRYABANDON message will signal this when no longer detected.

<message type=”PARKENTRYEVENT*”* time=”*timestamp*”>

<publicid>*public id of card or band*</publicid>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*reader device id*</readerdeviceid>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| publicid | Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| xbrcreferenceno | xConnect xBRC system identifier of the media that triggered the abandon message. Do not use this internal field used by the xBRC.  Ex. <xbrcreferenceno>?RFID=MD2C81364312657654</xbrcreferenceno> |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “waypoint1”, “waypoint2”, etc.). The names are unique per defined location. |
| readername | Identifies the individual reader (by name) originating the event. |
| readerdeviceid | Unique reader device id |

### PARKENTRYABANDON Message

The PARKENTRYABANDON message is used in the Park Entry model. The PARKENTRYABANDON message is used to identify a guest who was previously being sensed at one of the park entry locations is no longer being sensed by any. For example: any one of several long range readers at park entry will trigger a PARKENTRYEVENT as they approach the readers, when they walk away from this set of entry locations the PARKENTRYABANDON message will signal this when no longer detected.

<message type=”PARKENTRYABANDON*”* time=”*timestamp*”>

<publicid>*public id of card or band*</publicid>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*unique reader id*</readerdeviceid>

<lastxmit>*time*</lastxmit>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| publicid | Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| xbrcreferenceno | xConnect xBRC system identifier of the media that triggered the abandon message. Do not use this internal field used by the xBRC.  Ex. <xbrcreferenceno>?RFID=MD2C81364312657654</xbrcreferenceno> |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “entry”, “load”, etc.). The names are unique per defined location. |
| readername | Identifies the individual reader (by name) originating the event. |
| readerdeviceid | Unique reader device id |
| lastxmit | Specifies the time (in ISO 8601 format) when the last activity was noted from the band |

### BANDSTATUS Message

The BANDSTATUS message is used in both the Attraction and Park Entry models. The BANDSTATUS message is used to identify a guest who tapped at a reader location, but was not also sensed by a corresponding LRR reader for the same venue location. This message is intended to identify the possibility of MagicBands that are not functioning for long range reader scenarios due to improper wake-up command transmission, low or dead batteries, or other malfunctions internal to the band devices. For example: A guest with a MagicBand that is a few years old arrives at the park or an attraction and taps to check for entitlement, since the xBRC is aware of long range readers for the same xi and/or park entrance detection near the given reader it would verify that the same public ID for the media that was tapped, is also one that was recently read from the LRR readers. Since this older band was unable to wake up, but behaved normally for the tap reader, a BANDSTATUS message would be sent indicating a health concern. Each of these messages are not absolute indicators of bad health, they should be treated as a symptom of sickness or inability to read for some reason at that time.

Note: Card media tapping at a given venue will produce a BANDSTATUS message, this is expected for the interim period till MagicBands are fully deployed. Downstream application use of this message, such as xBMS should check the type of media that the public ID in the given message and discards known cards. Those BANDSTATUS messages that actually apply to MagicBands should increment a counter to track the number of occurrences of this condition.

<message type=”BANDSTATUS*”* time=”*timestamp*”>

<publicid>*public id of card or band*</publicid>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*reader device id*</readerdeviceid>

<rfid>*rfid of card or band*</rfid>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| publicid | Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “waypoint1”, “waypoint2”, etc.). The names are unique per defined location. |
| readername | Identifies the individual reader (by name) originating the event. |
| readerdeviceid | Unique reader device id |
| rfid | The RFID of the band that generated the touch event. Note that if the band is not assigned, that the value of the guestid will be empty while this element will still report the band’s RFID. |

### LOCATIONEVENT Message

The LOCATIONEVENT message is used in the Attraction and Space models. The LOCATIONEVENT message is used to identify a guest who is being sensed at a reader location. For example: if a long range reader is used at a waypoint location, a guest may be reported by a LOCATIONEVENT as they approach the reader, when they walk away from this reader location the LOCATIONABANDON message will signal this when no longer detected. If the LOCATIONEVENT was from a tap reader, no LOCATIONABANDON messages would follow as these are only used in conjunction with long range reader events.

