Electric Mayhem xTP

Interface Control Document

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Description** |
| 0.1 |  | GJS | First draft |
| 0.2 | 7/1/11 | GJS | GET events  Added more explanation to GET events  Added packet sequence number to read event information  Added radio frequency to read event information  Added a “max” parameter to GET events  PUSH update\_stream – limited to one URL  Time set via Network Time Protocol (NTP)  RID (reader ID) changed to reader-name  XID changed to XLRID (xBand Long Range ID)  Reader name defaults to MAC address |
| 0.3 | 7/8/11 | GJS | Added optional ‘max’ and ‘since’ parameters to POST update\_stream  Removed URL parameter from DELETE update\_stream  Added POST hello to xBRC  Added more text in introductory sections  Added ‘event number’ field. ‘since’ parameter now specifies event number instead of time. |
| 0.4 | 7/15/11 | GJS | Shortened tag names on events |
| 0.5 | 7/16/11 | GJS | “since” parameter changed to “after” and it acquires events after the given event number.  Changed time format to ISO 8601 standard (hh:mm:ss.SSS for milliseconds). |
| 0.6 | 7/29/11 | GJS | Merged GET reader/name.json and GET reader/version.json into a single service GET reader/info.json. The payload in POST hello matches the payload in GET reader/info.json. |
| 0.7 | 7/29/11 | GJS | Added xBR, xTP and Park Entry xTP sections for events and commands specific each type of reader. |
| 1 | 8/12/11 | GJS | Added “port” and “next eno” to info.json and hello payload.  Changed reader controller “hello” address to 192.168.0.2:8080 |
| 2 | 8/16/11 | GJS | Changed most POSTs to PUTs  Added a “type” field to all events  Added xBIO version to park-entry xTP info.json  Added park-entry commands for setting and playing sounds  Added command to set park-entry mode. |
| 3 | 9/6/11 | GJS | Changed PUT biometric/match data from multipart to json.  Changed “park-entry”reader type to “xFP+xBIO” |
| 4 | 9/23/11 | GJS | Added “tail” parameter to GET events.json  Removed “PUT event” and “DELETE event”  Added “num events”, “oldest event”, and “push URL” to GET info.json  Replaced PUT application/software with PUT upgrade command.  Added PUT biometric/firmware and a firmware upgrade completed event for the park entry tap reader. |
| 5 | 9/27/11 | MJW | Change fonts & paragraph layout.  Separated out “hello” address.  Added MAC format.  Removed GET events/last.json  Define xbands/commands.  Define xBR specific data for reader-info. |
| 6 | 10/5/11 | GJS | Added PUT biometric/cancel for xTP |
| 7 |  | GJS | Made the templates in the biometric/match command optional. |
| 8 | 10/19/11 | GJS | Updated reader/info message for park entry xTP |
| 9 | 10/26/11 | GJS | Added biometric/image URL to retrieve xBIO image data |
| 10 | 10/31/11 | GJS | Added xbio-scan-error event type for park entry DAP. |
| 11 | 12/2/11 | GJS | Replaced “mode” command with “biometric/options” and “rfid/options”.  Expanded “light” command to include both direct light control and lighting/sound sequences. |
| 12 | 12/5/11 | GJS | Added some new fields to xTP diagnostics event. |
| 13 | 12/20/11 | GJS | Added “max” temperature to GET temperature.  Added “temp status” to diagnostic event.  Added “PUT rfid/tap” command |
| 14 | 12/29/11 | GJS | Added tap/option commands |
| 15 | 1/13/12 | GJS | Updated xTP diagnostics event fields |
| 16 | 1/31/12 | GJS | Added public and secure IDs to RFID event. Changed “RFID” to “uid” in all events. Added rfid/option to turn secure IDs off. |
| 17 | 2/2/12 | GJS | Changed “PUT upgrade” to “PUT install” and added place holder for future “PUT upgrade” |
| 18 | 2/13/12 | GJS | Updated format of secure ID to match secure ID spec and added “iin” field to RFID events. |
| 19 | 3/14/12 | GJS | Added gxp\_success and gxp\_exception to list of available lighting sequences.  Added “iin” field to PUT rfid/tap command. |
| 20 | 3/29/12 | GJS | Removed “RFID” parameter from GET events. |
| 21 | 4/2/12 | GJS | Added “public\_id” to the rfid/options. |
| 22 | 5/4/12 | GJS | Added GET diagnostics. |
| 23 | 5/8/12 | GJS | Replaced xFP with xTP in appropriate places. |
| 24 | 6/12/2012 | GJS | Added “HW type” to reader/info and clarified meaning of “next eno”. |
| 25 | 7/16/2012 | CJW | Added “PUT xbrc” command. |
| 26 | 7/31/2012 | CJW | Added “PUT upgrade” command details. |
| 27 | 7/31/2012 | CJW | Added new /media/\* commands and removed obsoleted interfaces. Added section that describes the format for the media sequence scripts. |
| 28 | 8/3/2012 | CJW | Added “PUT events” command. Corrections to a few command names. |
| 29 | 9/11/2012 | CJW | Added details of statistical data on RFID read timing included in the xTP diagnostics message. |
| 30 | 9/19/2012 | CJW | Added “DELETE /xbrc” command. Updated “PUT /media/color” command. Added new media script commands. |
| 31 | 9/21/2012 | CJW | Corrected incorrect parameter in the “PUT /upgrade” command. |
| 32 | 9/24/2012 | GJS | Added clarification on when the diagnostics event is generated |
| 33 | 9/26/2012 | GJS | Added clarifications on contents of diagnostics message and use of PUT /application/reset. Corrected field names in GET /temperature and GET /amb\_light |
| 34 | 10/8/2012 | GJS | Added sequence name “off” to PUT /media/sequence and added PUT /media/off as two ways of turning off LEDs. |
| 35 | 10/9/2012 | CJW | Updated “PUT /reader/name” command.  Added section on how xBRC address is obtained.  Added section for “GET /update\_stream” command. |
| 36 | 10/12/2012 | CJW | Updated “GET /update\_stream” command.  Removed “GET /amb\_light” and “GET /temperature” commands. |
| 37 | 11/1/2012 | CJW | Added “GET /media/sequence” and “GET /reader/log” commands. |
| 40 | 11/7/2012 | GJS | Added clarification on the use of timeout parameter for media/color and media/sequence. |

