xBRMS Architecture

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Description** |
| 1.0 | 5/1/12 | Arkady Glabek | Release Version |
| 1.6 | 4/22/12 | Iwona Glabek | Updates for 1.6.x |

**Document Approvers & Sign-Off**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Approver** | **Role** | **Document Accept/Reject** |
| 5/12/12 | Glenn Curtis | Development Manager | Accept |
| 4/15/12 | Glenn Curtis | Development Manager |  |

**Table of Contents**

[1 Introduction 4](#_Toc356404551)

[1.1 Purpose 4](#_Toc356404552)

[1.2 Referenced Documents 4](#_Toc356404553)

[1.3 Definitions 4](#_Toc356404554)

[2 System Overview 5](#_Toc356404555)

[3 Deployment 6](#_Toc356404556)

[3.1 DISC Infrastructure 7](#_Toc356404557)

[3.2 LDU Infrastructure 8](#_Toc356404558)

[3.3 Configuration Properties 8](#_Toc356404559)

[3.4 Park Setup 10](#_Toc356404560)

[3.5 BigIP Configuration 11](#_Toc356404561)

[4 High Availability Support 12](#_Toc356404562)

[5 Messaging Protocols 12](#_Toc356404563)

[5.1 Monitoring 13](#_Toc356404564)

[5.1 REST Query Interface 13](#_Toc356404565)

[6 Database Structure 14](#_Toc356404566)

[6.1 Table Structure 14](#_Toc356404567)

[6.2 Stored Procedures 15](#_Toc356404568)

[7 Access Control 15](#_Toc356404569)

[8 Functionality 15](#_Toc356404570)

[8.1 Web Based User Interface 16](#_Toc356404571)

[8.1.1 Parks Setup 16](#_Toc356404572)

[8.2 Global Service 16](#_Toc356404573)

[8.2.1 Assign Readers 16](#_Toc356404574)

[8.2.2 Replace Reader 16](#_Toc356404575)

[8.3 Environment Specific Service 16](#_Toc356404576)

[8.3.1 System and Reader Health 16](#_Toc356404577)

[8.3.2 Stored xBRC Configurations 16](#_Toc356404578)

[8.3.3 xBRC Configuration Edit 17](#_Toc356404579)

[8.3.4 REST Query Interfaces 17](#_Toc356404580)

[8.4 Auto Discovery Mechanism 17](#_Toc356404581)

[9 System Health Design 17](#_Toc356404582)

[10 Stored xBRC Configurations 19](#_Toc356404583)

[11 xBRC Configuration Edit 21](#_Toc356404584)

[12 REST Query Interface 22](#_Toc356404585)

[13 UI Web Interface 22](#_Toc356404586)

# Introduction

xBRMS stands for the xBand Reader Management System. The xBRMS application provides monitoring of as well as inventory and configuration management for the xConnect system. xBRMS makes its data and functionality available both via a Web based UI (User Interface) and REST (Representational State Transfer) Service interfaces.

## Purpose

This document describes [system overview](#_System_Overview), [deployment](#_Deployment), [high availability support](#_High_Availability_Support_1), [messaging protocols](#_Messaging_Protocols), [database structure](#_Database_Structure), [access control](#_Access_Control_1), [functionality](#_Functionality), and Web service interfaces of the xConnect xBRMS system.

## Referenced Documents

|  |  |
| --- | --- |
| **Document Name** | **Purpose** |
| xBRMS Interface Control Document (900-0059) | Describes all available REST endpoints. |
| xBRMS User Guide (900-0061) | User guide to xBRMS User Interface. |
| xBRC Interface Control Document (900-0059) | xBRC ICD |
| xConnect BigIP HA Configuration (900-0174) | Describes BigIP configuration of xConnect system. Includes appropriate iApp templates. |

## Definitions

|  |  |
| --- | --- |
| **Term** | **Definition** |
| xBRMS | xBand Reader Management System |
| xBRC | xBand Reader Controller |

# System Overview

xBRMS application is a web service written in Java programming language that runs on TCServer web server. All components of the xBRMS application are [highly available](#_High_Availability_Support) and access to these components is [protected](#_Access_Control). Each REST Service requires a dedicated instance of [SQL Server database](#_Database_Structure) and DNS support. Communication between the Web UI and the REST Service components is accomplished over HTTP REST. Communication between the xBRMS REST Services and external applications is accomplished over HTTP Rest and Java Message Service (JMS).