<message type=”LOCATIONEVENT*”* time=”*timestamp*”>

<guestid>*guest id*</guestid>

<publicid>*public id of card or band*</publicid>

<linkid>*link id value*</linkid>

<linkidtype>[*xbms-link-id | gxp-link-id | fidelio-link-id | other*]</linkidtype>

<bandtype>[*Guest | Attraction Card | Cast Member*]</bandtype>

<xpass>*true | false*</xpass>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*reader device id*</readerdeviceid>

<rfid>*band RFID*</rfid>

<confidence>*[0-100]*</confidence>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| guestid | xConnect IDMS system identifier of the guest that triggered the reader event message. |
| publicid | Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| linkid | Global identifier the guest. Typically the xBMS link identifier GUID of a guest. If the link identifier type order setting of the xBRC specifies an alternate preference, then the first type found and matched will be returned |
| linkidtype | Indicates the type of link identifier used in the linkid. Typically “xbms-link-id”, or “fidelio-link-id”, or “gxp-link-id” depending on link identifier type preference order match. |
| bandtype | Value used to indicate the role of the card or band which was used to identify the owner. Typically “Guest”, other forms may include “Attraction Card”, “TEST”, “Cast Member” |
| xpass | “true” if guest entered the venue through an xPass queue. “false” if through standby queue |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “waypoint1”, “waypoint2”, etc.). The names are unique per defined location. |
| readername | Identifies the individual reader (by name) originating the event. |
| readerdeviceid | Unique reader device id |
| rfid | The RFID of the band that generated the touch event. Note that if the band is not assigned, that the value of the guestid will be empty while this element will still report the band’s RFID. |
| confidence | A percentage value that indicates the calculated probability that a user is at the given reader. |

### LOCATIONABANDON Message

The LOCATIONABANDON message is used in both the Attraction and Space models. The LOCATIONABANDON message is used to identify a guest who was previously being sensed at a reader location is no longer being sensed at that location. For example: if a long range reader is used at a waypoint location, a guest may be reported by a LOCATIONEVENT as they approach the reader, when they walk away from this reader location the LOCATIONABANDON message will signal this when no longer detected. If the LOCATIONEVENT was from a tap reader, no LOCATIONABANDON messages would follow as these are only used in conjunction with long range reader events. For clarification, when the guest is considered to be no longer at a venue at all, and they did not exit or leave normally, then the ABANDON message would be issued.

<message type=”LOCATIONABANDON*”* time=”*timestamp*”>

<guestid>*guest id*</guestid>

<publicid>*public id of card or band*</publicid>

<linkid>*link id value*</linkid>

<linkidtype>[*xbms-link-id | gxp-link-id | fidelio-link-id | other*]</linkidtype>

<bandtype>[*Guest | Attraction Card | Cast Member*]</bandtype>

<xpass*>true | false*</xpass>

<locationid>*reader location identifier*</locationid>

<readerlocation>*reader location name*</readerlocation>

<readername>*reader name*</readername>

<readerdeviceid>*unique reader id*</readerdeviceid>

<lastxmit>*time*</lastxmit>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| guestid | xConnect IDMS system identifier of the guest that triggered the abandon message. |
| publicid | Identifies the card or band used at the location of the touch or long range read event for this location. In long range reads, the band’s provisioned LRID is the public ID (pid). This value is in decimal. |
| linkid | Global identifier the guest. Typically the xBMS link identifier GUID of a guest. If the link identifier type order setting of the xBRC specifies an alternate preference, then the first type found and matched will be returned |
| linkidtype | Indicates the type of link identifier used in the linkid. Typically “xbms-link-id”, or “fidelio-link-id”, or “gxp-link-id” depending on link identifier type preference order match. |
| bandtype | Value used to indicate the role of the card or band which was used to identify the owner. Typically “Guest”, other forms may include “Attraction Card”, “TEST”, “Cast Member” |
| xpass | “true” if guest entered the venue through an xPass queue. “false” if through standby queue |
| locationid | Identifies the location (a administrator configured string ID value) associated with the reader location. |
| readerlocation | Identifies the reader(s) reporting the event. Typically, there will be multiple readers “ganged” in each reader location. The *readerlocation* element identifies the abstract location name of the readers (e.g. “entry”, “load”, etc.). The names are unique per defined location. |
| readername | Identifies the individual reader (by name) originating the event. |
| readerdeviceid | Unique reader device id |
| lastxmit | Specifies the time (in ISO 8601 format) when the last activity was noted from the band |