Table of Contents

Contents

[1 Introduction 7](#_Toc339557098)

[1.1 Purpose 7](#_Toc339557099)

[1.2 Scope 7](#_Toc339557100)

[1.3 Background 7](#_Toc339557101)

[1.3.1 Park Entry xTP 7](#_Toc339557102)

[1.4 Assumptions and Constraints 9](#_Toc339557103)

[1.5 Definitions 9](#_Toc339557104)

[1.6 Reference 9](#_Toc339557105)

[2 General Formatting Notes 10](#_Toc339557106)

[2.1 Time 10](#_Toc339557107)

[2.2 MAC 10](#_Toc339557108)

[2.3 Payload Formats 10](#_Toc339557109)

[2.4 POST vs. PUT 10](#_Toc339557110)

[2.5 Controller (xBRC) Address 10](#_Toc339557111)

[3 xTP to xBRC Messages 12](#_Toc339557112)

[3.1 PUT ControllerServer/hello 12](#_Toc339557113)

[4 xTP Events 13](#_Toc339557114)

[4.1 Events Payload 13](#_Toc339557115)

[4.2 Event Types 14](#_Toc339557116)

[4.2.1 RFID Event 14](#_Toc339557117)

[4.2.2 Biometric read event 15](#_Toc339557118)

[4.2.3 Biometric Scan Error Event 15](#_Toc339557119)

[4.2.4 Biometric Image Event 16](#_Toc339557120)

[4.2.5 xTP Diagnostics Event 17](#_Toc339557121)

[4.2.6 xBIO Diagnostic Event 19](#_Toc339557122)

[4.2.7 xBIO FW Upgrade Event 19](#_Toc339557123)

[5 xTP Commands 20](#_Toc339557124)

[5.1 GET events 20](#_Toc339557125)

[5.2 PUT events 20](#_Toc339557126)

[5.3 PUT reader/name 20](#_Toc339557127)

[5.4 GET reader/info 21](#_Toc339557128)

[5.5 GET reader/log 22](#_Toc339557129)

[5.6 PUT update\_stream 22](#_Toc339557130)

[5.7 GET update\_stream 23](#_Toc339557131)

[5.8 DELETE update\_stream 23](#_Toc339557132)

[5.9 PUT application/reset 23](#_Toc339557133)

[5.10 PUT system/reset 23](#_Toc339557134)

[5.11 PUT install 23](#_Toc339557135)

[5.12 PUT upgrade 24](#_Toc339557136)

[5.13 PUT time 25](#_Toc339557137)

[5.14 GET time 25](#_Toc339557138)

[5.15 PUT rfid/tap 25](#_Toc339557139)

[5.16 PUT rfid/options 26](#_Toc339557140)

[5.17 GET rfid/options 26](#_Toc339557141)

[5.18 PUT tap/options 26](#_Toc339557142)

[5.19 DELETE tap/options 27](#_Toc339557143)

[5.20 GET tap/options 27](#_Toc339557144)

[5.21 GET diagnostics 27](#_Toc339557145)

[5.22 PUT xbrc 27](#_Toc339557146)

[5.23 DELETE xbrc 28](#_Toc339557147)

[6 Media interface 28](#_Toc339557148)

[6.1 GET media/inventory 28](#_Toc339557149)

[6.2 PUT media/package 28](#_Toc339557150)

[6.3 DELETE media/package 29](#_Toc339557151)

[6.4 PUT media/idle 29](#_Toc339557152)

[6.5 DELETE media/idle 29](#_Toc339557153)

[6.6 PUT media/sequence 29](#_Toc339557154)

[6.7 GET media/sequence 30](#_Toc339557155)

[6.8 PUT media/color 30](#_Toc339557156)

[6.9 PUT media/off 31](#_Toc339557157)

[6.10 PUT media/sound 31](#_Toc339557158)

[6.11 PUT media/ledscript 31](#_Toc339557159)

[6.12 PUT media/brightness 32](#_Toc339557160)