The Web UI is composed of dynamic web pages generated using the Struts2 application framework and JavaServer Pages (JSP) technology. The xBRMS Web UI is HTML5 enabled and supports the following Web browsers: Mozilla Firefox, Chrome, and IE7 and up.

xBRMS REST Service communicates over HTTP REST with xBRC, IDMS, JMS Listener, xGreeter, GxP applications, and the xTP and xBR reader devices. JMS messaging is used by the xBRMS [auto discovery mechanism](#_Auto_Discovery_Mechanism). Access to both the xBRMS Web UI and its REST endpoints is protected. Every request is routed via the xConnect xAC ([Access Control](#_Access_Control_1)) component. The xAC component authenticates requests against the KeyStone Service, when available. When an instance of KeySton is not available a local XML file is used for authentication.

The following diagram shows how xBRMS application interacts with other systems, both xConnect and external.



# Deployment

xBRMS application is composed of one Web UI Service, one Global REST Service, and multiple Environment Specific REST Services. Each of these services is [highly available](#_High_Availability_Support_1) and access to them is [protected](#_Access_Control). Both the WEB UI and REST Services run on TCServer. Each of the REST Services requires its own [SQL Server database](#_Database_Structure) and DNS support. The Web UI and the Global REST Services are [deployed to the DISC](#_DISC_Infrastructure) while the environment specific REST Services are [deployed to their respective environment’s LDUs](#_LDU_Infrastructure).

xBRMS REST Services use multiple configuration properties. All the access sensitive properties that must be password protected are specified in the NGE /etc/nge/config/environment.properties file. Properties specific to [park setup](#_Parks_Setup) are specified in the /usr/share/xbrms/parks.xml file generated by the xBRMS REST Services. All the other configuration properties are housed in the [dbo.config database](#_Database_Structure) table and can be updated via the xBRMS Web UI.

The following two subsections showcase the DISC and LDU infrastructure using the Release Lab environment as an example.

## DISC Infrastructure



The following elements are required for a typical DISC deployment:

* [SQL Server database](#_Database_Structure). The database could be created in the SQL Server cluster the IDMS uses.
* Two load balanced xBRMS REST Service VMs (VIP: nge-prod-disc-xbrms) acting as a Global xBRMS.
* Two load balanced xBRMS Web UI VMs (VIP: nge-prod-xbrms-ui). Only one xBRMS Web UI Service is needed to service all xBRMS REST Services.
* The \_rest.\_tcp.xbrms DNS SRV record pointing to the xBRMS REST Service’s VIP address (nge-prod-disc-xbrms). Readers use this record to find the global xBRMS to which to report to for [xBRC assignment](#_Assign_Readers).
* DNS Aliases (CNAME records) corresponding to the Global xBRMS REST Service and the environment specific xBRMS REST Services existing in the LDUs: nge-prod-disk-xbrms-ui , nge-prod-mk-xbrms-ui, nge-prod-hs-xbrms-ui-, nge-prod-ep-xbrms-ui, nge-prod-ak-xbrms-ui, etc. These make it possible for users to open multiple copies of the xBRMS UI in a single browser window with each tab opened to the xBRMS UI working with a different environment specific xBRMS REST Service.