### DISCOVERY Message

The DISCOVERY message is used in all of the XBRC models. The DISCOVERY message is used to identify the presence of the XBRC service host. The XBRMS service of the xConnect solution uses this message to automatically update its list of known services, and monitor the role, model, and venue name that the service represents.

<message type=”DISCOVERY*”* time=”*timestamp*”>

<name>*xBRC name*</name>

<port>*xBRC service REST interface port number*</port>

<venue>*xBRC venue name*</venue>

<model>*com.disney.xband.xbrc.[parkentrymodel|spacemodel|attractionmodel].CEP*</model>

<haStatus>*master | slave*</haStatus>

<configurationChangedTime>*time*</configurationChangedTime>

<ip>*xBRC IP address*</ip>

<hostname>*xBRC host name*</hostname>

<remoteSVUID>*1*</remoteSVUID>

<discoveryInterval>*time in seconds*</discoveryInterval>

<sequence>*sequence number*</sequence>

</message>

|  |  |
| --- | --- |
| Element | Purpose |
| name | Name of xBRC service that the discovery message is from |
| port | Service port number the xBRC service supports REST interface access |
| venue | Venue name of the xBRC |
| model | Model type of the xBRC, usually one of Space, Attraction, or Park Entry |
| haStatus | xBRC high availability status for the given xBRC. When two xBRCs are teamed together to service a given purpose, one of them would act as the Master xBRC and the other would assume a Slave (backup) role. |
| configurationChangedTime | Date and Time of last xBRC configuration update |
| ip | IP address of the computer running the xBRC service |
| hostname | Hostname of the computer running the xBRC service |
| remoteSVUID | Serial version of this object on the remote system (set by publisher only) |
| discoveryInterval | Time in seconds that the xBRC service will use to send next discovery messages |

## JMS Transport Details

The xBRC will use a “publish and subscribe” endpoint called *com.disney.xbrc* on a configured Sonic MQ message broker. It will send *text* messages to this endpoint containing the XML described in this document. Three JMS string properties will be set on all messages:

|  |  |
| --- | --- |
| JMS string property | Description |
| *xbrc\_facility* | Identifies the facility that is sending the message. Typically, this value is a One Source originated entertainment id. |
| *xbrc\_facility\_type* | Identifies the model in use by the xBRC. This is a “short” form, not a full Java type. The current values are “attractionmodel”, “parkentrymodel” and “spacemodel”. |
| *xbrc\_message\_type* | Identifies the message type of the message. |

## Using Multiple xBRCs at a Facility

The general design of the xBRC is optimized with the assumption that a single xBRC will be used for a facility. The value of the configured *venue* property is used to identify an xBRC when communicating with other systems. This is also the value used in the <venue> tag described in this section and in the *xbrc\_facility*property described in section 5.5. Generally, the property’s value is set to the OneSource entertainment id associated with the facility.

In certain use cases, however, it may be necessary to use multiple xBRCs at a particular facility. Some facilities, for example, may use a space-model xBRC to handle interactive guest activities while using a different attraction-model xBRC to handle Fastpass+ queues. Other facilities (for example, large restaurants) may need multiple space-model xBRCs to handle hundreds of readers. In these scenarios, it is important that the venue property be carefully configured in each xBRC to avoid problems. Each xBRC must have a unique venue setting. Here are the rules that must be followed:

If multiple xBRCs are used, their *venue* properties should be set using the form, “XXXXX-#” where “XXXXX” is the OneSource entertainment id for the facility and “-#” (e.g. “-1”, “-2”) is a suffix used to distinguish the xBRCs.