[7 Biometric Commands 32](#_Toc339557161)

[7.1 PUT biometric/enroll 32](#_Toc339557162)

[7.2 PUT biometric/match 32](#_Toc339557163)

[7.3 PUT biometric/cancel 32](#_Toc339557164)

[7.4 PUT biometric/firmware 32](#_Toc339557165)

[7.5 PUT biometric/options 33](#_Toc339557166)

[7.6 GET biometric/options 33](#_Toc339557167)

[7.7 GET biometric/image 33](#_Toc339557168)

[7.8 PUT biometric/image/send 33](#_Toc339557169)

[8 Media Sequence Script Specification 33](#_Toc339557170)

[8.1 Comments lines 34](#_Toc339557171)

[8.2 Commands 34](#_Toc339557172)

[8.3 Frames 34](#_Toc339557173)

[8.4 Example Script 34](#_Toc339557174)

# Introduction

## Purpose

Define interface between readers and xTP.

## Scope

This document describes the network interface to the xBand focal point readers (xTP). This includes both “normal” xTPs, and the park entry xTPs that include a biometric reader (finger print reader).

## Background

Each reader interfaces with one reader controller (xBRC), while a single xBRC may interface with multiple readers.

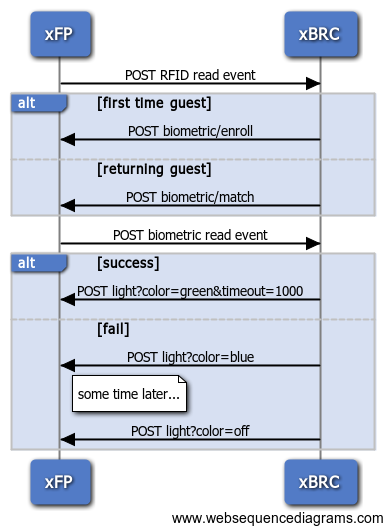
Readers acquire their IP address via DHCP if possible. If no DHCP server is found, then the readers default to IP address 169.254.0.2. This allows for direct connection of a single reader to a PC for debugging and configuration purposes.

The IP address of the xBRC associated with each reader must be configured in the reader and is stored in the file /mayhem/config.json.

The reader API uses a HTTP/REST based protocol with payloads formatted as JSON. Events can be pulled with GET commands or readers can be instructed to push events with PUT commands to a given URL.

### Park Entry xTP

The park entry xTP is similar to the standard xTP with the addition of a Lumidigm biometric finger print reader. Used at park entrances, guests will enter the park by tapping their xBand to the park entry xTP and placing their finger on the biometric reader. On first use, the system will enroll the guests by capturing finger print biometric data (called a template). On subsequent visits, the guest’s xBand RFID will be matched to the biometric template to verify whether the same guest is using the xBand. The following diagram shows the possible message sequences between the reader and the xBRC:



The “fail” case in the above diagram is when a guest fails the biometric match. In this case, the light is turned blue by the xBRC. A cast member can then clear that reader through a cast member user interface. When this occurs, the xBRC sends another command to turn the light off.

Another possible path not shown in the sequence diagram is when it is determined that an xBand is not entitled to enter the park. In this case the xBRC would turn the light blue immediately without initiating gathering of biometric finger print data.

Not shown in the above sequence diagram is the xBRC’s interaction with other applications on the network to determine ticket entitlements, gather existing finger print templates, talk with the cast member user interface, etc.

## Assumptions and Constraints

## Definitions

## Reference

| Document Name & Version | Relationship |
| --- | --- |
| xBRC Interface Control Document (ICD) | Documents interface between xBRC and computers on system bus. |

# General Formatting Notes

## Time

Date and Time is represented using the ISO 8601 standard with, except with an optional milliseconds field appended to the end.

Example:

2011-06-20T13:41:00.891

## MAC

The MAC (Media Access Control address or Ethernet Hardware Address (EHA)) is formatted according to IEEE MAC-48, using ‘:’ as a separator character, in transmission order. See <http://en.wikipedia.org/wiki/MAC_address>, Notational Conventions.

Example:

01:23:45:67:89:ab

## Payload Formats

Commands that return a particular format have the format type appended to the command URL (e.g. events.json). Currently, JSON is the only supported payload type, but the naming convention allows for other payload types in the future if required.

## POST vs. PUT

We always use PUT to the xBRC, because the web server software used on the xBRC expects POST to include form data and doesn’t work so well in our case.

The readers are actually agnostic as to whether the verb is PUT or POST, they are treated identically.

## Controller (xBRC) Address

The reader can retrieve its xBRC address through three different means. These methods in order of priority are:

1. Manually by setting the “xbrc url” setting in the configuration file located at /var/mayhem/grover.conf.
2. Through DHCP: The reader requests Vendor-Specific Information (option 43), with Vendor Class Identifier (option 60) set to “SYNAPSE”. If the DHCP server responds with corresponding options, the http address specified in Vendor-Specific Information will be written to /var/lib/dhcp/xbrc-url. The reader application reads this file each time it attempts to contact the xBRC.
3. Through the xBRMS: The reader will query the local DNS server for a SRV record under the service name “\_rest.\_tcp.xbrms.<domain name>” (the domain name is determined at run time) which should give the reader a list of xBRMS servers that can be contacted. The reader will then send a “hello” message (see section 3.1**.**) to identify itself to the xBRMS. If the xBRMS has an xBRC in which to direct the reader, it will send a “PUT /xbrc” command to the reader to configure its xBRC address.