## LDU Infrastructure



The following elements are required for a typical LDU deployment:

* [SQL Server database](#_Database_Structure).
* Two load balanced xBRMS REST Service VMs (VIP: nge-prod-XX-xbrms). These are environment specific. Typically, there are four xBRMS REST Services deployed (one per LDU) representing the four parks: nge-prod-mk-xbrms, nge-prod-ep-xbrms, nge-prod-ak-xbrms, and nge-prod-hs-xbrms, but there could be more. For example, Thphoon Lagoon could be a separate environment.
* Environment specific DNS SRV records: \_rest.\_tcp.mk-xbrms, \_rest.\_tcp.hs-xbrms, \_rest.\_tcp.ep-xbrms, \_rest.\_tcp.ak-xbrms, pointing to the appropriate VIP address of the environment specific xBRMS REST Service. These records are currently used by the xGreeter application to discover a per-park list of /facilities.

## Configuration Properties

The below table describes all environment.properties configuration properties required by the xBRMS REST Service. These properties must be available on startup. Changes to any of the below properties require an application restart.

|  |  |  |
| --- | --- | --- |
| **Property** | **Value** | **Description** |
| **JMS Related Properties** | | |
| nge.xconnect.parkid | Alphanumeric ID uniquely identifying an environment. | Used by the xBRMS DISCOVERY mechanism to filter JMS messages on. Messages from other environments with different park IDs or those that do not include that property will be ignored. |
| nge.eventserver.brokerDomain |  | Properties used by the xBRMS JMS agent to connect to the JMS Bus. JMS messages are used by the xBRMS DISCOVERY mechanism. |
| nge.eventserver.xbrc.connectionfactory.jndi.name | XBRCConnectionFactory |
| nge.eventserver.mgmtBrokerUrl | <management broker hostname>:<port> |
| nge.eventserver.xbrc.uid | User ID |
| nge.eventserver.xbrc.pwd | Password |
| **Database Related Properties** | | |
| nge.xconnect.xbrms.dbserver.url | jdbc:jtds:sqlserver://<DB\_HOSTNAME>:1433/<DB\_NAME> | Used to discover which database to connect to on startup. |
| nge.xconnect.xbrms.dbserver.uid | Database user. | Used by the xBRMS REST Service to connect to its database. Must have write, read, and execute privileges. |
| nge.xconnect.xbrms.dbserver.pwd | Database user password. | Used by the xBRMS REST Service to connect to its database. |
| nge.xconnect.xbrms.c3p0.maxPoolSize | An integer (ex: 20). | Initialization properties of the c3p0 JDBC database connection pool the xBRMS REST Service uses to query its database. |
| nge.xconnect.xbrms.c3p0.maxStatementsPerConnection | An integer (ex: 50). |
| nge.xconnect.xbrms.c3p0.maxStatements | An integer (ex: 400) |
| **Access Control Related Properties** | | |
| nge.xconnect.ac.logonServer | hostname | Address of the xConnect webapp that serves authentication requests. Default: the same host your browser points to. |
| nge.xconnect.ac.logonPort | port number | Default: 8080 |
| nge.xconnect.ac.logonAppNamea |  | Name of the webapp that serves authentication requests for the xConnect components. Default: IDMS |
| nge.xconnect.ac.ksDirs | Comma separated list of Keystone directories. | This list is only used when the KeyStone service is not available at the time xBRMS starts up. |
| nge.xconnect.ac.ksApplicationId | 1af99062-f279-4579-94b3-0b1a144566b5 | ID uniquely identifying xConnect applications. |
| nge.xconnect.ac.ksPrimaryAuthURL | URL | URL of the primary |
| nge.xconnect.ac.ksSecondaryAuthURL | URL |  |