When an attraction-model xBRC interacts with GxP, it will strip off any non-numerical suffix (i.e. the “-#”) in its communications.

When an xBRC sends messages using HTTP (instead of JMS), it will strip off the non-numerical suffix when setting the *name* attribute in the <venue> tag.

When the xBRMS enumerates its known facilities via its web services, it will strip off non-numerical suffixes. As this may result in non-unique venue settings, clients of xBRMS must use other mechanisms (for example, the configured xBRC model) to disambiguate them.

Messages sent through JMS are not stripped of their suffixes. Consequently, multiple xBRCs will appear as multiple facilities in the database tables populated by JMSListener.

# HTTP Update Stream use case scenarios

## Sequence Diagrams

The following sequence diagram describes the general operation of the xBRC in http update stream scenarios:



Annotations for Sequence diagram:

1. The xBRC asks the xBRMS for information to pre-seed its band id cache. This cache allows the xBRC to associate a touch (RF) or long-range ID with a particular guest ID.

2. The Band Location Service retrieves location information from the xBRC. The xBRC responds with a description of each of installed xBR or xTP readers.

3. The Band Location Service pre-seeds its table of guest locations by querying the xBRC. The xBRC responds with a list of Guests and their locations.

4. The Band Location Service asks the xBRC to send Guest location changes to a named URL.

5. As Guests move inside a monitored space, the xBRC informs the Band Location Service by invoking HTTP PUT operations to the URL identified in step 4.

# Park Entry

The Park Entry scenario represents a specialized implementation of the xBRC with a unique workflow and validation partners for touch events that are found nowhere else in the park. Park Entry is a three step process of validating the entitlement, validating a biometric (fingerprint) parameter associated with the guest id and incrementing the entitlement counter. In addition to the transactions validating the entitlement, the entire process is monitored by cast members with a hand held device.  
  
Park entry xBRC implementation is documented in the “xBRC Park Entry Model.docx” document.

## GXP Interaction

This section details how the xBRC interacts with GXP in order to validate and consume entitlements. This process is initiated by touching an xTP reader and is concluded by the light on the reader displaying green (successful status) or blue (further action required). In addition to xBRC and GXP, the process involves a separate cast application that allows the cast member staffing the reader to resolve the blue light status by either denying access or by requesting an “override” to the xPass entitlement system.

The diagrams on the next page (borrowed from Disney documents) depict the interactions between the various components. The key interactions explored in this document are those identified as step ➁ in the first diagram and step ➇ in the second. Step ➁ describes how the xBRC interacts with the GXP in order to validate/redeem entitlements while step ➇ describes how the cast application/GXP can reset the status of an xTP after a “blue light” scenario.