When the reader receives a “PUT /xbrc” command, it will save the URL in persistent storage. If the above three methods are not successful in setting the address then the reader will default to this address until instructed otherwise.

# xTP to xBRC Messages

## PUT ControllerServer/hello

This message is sent to the xBRC shortly after power up, and then periodically after that. This message lets the xBRC know that the reader is alive and well. The response lets the xTP know that the xBRC is alive and well.

Payload

The payload in the hello PUT is the same as the response payload for the GET reader/info.json service (See 5.3 below).

# xTP Events

Events on an xTP are things like RFID reads, finger print scans, and periodic diagnostic events. Events are queued up by the xTP and can be pulled by the xBRC via this command, or the xBRC can be instructed to push events with the “PUT update\_stream” command.

## Events Payload

Whether events are pulled via a GET, or pushed, the payload looks the same and looks like this:

{

“reader name”: <reader name>,

“events”:

[

Events…

]

}

Fields

|  |  |
| --- | --- |
| Name | Description |
| reader name | Name assigned to reader. Mac address provided if no reader name assigned. |
| Events | An array of events sorted by time of arrival. The format of each event depends on the type of event. |

## Event Types

### RFID Event

Each xTP event represents a read of the RFID from an xBand. The format of each event is as follows:

{

“eno”: 987,

“type”: “RFID”,

“uid”: <UID>,

“pid” : <public ID>,

“sid” : <secure ID>,

“iin” : <IIN>,

“time”: <time>

}

Fields

|  |  |
| --- | --- |
| Name | Description |
| Eno | Event number. |
| Type | Event type. In this case “RFID” |
| Time | Time packet was received. Millisecond resolution. |
| Uid | Manufacturer’s Unique ID (MUID) or serial number from the RFID chip. |
| Pid | Public ID (also read by long range readers) |
| Sid | Account number portion of the secure ID. A 16 digit decimal number. Not included if the RFID option “secure\_id” is turned off. See “PUT rfid/options”. |
| Iin | The IIN portion of the secure ID – a six digit decimal number. Not included if the RFID option “secure\_id” is turned off. |

Example:

{

“reader name”: “Charlie”,

“events”:

[

{ “eno”: 100,

“type”: “RFID”,

“time”: “2011-07-08T13:41:03.750”,

“uid”: “12A36E03ABD32103”

“pid”: “54AC3B4901”,

“sid”: “1234567890123456”,

“iin”: “123456”

}

]

}

### Biometric read event

Biometric read events are formatted as follows:

{

“eno”: <event number>,

“type”: <event type>,

“time”: <time>,

“uid”: <most recently read RFID>,

“xbio data”: <xbio data>

}

Fields

|  |  |
| --- | --- |
| Name | Description |
| eno | Event number |
| type | Event type. Can be “bio-enroll” or “bio-match”. |
| time | Time packet was received. Millisecond resolution. |
| uid | Most recently read UID |
| xbio data | xBIO data structure sent as a base 64 encoded blob. This structure includes a new template. If in test mode, the structure also includes fingerprint images. |

### Biometric Scan Error Event

Biometric scan error events are formatted as:

{

“eno”: <event number>,

“type”: <event type>,

“time”: <time>,

“uid”: <most recently read RFID>,

“reason”: <reason for error>

}

Fields

|  |  |
| --- | --- |
| Name | Description |
| eno | Event number |
| type | “xbio-scan-error” |
| time | Time of event |
| uid | Most recently read UID |
| reason | Reason for the scan error. One of the following:  “lift off”  ”movement”  “timeout” |

### Biometric Image Event

Biometric Image events are formatted as:

{

“eno”: <event number>,

“type”: <event type>,

“time”: <time>,

“uid”: <most recently read RFID>,

“xbio-image”: <reason for error>

}

Fields

|  |  |
| --- | --- |
| Name | Description |
| eno | Event number |
| type | “xbio-scan-error” |
| time | Time of event |
| uid | Most recently read UID |
| xbio-image | Image data from the last biometric read, base64 encoded. (Note this data will be large, ~1.5 megabytes). |

### xTP Diagnostics Event

This event occurs every 60 seconds and relays diagnostic information about the reader. The event is not generated if the link to the xBRC is down, that is it is only generated if events are being successfully pushed to a URL set via PUT /update\_stream. The diagnostics event contents can also be acquired with GET /diagnostics.