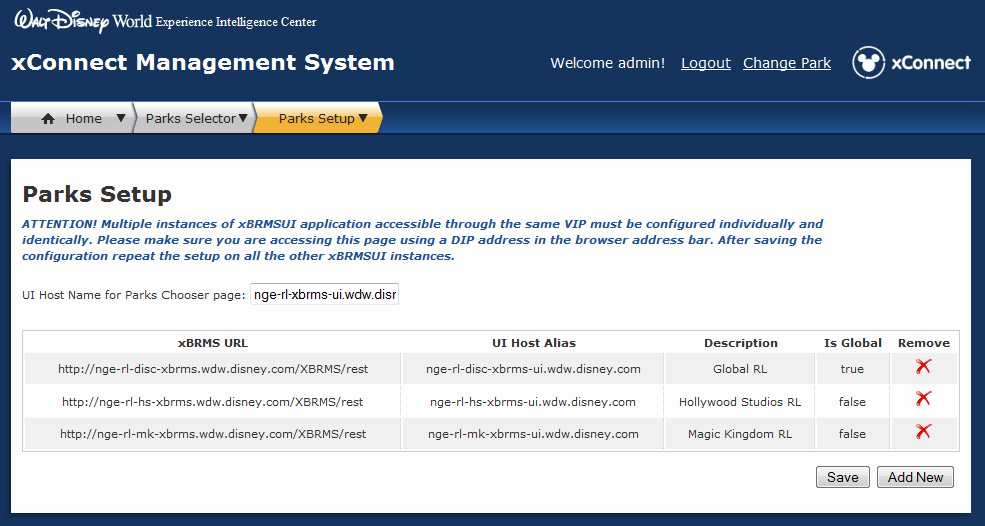
The table below describes all the required configuration properties housed in the dbo.config database table. The xBRMS application will start even when these properties are missing, but it might not work as expected. Changes to any of these properties do not require a restart.

|  |  |  |
| --- | --- | --- |
| **Class** | **Property** | **Description** |
| MSConfig |  |  |
|  |  |
| XBRMSConfig |  |  |
|  |  |

## Park Setup

The Park Selector page in xBRMS Web UI allows users to configure the xBRMS Web UI Service to work with multiple xBRMS REST Service installations. **IMPORTANT**: multiple instances of xBRMS Web UI application accessible via the same VIP must be configured individually and identically.

The topology reflected in the below screenshot is described in detail in the [DISC Infrastructure](#_DISC_Infrastructure) section of this document.



## BigIP Configuration

xBRMS HA (High Availability) architecture supports one-to-many servers per each HA group.

Servers belonging to the same xBRMS Web UI HA group should be load balanced in a round robin fashion.

The load balancing of the servers belonging to the same xBRMS REST Service HA group is based on the response from the HTTP GET /rest/status request.

For details on how to configure xBRMS load balancing refer to the *xConnect BigIP HA Configuration* document.

# High Availability Support



# Messaging Protocols

The following diagram illustrates the various types of interfaces implemented by the xBRMS Web Service. Refer to *xBRMS Interface Control Document* for details on the available REST endpoints.

Figure 1 - xBRMS Interfaces

## Monitoring

The xBRMS Monitoring Interface allows xBRMS to monitor the health of various system components. Two communication channels are used, HTTP and JMS (Java Message Queue). The JMS channel is used by the xBRMS to subscribe to the discovery message sent out by the xBRC. This allows for automatic discovery of all xBRC installations on the network. The HTTP channel is used to periodically query the status of any monitored applications.

The xBRMS is capable of monitoring diverse applications. Each new monitored application is added to the xBRMS by subclassing a HealthItem Java class. The HealthItem implementation communicates via HTTP or other protocol with the monitored application to obtain the status of the application. The XBRC, IDMS, and JMS Listener all use the HTTP /status call for this purpose.

Only the master xBRMS pings the monitored applications for status and persists that status into the SQL Server database. The slave xBRMS servers then read that information from the database.

Information on readers’ health comes directly from the xBRC.

## REST Query Interface

The REST Query Interface is used by the xBRMS UI to retrieve and manipulate data collected by the xBRMS servers. It also allows other applications to query the xBRMS system for information about applications it monitors. Most of the REST endpoints the xBRMS servers make available are protected by [Access Control](#_Access_Control). However, some of the endpoints, for example the /facilities endpoint, are left unprotected for backwards compatibility.