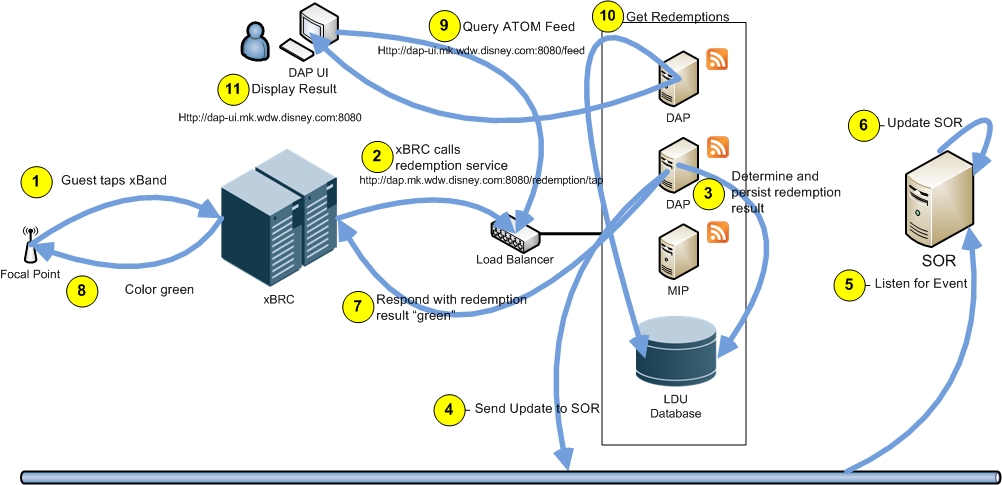


Figure - Successful xPass Redemption

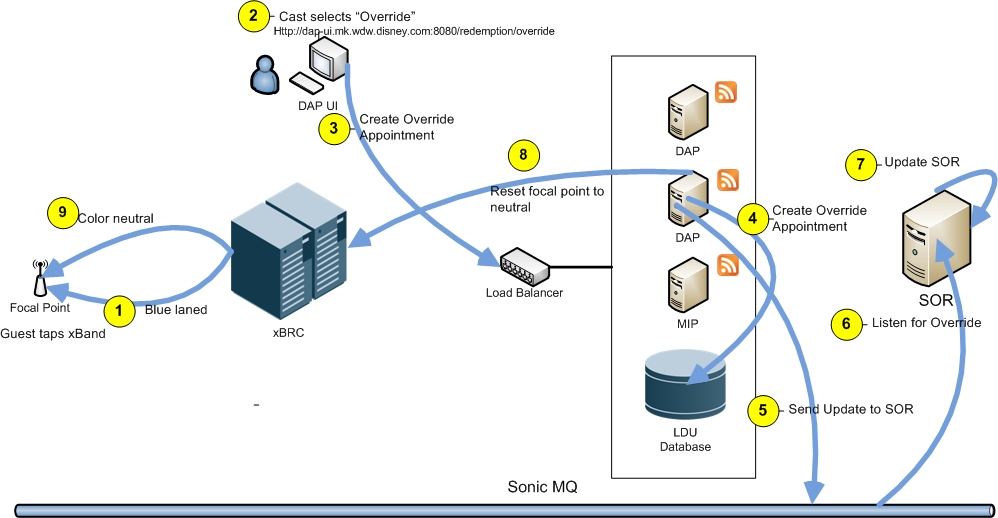


Figure - Unsuccessful Redemption with Override

## xBRC-to-GXP xPass Redemption Interaction

The xBRC will be configured with an HTTP URL identifying the GXP DAP (s) to use for xPass interactions. When a guest touches an xTP designated as requiring xPass interaction (for example, an “xpassentry” or “merge” reader in an attraction), it performs an HTTP POST to the configured URL. The path for the POST will be “/tap” and the payload will be of the form:

{  
 “tapRequest”:

{  
 “tapDate”: *timestamp*,  
 “xBandId”: *band id*,  
 “location”: *location id*,  
 “unitType”: *unit type*,  
 “entertainmentId”: *facility code*,  
 “side”: *side*,  
 “callback”: *url*   
 }  
}

|  |  |
| --- | --- |
| Element | Purpose |
| tapDate | Timestamp of touch in ISO 8601 UTC format. |
| xBandId | The band id used to touch the reader. |
| location | The location where the touch occurred. This is a number assigned by One Source |
| unitType | “Entry”, “Merge” or “Combo”. The “Combo” type is used when there is a single station in a location. Note that station types may be reconfigured dynamically as mentioned below. |
| entertainmentId | The facility code where the touch occurred. A number assigned by One Source. |
| side | “Left” or “Right”. For use by the associated cast application |
| callback | The URL that should be used by the cast application in order to change the state of the light on the originating reader |

Note that *callback* might refer to a specific xBRC or might address the venue’s xBRMS. In the case of the latter, the example might look more like “xbrms.wdw.disney.com/ /kimp/light/norway-entrance-left”.