The events are formatted as follows:

{

“eno”: <event number>,

“type”: “xfp-diagnostics”,

“time” : <date-time>

“status” : <overall status>,

“status msg” : <message if status not ‘Green’>,

“stats”: {

“taps”, <number of track taps>,

“min tap time”, <smallest tap time, ms>,

“max tap time”, <largest tap time, ms>,

“mean tap time”, <average tap time, ms>,

“period”: <stat collection period before refresh, s>,

“time”: <current collection time, s>

}

“RFID status” : <xBIO status>,

“RFID msg” : <XBIO status message>,

“xbio status” : <xbio status>,

“xbio msg” : <xbio status message>,

“amb”: <ambient light reading>,

“temp”: <temperature>,

“max temp”: <max temperature>,

}

Fields

|  |  |
| --- | --- |
| Name | Description |
| eno | Event number |
| type | “xfp-diagnostics” |
| time | Date and time |
| status | Status of dap reader. Either “Green”, “Yellow” or “Red”. |
| status msg | Reason for problem if status is other than “Green”, otherwise empty string. |
| stats | Collects statistical data of timings related to RFID reads. See format example above for details on the sub-fields. |
| RFID status | Status of the Feig RFID reader (“Green”, “Yellow” or “Red”). |
| RFID msg | Reason for problem if RFID status is other than “Green”. |
| xbio status | xBIO status (“Green”, “Yellow” or “Red”). (park entry daps only) |
| xbio msg | Reason for problem if xBIO status is other than “Green”. |
| Amb | Ambient light reading |
| Temp | Internal temperature reading |
| max temp | Maximum temperature reading since program started. |

Example

{

“eno”: 8923,

“type”: “xfp-diagnostics”,

“time” : <date-time>

“status” : “Red”,

“status msg” : “No response from Feig”,

“RFID status” : “Red”,

“RFID msg” : “No response from Feig”,

“xbio status” : “Green”,

“xbio msg” : “”,

“amb”: 3543,

“temp”: 33.45,

“max temp”: 42.3,

}

### xBIO Diagnostic Event

This event occurs at a TBD periodic rate when the park-entry xTP acquires diagnostic information from the xBIO device. The events are formatted as follows:

{

“eno”: <event number>,

“type”: “xbio-diagnostic”,

“data”: <xbio diagnostic data>

}

Fields

|  |  |
| --- | --- |
| Name | Description |
| eno | Event number |
| type | “xbio-diagnostic” |
| data | xBIO diagnostic data formatted as a base 64 encoded blob. |

### xBIO FW Upgrade Event

This event is sent on completion of a firmware upgrade of the xBIO device.

{

“eno”: <event number>,

“type”: “xbio-fw-upgrade”,

“status”: <success or failure>

“version”: <version number reported by the newly upgraded firmware>

}

Fields

|  |  |
| --- | --- |
| Name | Description |
| eno | Event number |
| type | “xbio-fw-upgrade” |
| status | Status of the upgrade. Can be “success” or “failed” |
| version | Version number reported by the firmware after the upgrade. |

# xTP Commands

## GET events

Retrieves events. Each reader will maintain a list of the last TBD events. If this command is issued without specifying an “after” parameter, then the command will return events starting after the last event previously retrieved. When a reader is connected to a single host, the host could retrieve all events by periodically issuing GET events commands with no parameters. The “after” parameter provides a way for multiple hosts to retrieve events.

The response to this service always includes a list of events. The fields included for each event vary depending on the type of event, as documented under the Events section.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| after | Yes | Retrieves events after the given event number. |
| max | Yes | Maximum number of events to return. If the number of available events exceeds ‘max’, then the earliest events are returned. |

Examples:

GET events

GET events?after=54893

Response (xTP)

See the Events Payload sub-section under the Events section.

## PUT events

Post one or more arbitrary events to the reader’s event queue. Used for testing purposes. Body of the message must be JSON encoded and can either be a single event, or an array of events. No validation is performed on the contents of the individual events.

## PUT reader/name

Set the reader name. The reader name is stored by the xTP in flash and remembered after a power cycle. If not set, the reader name defaults to an empty string.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| name | No | Reader name |

Examples

PUT reader/name?name=Charlie

PUT reader/name?name=534

## GET reader/info

Get information about the reader.

NOTE: This service returns the same payload as it transmitted in the “hello” PUT sent out at program start up.

Response

{

“mac”: <mac address>,

“port”: <listening port number>,

“reader name”: <reader name>,

“reader type”: <reader type>

“reader version”: <reader application version>,

“linux version”: <Linux version>,

“HW type”: “xTP2”,

“next eno”: <next event number>,

“num events”: <number of stored events>,

“queued events”: <number of events not yet sent>

“oldest event”: <date/time of oldest event>,

“push url”: <URL of events pushed to>,

“RFID description” : <description provided by Feig firmware>,

“RFID fw version” : <version provided by Feig firmware>,

“xbio fw version” : <xbio firmware version>, // park entry only

“xbio hw version” : <xbio hardware version>, // park entry only

“xbio serial no” : <xbio serial number> // park entry only

}

Fields

|  |  |
| --- | --- |
| Name | Description |
| Mac | Mac address |
| Port | Listening port number |
| reader name | Reader’s name |
| reader type | Reader type. One of the following  xFP  xFP+xBIO (park entry) |
| reader version | Reader application software version number |
| linux version | Linux kernel version number |
| HW type | Hardware type/revision |
| next eno | Event number to be assigned to next event number that occurs. |
| num events | Number of stored events |
| queued events | Number of queued events (events that have not yet been retrieved by the xBRC) |
| oldest event | The time/date of the oldest event |
| push URL | The URL that events are being pushed to |
| RFID description | Description of the RFID reader device as provided by the RFID firmware. |
| RFID fw version | RFID reader firmware version |
| xbio fw version | Biometric reader firmware version (park entry only) |
| xbio hw version | Biometric reader hardware version (park entry only) |
| xbio serial no | Biometric reader serial number (park entry only) |