# Database Structure

A separate instance of the SQL Server database is needed in each LDU and one in the DISC. Each instance of XBRMS database in the DLU is used by the xBRMS REST Service HA group, and XI and JMS Listener xConnect applications. xBRMS application shares the FacilityID and FacilityTypeID tables shown in the blow diagram with the xConnect XI application.

The database user used by the xBRMS application requires read, write, and execute permissions.

## Table Structure



|  |  |
| --- | --- |
| **Table** | **Description** |
| config | Stores non-sensitive application configuration parameters |
| XbrmsHaGroup | Keeps track of all xBRMS REST servers in an HA group. All members of the HA group write their heartbeat to this table. Also, as soon as one of the members assumes master’s responsibilities it indicates that fact in this table. |
| HealthItem | Collection of xConnect application servers currently being monitored by this xBRMS REST Service. |
| HealthItemField | Each health item, regardless of type, has a set of mandatory properties such as IP, hostname, or port. xBRMS also collects health item type specific properties which are stored in this table. En example of such property is the ‘Model’ which only makes sense for health items of type xBRC. |
| PerformanceMetric | Collection of performance metric readings collected over time. |
| PerformanceMetricDesc | PerformanceMetric table metadata. |
| Facility | Attraction/xBRC. |
| FacilityType | Type/model of an attraction/xBRC: attraction, parkentry, or space. |
| schema\_version | Current and historical data on database schema version. |

## Stored Procedures

xBRMS application uses two stored procedures:

* dbo.usp\_PerformanceMetric\_create
* dbo.usp\_PerformanceMetricDesc\_create

# Access Control

# Functionality

The xBRMS Web Service provides the following functionality.

## Web Based User Interface

The UI Web Interface is a collection of web pages written in jsp allowing the user to manage stored configurations, configure the xBRC , monitor the health status of the system, assign readers to xBRCs, replace faulty readers, and more. Functionality made available in the UI when the UI is connected to the Global server is different from when it is connected to one of the Environment Specific servers.

For details on all the xBRMS functionality available via the Web UI refer to the *xBRMS User Guide* document.

### Parks Setup

The **Parks Setup** page allows the user to configure the xBRMS Web UI Services to service a list of this xBRMS Web UI Service should serve.

## Global Service

### Assign Readers

Readers that haven’t yet been configured to report to a specific xBRC phone home to the Global xBRMS and wait on the **Found Readers** page until a user assigns them to a specific xBRC. The readers find the xBRMS to use for this purposes via a SRV record.

### Replace Reader

The **Replace Reader** page allows users to replace faulty readers with fully functional readers. First, the user selects the park and the xBRC the faulty reader resides in and the reader that needs replacing. xBRMS then displays a list of appropriate replacement candidates and, upon user confirming the replacement action, delegates the responsibility of replacing the faulty reader with the new reader to its parent xBRC.

## Environment Specific Service

### System and Reader Health

The **System Health** monitors the state of the various software and hardware components that make up the xConnect system. This currently includes the xBRC, IDMS, JMS Listener and the xTP and xBR readers. The status of the monitored components is presented on a web page allowing quick assessment of the health of the entire system. Systems that should be monitored by a given xBRMS are either added to that xBRMS manually or are discovered over JMS. Reader health is reported to the xBRMS indirectly by the xBRCs these readers are assigned to.

### Stored xBRC Configurations

The xBRMS provides a repository of stored **xBRC Configurations** as well as the ability to deploy a stored configuration to a selected xBRC. A stored xBRC configuration contains a complete set of settings to configure an xBRC, including a list of readers. It is also possible to store and deploy partial configurations.

### xBRC Configuration Edit

The **xBRC Configuration Edit** mechanism allows for modification of configuration parameters for a single xBRC or a selected group of xBRCs. The main intent of this functionality is to simplify making similar changes to multiple xBRCs.

### REST Query Interfaces

The REST Query Interface allows other systems to query the xBRMS for the information about monitored xBRC applications. This service is helpful to any application that needs to dynamically build an inventory of xBRC installations. For each xBRC a contact URL is provided to make further REST calls to the xBRC. The REST Query Interface is described in detail in a separate document titled. Refer to *xBRMS Interface Control Document* for details on the available REST endpoints.