The interpretation of the “/tap” POST is up to the GXP/DAP. In general, however, the GXP will interpret the message according to the *stationtype* field. If the value of this field is “Entry”, the GXP will interpret the request as an entitlement verification. If the field is set to “Merge” then the request will be interpreted as an entitlement redemption. Finally, if the value is “Combo” then the request will perform entitlement redemption while still allowing a cast member override as in the verification step. Note that the details of these interactions are more fully described in Disney GXP documents.

The GXP’s response to the POST /tap request will look like:

{  
 “tapResponse”:  
 {  
 “green” : *boolean,* “reason” : *string*,  
 … other fields …  
 }  
}

|  |  |
| --- | --- |
| Element | Purpose |
| green | *True* if the light should be turned green; *False*  if blue. |
| reason | Readable cause of an “invalid” *state* |
| … other fields … | Ignored by the xBRC. |

## GXP-to-xBRC Light Control

In the request payload described above, the xBRC provides a URL that allows GXP cast applications (indirectly through a GXP DAP) to communicate back with an xBRC (perhaps through the xBRMS) in order to control the light on an xTP. The GXP DAP performs an HTTP POST to this URL with the following payload:

{  
 “stateChange”:  
 {  
 “timestamp”: 183641341322,  
 “status”: “Reset”  
 “timeout”: 0  
 }  
}

|  |  |
| --- | --- |
| Element | Purpose |
| timestamp | Milliseconds since the start of the epoch (midnight, January 1st, 1970) |
| state | “Reset” to turn off all lights. “Green” to turn the light green. “Blue” to turn the light blue. |
| timeout | Number of milliseconds that the light should remain in the specified state (only relevant for “Green” and “Blue”). 0 indicates that the light should remain in that state indefinitely. |

## Reconfiguring xTP Readers

Not all venues (facilities) will have both xPass entry and merge readers. Some may have a single “combo” reader location. Additionally, some venues may have multiple xPass entry locations (with a single merge location). Finally, in some venues, the configuration of reader locations may be changed “on-the-fly”. For example, a venue may start out with a single “combo” reader location but may then switch to two locations, “entry” and “merge”.

Attraction cast members will be able to change the venue’s (facility’s) reader configuration by an application that will enumerate the available configurations (by name) and will allow them to select the desired one. In order to support these operations, the xbrc will implement two HTTP RESTful paths:

### Get configuration

This path will return the available configurations

{  
 “configurations”:  
 [  
 {   
 “configurationId” : *number*,  
 “name”: *string,*  
 “description” : *string*  
 },  
 {   
 “configurationId” : *number*,  
 “name”: *string,*  
 “description” : *string*  
 },  
 …

]  
}

|  |  |
| --- | --- |
| Element | Purpose |
| configurationId | Numeric identifier used to refer to the configuration |
| name | Name of the configuration. |
| description | Human readable description of the configuration. |

### PUT selectconfiguration/name/<name>

**PUT selectconfiguration/id/<id>**

These requests replace the current configuration with a stored configuration identified by its name or by its assigned id.

# Reader Registration

This section describes how readers (long range, focal point and others) register themselves with an xBRC. Typically, this process starts out by registering a reader with a DHCP server. The DHCP server is configured with the network MAC address of the reader and with the IP address of the xBRC with which that reader is supposed to communicate. When the reader is powered on, it looks for a DHCP server, obtains an IP address lease but also gets configuration information from the DHCP server. The configuration information includes the URL of the xBRC with which the reader is supposed to communicate.