Example

{

“mac”: “01:23:45:67:89:ab”,

“port”: 8008,

“reader name”: “entry-1”,

“reader type”: “xFP+xBIO”,

“reader version”: “1.03”,

“linux version”: “2.6.30”,

“next eno”: 25123,

“num events”: 10200,

“queued events”: 25

“oldest event”: “2011-06-20T13:41:00.891”,

“push url”: “192.168.0.2:8080/LRRHello”,

“RFID description” : “Feig (C4 55 0d30)”,

“RFID fw version” : “769 – 0”,

“xbio fw version” : 16223,

“xbio hw version” : 0,

“xbio serial no” : 125

}

## GET reader/log

Return the contents of the reader application log.

Response

Opening config file </var/mayhem/config.json>

Web server port is 8080

Added webserver option 'listening\_ports' -> '8080'

Added webserver option 'ssl\_certificate' -> '/usr/lib/ssl/server.pem'

Checking for xbio update

## PUT update\_stream

This command causes the reader to begin posting events to a given URL at a specified frequency. The reader will post to only one URL at a time, so subsequent PUT update\_stream commands will overwrite a previously stored posting URL. The format of the data posted to the specified URL is identical to the response data of the GET events command.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| url | No | URL to post updates to |
| Interval | Yes | Posting interval in milliseconds |
| Max | Yes | Maximum number of events per POST |
| After | Yes | Retrieves events after the specified event number |

Example:

PUT update\_stream?url=168.192.0.2/services/events&interval=2000

## GET update\_stream

Return the current values for URL, interval, and maximum number of events to send, set through the “PUT /update\_stream” command.

***Response***

{

"interval" : 100,

"max" : 100,

"url" : "http://"

}

## DELETE update\_stream

Stops posting of events to a URL.

Example

DELETE update\_stream

## PUT application/reset

Resets the reader application software.

NOTE: This command relies on the reader monitor script that is run as part of the normal reader start up script (i.e. “/etc/init.d/reader start”). Because of this, this command will not work if you are running the reader program from the command line.

## PUT system/reset

Resets the reader (restarts Linux)

## PUT install

Provides a URL to run “opkg” on to install some software.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| url | No | URL for opkg to pull package from |

Example

PUT install?url=http://192.168.0.2./upgrade/xTP.ipk

## PUT upgrade

Upgrade the device using “opkg” and a set of given repositories. Device will return the output of the upgrade and reboot upon completion.

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| repos | No | Repositories information, an array of parameter objects – { “name”: ”<value>”, “url”: ”<value>”, ”weight”: ”<value>”}  Details of the parameter fields for each repository follow. |
| name | No | *Repository Field:* Name of the source repository. |
| weight | No | *Repository Field:* Relative value of files in this repository compared to the other repositories listed. Higher values are selected first, in the case where a file can be found in multiple repositories. |
| path | No | *Repository Field:* Supplies the http address of the repository. |
| downgrade | Yes | Indicates that a downgrade is intended. Default is false. |

Example

{

"repos":

[

{

"name":"overo",

"path":"http://192.168.73.56/reader/repos/overo",

"weight":"51"

},

{

"name":"armv7a",

"path":"http://192.168.73.56/reader/repos/ armv7a ",

"weight":"46"

},

{

"name":"all",

"path":"http://192.168.73.56/reader/repos/all",

"weight":"1"

},

{

"name":"xBR",

"path":"http://192.168.73.56/reader/repos/xBR",

"weight":"90"

}

],

“downgrade”: false

}

## PUT time

Sets the time on the reader to the specified time, see time format section above.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| Time | No | Reader time |

Examples

PUT reader/time?time=2011-06-20T13:41:00.891

## GET time

Get the current date/time. The time is set via the Network Time Protocol (NTP).

Response

**{**

“time”: <time>

}

Example

{

“time”: “2011-06-23T18:30:23”

}

## PUT rfid/tap

Add a simulated tap event to the event queue. Used for testing purposes.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| Uid | no | UID |
| Pid | no | Public ID |
| Sid | yes | Secure ID Account Number |
| Iin | yes | Secure ID IIN |

Example

PUT rfid/tap?uid=54E03ABD32103323&pid=123456&sid=890ABC

## PUT rfid/options

Set options related to RFID reader.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| test\_loop | yes | Turn RFID test loop on or off. Value can be “on” or “off” and defaults to off. |
| public\_id | yes | Turn reading and reporting of public ID on or off. Value can be “on” or “off” and defaults to on. Note that if both public\_id and secure\_id are turned off, then the xTP will read and report the UID for non-provisioned RFID tags. |
| secure\_id | yes | Turn reading and reporting of secure ID on or off. Value can be “on” or “off” and defaults to on. |

Example

PUT rfid/options?test\_loop=off

## GET rfid/options

Get current rfid options settings.