## Auto Discovery Mechanism

Provided the xBRMS discovery mechanism is enabled, the environment specific xBRMS REST Services consume JMS discovery messages from xBRCs with the [nge.xconnect.parkid](#_Configuration_Properties) property matching its own.

Multiple message properties are set on the discovery messages but xBRMS only cares about the following two properties:

xbrc\_message\_type=DISCOVERY

xconnect\_parkid=<given environment’s unique identifier>

The payload of the xBRC discovery message looks like this:

{"name":"123456787","port":8080,"haStatus":"unknown","venue":"123456787","model":"com.disney.xband.xbrc.spacemodel.CEP","configurationChangedTime":"2013-05-02T21:16:53","ip":"10.110.1.78","hostname":"SIT-XBRC-GFF-B","remoteSVUID":1,"discoveryInterval":60}

For more information on messages sent by the xBRC refer to the xBRC ICD.

To enable the discovery mechanism, configure the [jmsxbrcdiscoverytopic](#_Configuration_Properties) xBRMS property to the JMS topic used by the xBRCs. To disable this service, set the [jmsxbrcdiscoverytopic](#_Configuration_Properties) to #.

# System Health Design

The System Health monitoring is only performed by the environment specific xBRMS REST Services. Iis implementation consists of the following components.



Figure 2: System Health Components

1. A Java class for each type of monitored application that inherits from com.disney.xband.xbrms.server.model.IHealthSystem interface. The main functionality of this class is to provide a refreshStatus() method which pings monitored applications for their health status and persists that information to the database.
2. A registration mechanism where each type of monitored application is registered with the xBRMS Web Service. Currently, this registration mechanism includes 1) automatic registration using JMS DISCOVERY messages sent by the xBRC and 2) manual registration initiated by the user by supplying DIP hostname and Port of the monitored application using a web page.
3. A background thread that periodically checks the status of each monitored application by calling the IHealthSystem.refreshStatus() method.
4. A UI web page that shows the status of all monitored applications on a single web page that is automatically refreshed. The UI web page offers drill-down functionality to investigate in more detail the status of any monitored application.

The following figure shows the System Health UI web page.

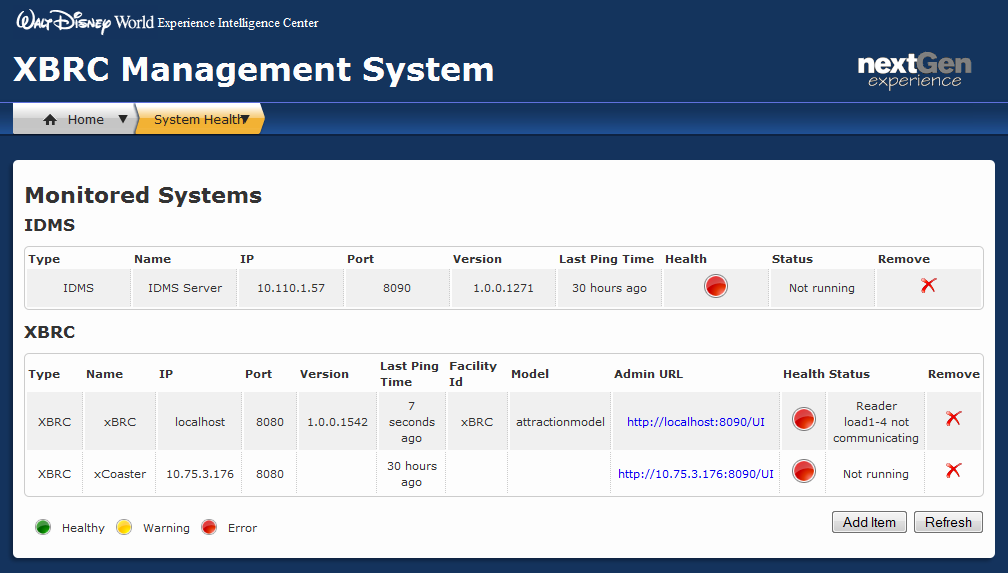


Figure 3: System Health Page

# Stored xBRC Configurations

The following diagram illustrates the components of the Stored xBRC Configurations functionality.