Once the reader knows its xBRC, it then performs an HTTP “PUT hello” request to that xBRC. The payload of the operation is a JSON object that looks like this:

{  
 “mac” : *string,*  
 “port” : *number*,  
 “next eno”: *number*,  
 “reader name”: *string*,  
 “reader type”: *string*,  
 “reader version”: *string,* “min xbrc version”: *string,* “linux version” : *string,*  
 “location id” : *number*  
}

The meaning of these elements is as follows:

|  |  |
| --- | --- |
| Element | Purpose |
| mac | The mac address of the reader (in “XX:XX:XX:XX:XX:XX” form” |
| port | The port number that the reader listens to for incoming HTTP requests. |
| next eno | The next “event number” that will be used by the reader when sending events to the xBRC |
| reader name | The name of the reader |
| reader type | The type of the reader. Currently, the known types are “Long Range”, “xTP”, “xFP+xBIO” and “Mobile Gxp” |
| reader version | (optional) A software version string of the form “a.b.c.d” |
| min xbrc version | (optional) A string of the form “a.b.c.d” identifying the minimum version of the xBRC software expected by the reader. |
| linux version | (optional) A string identifying the version of operating system used by the reader. |
| location id | (optional) The numeric id of the location where the reader is operating. |

When a reader registers itself with an xBRC by sending the PUT hello request, if the xBRC does not “know” the reader (in other words, it has not been configured to anticipate the reader registration), the xBRC will associate the reader with the “UNKNOWN” location id. Note that the reader can override this behavior by explicitly stating a “location id” element.

In addition to letting the xBRC know that a reader is attaching itself, the PUT hello request also serves other purposes. First, the xBRC will perform a software consistency check by analyzing the reported “reader version” and “min xbrc version” strings. If the xBRC has newer software available for the reader (and the reader supports automatic software updates), the xBRC will send an HTTP request to the reader informing it of the available software and allowing it to request the software for automatic update. Second, the xBRC will also respond to the PUT hello request by updating the reader’s clock (again, by sending it an HTTP request, but only if the reader supports such a request). Finally, the xBRC may also perform other reader type-specific operations to initialize the reader.

Long range and focal point readers (with or without biometric readers) listen to the designated *port* for incoming HTTP requests from the xBRC. Other reader types (name, Mobile Gxp) do not. In the former case, the xBRC further responds to the PUT hello request by sending an HTTP POST update\_stream request back to the reader. This request tells the reader to send subsequent event information to a designated URL in the xBRC. In the case of Mobile Gxp readers, these readers send events by performing HTTP PUT stream requests to the xBRC.

Regardless of how readers are initialized, they ultimately send events to the xBRC by encoding them in JSON format and PUTing them to the “stream” path. The JSON formatted events look like:

{  
 “reader name”: *string*,  
 “events”:  
 [  
 {  
 “type”: *string*,  
 “eno”: *number*,  
 “time”: *ISO 8661 date/time string*,  
 … type specific data…   
 },  
 … more events…  
 ]  
}

The meaning of these elements is as follows:

|  |  |
| --- | --- |
| Element | Purpose |
| reader name | The name of the reader |
| type | The type of event. These include “LRR”, “RFID”, and several xBIO related types. |
| eno | A monotonically increasing event number |
| time | A GMT time/date encoded in ISO 8661 string format |

Type-specific data varies depending on the *type* of the event. For RFID data, for example, an event looks like:

{  
 “type”: *string*,  
 “eno”: *number*,  
 “time”: *ISO 8661 date/time string*,  
 “uid”: *string*,  
 “pid”: *string*,  
 “sid”: *string*,  
 “iin”: *string*  
}

Where the last four elements are:

|  |  |
| --- | --- |
| Element | Purpose |
| uid | The universal id of the band/card being touched. |
| pid | (optional) The public id of the band (will be set to the long range id of the band). |
| sid | (optional) The secure id of the band. |
| iin | (optional) The “issuer identification number” associated with the secure id. Disney is not presently using this field. |

## Automatic Reader Software Updates

The xBRC is capable of pushing new reader software to dap and long range readers. The reader software is one or more software package files with the \*.ipk file extension. The xBRC maintains a repository of these files in its web directory, typically in /usr/share/xbrc/www. Currently there are two methods of pushing new software to the readers, one using the /install restful reader call and second using the /upgrade restful reader call. The /install method pushes a single \*.ipk file to the reader, while the /upgrade method pushes a manifest file to the reader causing the reader to download multiple files from the xBRC and then install them.