Response

**{**

“test\_loop”: <on or off>,

“public\_id”: <on or off>,

“secure\_id”: <on or off>

}

Example

{

“test\_loop”: “off”,

“public\_id”: “on”,

“secure\_id”: “on”

}

## PUT tap/options

Set options related to taps. Note that the default is that the xTP sends an event on taps, but does not turn on any lights or play any sounds unless instructed to do so via a “PUT media” commands. The options in this message allows you to set up lights and/or a sound to be displayed or played automatically after every tap.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| Color | yes | Color of light to turn on, on tap. See the PUT light command for available colors. |
| sequence | yes | Light sequence to display on tap. See the PUT light command for available sequences. |
| Timeout | yes | Timeout for lights after tap |
| Sound | yes | Sound to play on tap |

Example

PUT tap/options?color=green&timeout=2000&sound=tap.wav

## DELETE tap/options

Delete all of the tap options. xTP will not turn on lights or play sounds on taps, but will continue to send out tap events.

## GET tap/options

Get the current settings of the tap options.

Response

**{**

“color”: <color or sequence name>,

“timeout”: <light timeout>,

“sound”: <sound file played on tap>

}

Example

{

“color”: “green”,

“timeout”: 2000,

“sound” : “tap.wav”

}

## GET diagnostics

Output the latest diagnostics. The format is in JSON and matches the format described previously for the Diagnostics Event, but does not include the “eno” and “time” fields.

## PUT xbrc

Set the xBRC URL. The URL value will persist on restart. This command will have no effect if the xBRC URL is configured either manually or through DHCP.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| url | no | Full URL to the xBRC. |

## DELETE xbrc

Clear the current xBRC URL. This command will have no effect if the xBRC URL is configured either manually or through DHCP.

Example

DELETE /xrbc

# Media interface

The following commands define the interfaces for installing and playing media content on the reader.

## GET media/inventory

Return information about the available media on the reader. The data is separated into two sections: one for defaults and one for an installed media package.

Response

**{**

"defaults" : {

      "ledscripts" : [ *<list of LED script names>* ],

      "sounds" : [ *<list of sound names>* ],

      "xbio" : [ *<list of xbio script names>* ]

   },

   "package" : {

      "ledscripts" : [*<list of LED script names>* ],

      "media hash" : <unique hash/identifier for the installed package>,

      "sounds" : [*<list of sound names>* ],

      "xbio" : [ *<list of xbio script names>* ]

   }

}

Example

GET /media/inventory

## PUT media/package

Install a new media package on the device, replacing any existing package. The body of this message should be in a gzip/tar format with the following structure:

ledscripts/*<scripts as CSV files>*

sounds/*<sounds as WAV files>*

xbio/*<xBIO XML script file>*

HTTP Header Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| X-Media-Hash | yes | A unique string used to identify the media package in its entirety. |

## DELETE media/package

Remove the currently installed media package, if any, from the reader.

Example

DELETE /media/package

## PUT media/idle

Set the idle sequence for the reader. The specified sequence must be a valid sequence on the device.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| Name | No | Name of the sequence to play on idle |

Example

PUT /media/idle?name=thinking

## DELETE media/idle

Remove the currently set idle sequence, if any.

Example

DELETE /media/idle

## PUT media/sequence

Initiate scripted lighting and sound effect. An optional timeout turns the light on for a specific period of time.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| Name | No | Name of the sequence to play |
| Timeout | Yes | Timeout in milliseconds |

Individual sequences are contained in .csv files that define LED scripts as described in the “Media Sequence Script Specification” section at the end of this document. When asked to play a sequence, the xTP first looks for a script file (.csv file) with the given name loaded via the PUT /media/package command, and if no match is found, then looks through the set of default script files that are installed with the application.

The script name “off” is special. By default “off” will cancel any script that is playing and turn off the LEDs. This behavior can be changed however, but loading an “off.csv” script file onto the xTP (with PUT /media/package), in which case the xTP will play the script “off.csv”.

When there is no “off.csv” script, the timeout parameter still works with “off”. In this case, the timeout is the amount of time that the LEDs are turned off before the idle script resumes.

Example

PUT /media/sequence?name=entry\_success

PUT /media/sequence?name=entry\_exception&timeout=10000

Both examples, find and play a script file “entry\_success.csv”. The first example plays the script until completion, cancelled, or replaced by another script. The second example stops the script after 10 seconds.

## GET media/sequence

Return the currently playing sequence or color effect. If a sequence is playing, then this command will return the name of the sequence or “Custom” if the sequence was initiated using the PUT /media/ledscript command. If a color effect is playing on no effect at all, then this command will return the RGB values on the currently shown color (with all zeros for off).

Response

**{**

"name” : “thinking”

}

## PUT media/color

Initiate a simple color effect. An optional timeout turns the light on for a specific period of time.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| Name | No | Name of the color effect to show or comma separated RGB values. |
| Timeout | Yes | Timeout in milliseconds. Idle script is played after the timeout, or LEDs are turned off if there is no idle script. |
|  |  |  |

Colors

A color name given alone will light all of the LEDs on the reader. Adding an “inner” prefix to the color name will target the inner LEDs, while prefixing “outer” will only target the outer LEDs. Available colors are: red, green, blue, yellow, and white.