Figure 4: Stored xBRC Configurations

The following figure shows the Stored xBRC Configurations web page.

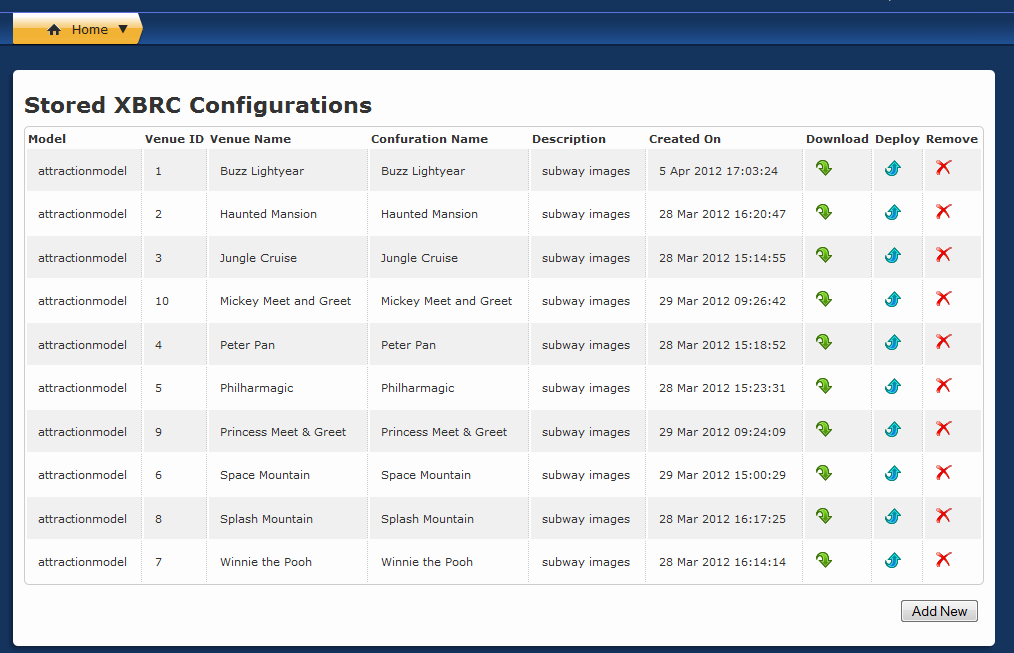


Figure 5: Stored xBRC Configurations Page

Please refer to the *xBRC Interface Control Document*for the description of the xBRC Configuration REST interface.

# xBRC Configuration Edit

The following figure shows the xBRC Configuration Edit page.



Figure 6: xBRC Configuration Edit Page

The xBRC Configuration Edit functionality is implemented using REST /properties call to the xBRC. The individual properties for the xBRCs are stored in the Mayhem.Config MySQL table on each of the xBRC machines. The xBRMS Web Service retrieves the properties from all xBRCs that are selected by the user using the checkboxes on the right side of the xBRC Configuration Edit Page. Once the properties are retrieved they are grouped by model (Space, Attraction, Parkentry). For each xBRC with a matching model all properites are compared to see if they are the same. If the property is the same for multiple xBRCs then the property value is displayed in the corresponding edit box. If the property is not the same for multiple xBRCs then a “Do Not Change” text is displayed instead. This mechanism allows to modify some properties making them the same for all selected xBRCs while preserving the unique values of other properties not being modified by the user.

# REST Query Interface

Refer to the *xBRMS Interface Control Document* for the description of the REST Query Interface.

# UI Web Interface

Refer to the xBRMS User Guide document for details on *xBRMS Web User Interface*.