### How does the xBRC choose which version to install

The xBRC will automatically locate all the \*.ipk, or manifest files in its repository and choose the latest version to push to the reader. A new version will be pushed if the reader version received in the hello message from the reader does not match the latest package version in the xBRC repository. If the reader version is greater than the latest version in the xBRC repository then the reader will be downgraded to the latest version in the xBRC repository.

### Minimum xBRC version

If the “min xbrc version” sent to the xBRC from the reader in the hello message is greater than the currently installed version of the xBRC, the xBRC will not process events from the reader and will report in the xBRC status that there is a problem with the reader version.

### How does the xBRC choose which install method to use

As described earlier, the xBRC can push new software to the readers either using the /install or /upgrade restful calls. The choice is based on the presence of the /usr/share/xbrc/www/packages directory or /usr/share/xbrc/www/repos directory or both. If the /usr/share/xbrc/www/packages directory exists then /install will be called. If the /usr/share/xbrc/www/repos directory exists then /upgrade will be called. If both directories exist then priority is given the /upgrade method.

### Uploading new reader packages to the xBRC

There are four RPM packages that contain the reader software.

xbr-0.0.0-0.rpm

xreader-packages-0.0.0-0.rpm

xTP-0.0.0-0.rpm

xTP2-0.0.0-0.rpm

The version will vary depending on the actual version of the software being installed. The first three packages above install individual \*.ipk files used with the /install method while the last package xreader-packages installs multiple \*.ipk files including manifest files using with the /upgrade method.

If the xBRC is running while the packages are installed, then the restful call http://<host>:8080/refreshpackages must be called to cause the xBRC to refresh the list of available packages.

# Scheduling

Scheduling is a mechanism for power management of battery powered devices, like an xBRv4 with battery.

A single schedule is currently permitted and may be defined using the properties interface.

Time is specified in UTC using the yyyy-MM-dd'T'HH:mm:ss.SSS date format, with only the hours, minutes, and seconds, and milliseconds used. Both a start (readerPowerOnTime) and end (readerPowerOffTime) time must be specified for scheduling to work. The readerPowerOffPeriod defines the interval for which the device is powered off. Each time the interval expires, the device will turn on, check in with the xBRC, and potentially be told to turn off again.

Here is an example that schedules devices to be on for one hour between 9am UTC and 10am UTC each day, checking every 5 minutes (which is excessive – 30 minutes to 60 minutes is recommended):

<?xml version="1.0" encoding="UTF-8" standalone="yes"?><venue id="xBRC" name="xBRC" time="2013-02-21T01:41:15.927"><configuration>

<config><name>readerPowerOnTime</name><value>1900-01-01T09:00:00.000</value><type>java.lang.String</type><min>N/A</min><max>N/A</max><choices>N/A</choices><description>Date string (yyyy-MM-dd'T'HH:mm:ss.SSS) (only time is significant) representing the time when readers are to be powered on. empty = always on</description><updatable>true</updatable><clazz>com.disney.xband.xbrc.Controller.model.ControllerInfo</clazz><configClass>ControllerInfo</configClass></config>

<config><name>readerPowerOffTime</name><value>1900-01-01T10:00:00.000</value><type>java.lang.String</type><min>N/A</min><max>N/A</max><choices>N/A</choices><description>Date string (yyyy-MM-dd'T'HH:mm:ss.SSS) (only time is significant) representing the time when readers are to be powered off. empty = always on</description><updatable>true</updatable><clazz>com.disney.xband.xbrc.Controller.model.ControllerInfo</clazz><configClass>ControllerInfo</configClass></config>

<config><name>readerPowerOffPeriod</name><value>30</value><type>int</type><min>5</min><max>120</max><choices>N/A</choices><description>Duration in minutes that a reader sleeps before waking up to check in with the xBRC before (potentially) going back to sleep.</description><updatable>true</updatable><clazz>com.disney.xband.xbrc.Controller.model.ControllerInfo</clazz><configClass>ControllerInfo</configClass></config>

</configuration></venue>