If the color name is “off”, then any scripts are cancelled and the LEDs are turned off.

Example

PUT /media/color?name=red

PUT /media/color?name=0,255,255

PUT /media/color?name=green&timeout=10000

PUT /media/color?name=innerblue&timeout=10000

PUT /media/color?name=outergreen&timeout=10000

PUT /media/color?name=off

## PUT media/off

Cancels any LED script that is playing and turns off the LEDs. (This does not abort any sound file that is playing)

## PUT media/sound

Play a sound.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| Name | No | Name of the sound to play |

Default Sounds

The following sounds are available on the reader by default: biocue, bioprocess, exception, retry, success, and tap.

Example

PUT /media/sound?name=success

## PUT media/ledscript

Initiate a scripted sequence that is provided by the caller in the body of the request. Resources, such as sounds, must be present on the device in order to be played.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| Timeout | Yes | Timeout in milliseconds |

## PUT media/brightness

Set the global LED brightness.

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| value | No | Integer value between 0 (Off) and 255 (Max) |

# Biometric Commands

These commands are available only on the park entry version of the xTP.

## PUT biometric/enroll

Instructs the biometric reader to begin capturing a finger print template for a new guests.

## PUT biometric/match

Instructs the biometric reader to begin capturing a finger print template for matching. If templates are provided, then the reader will also use the Lumidigm biometric reader to do the match comparisons.

Payload

{

“templates”: [<template>, …]

}

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| templates | Yes | Each template is a base 64 encoded blob containing a single fingerprint template used by the xBIO firmware. The templates are optional. If provided, the matching will be performed by the xBIO. |

## PUT biometric/cancel

Cancel a biometric read operation and turn off the lights.

## PUT biometric/firmware

Downloads a firmware image to be installed in the xBIO device. An xBIO FW Upgrade event is output when the upgrade is completed.

Payload

The payload is the binary firmware image. Content-Type is application/octet-stream.

## PUT biometric/options

Set options related to biometric reader (xbio).

Parameters

|  |  |  |
| --- | --- | --- |
| Name | Optional | Description |
| image\_capture | yes | Turns image capture on or off. Value can be “on” or “off” and defaults to “off. |

Example

PUT biometric/options?image\_capture=on

## GET biometric/options

Get current biometric option settings.

Get current rfid options settings.

Response

**{**

“image\_capture”: <on or off>

}

Example

{

“image\_capture”: “off”

}

## GET biometric/image

Get the most recent image stack from the xBio.

If the xTP is in test mode, then after each biometric read, what is called the “image stack” is retrieved from the xBIO and placed into memory. The image stack is a large (~1.5Mbyte) block of data that contains raw images and other debug information useful to Lumidigm. This command will retrieve the stored image stack from the most recent biometric read.

The return payload is binary data with Content-Type “application/octet-stream”. If no data is available, the Content-Length will be 0.

## PUT biometric/image/send

Causes the xTP to post a xbio-image event with the latest biometric image data.

# Media Sequence Script Specification

The sequence scripts used to play the lighting and sound effects are plain text CSV files where each line is a LED frame to display, a command, or a comment. Scripts either run until completion or loop forever depending on how they are defined. Script line terminations can either be in DOS (CRLF) or Unix (LF) format.

## Comments lines

All comment lines start with the hash ‘#’ symbol. All other text after this symbol are discarded and ignored.

## Commands

The following table lists and describes the available commands:

|  |  |
| --- | --- |
| Command | Description |
| $color <name> <r> <g> <b> | Define a new named color with the specified RGB values. This color will be able to be referenced in frame line after this command. |
| $sound <name> | Play the named sound effect. |
| $xbio <name> | Play the named biometric reader script. |
| $xbiobrightness <1-8> | Set the brightness level of the biometric reader. |
| $repeat | Start the script from the beginning |
| $goto <lines> | Move execution of the script up/down the specified number of lines. |
| $timemode\_duration | Specify that frame delays are the time to wait before moving on to the next frame. i.e. A delay given as 2.0 will wait 2 seconds. This is the default mode. |
| $timemode\_absolute | Specify that frame delays are absolute. i.e. A frame delay given as 2.0 followed 2.1 will play 100ms apart. |
| $mintime <time> | Minimum number of seconds to play this script before it can be interrupted by a queued script. (Default: None) |
| $notinterruptible | Specify that the script cannot be interrupted by a tap event. |

## Frames

Frame lines instruct the reader to display LEDs in a specific pattern. Their format follows the given pattern:

*<delay>,<led1>, <led2>,…,<led48>*

The delay value is measured in seconds and can be a floating point value to represent fractions of seconds. Each led parameter is a named color defined by a previously given “$color” command.

## Example Script

# This and the following line will be ignored. Script will play for a minimum of 1 second

# and will light the first three LEDs one second apart in sequence repeatedly.

$color red 255 0 0

$color green 0 255 0

$color blue 0 0 255

$timemode\_duration

$mintime 1000

1.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

1.0,red,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

1.0,red,green,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

1.0,red,green,blue,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

$goto -3