IDMS Test Plan

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Description** |
| 1.0 | 5/1/12 | Rob Silvernagel | Release Version |

**Document Approvers & Sign-Off**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Approver** | **Role** | **Document Accept/Reject** |
| 5/1/12 | Mark Mecham | Synapse Test Manager | Accept |

**Table of Contents**

[1 Introduction 5](#_Toc337045273)

[1.1 Purpose 5](#_Toc337045274)

[2 Background 6](#_Toc337045275)

[3 Scope 7](#_Toc337045276)

[3.1 Dependencies 7](#_Toc337045277)

[3.2 Ownership 7](#_Toc337045278)

[3.3 Referenced Documents 8](#_Toc337045279)

[3.4 IDMS Architecture 8](#_Toc337045280)

[3.5 IDMS ID Relationship 9](#_Toc337045281)

[3.6 Sequence Diagram 10](#_Toc337045282)

[4 Testing Approach 11](#_Toc337045283)

[4.1 Use Cases - GxP Test 11](#_Toc337045284)

[4.2 Use Cases - Beyond the GxP Test 11](#_Toc337045285)

[4.3 Test Strategies 11](#_Toc337045286)

[4.3.1 Silt Data 11](#_Toc337045287)

[4.3.2 Test Data 12](#_Toc337045288)

[4.3.3 Test Execution 12](#_Toc337045289)

[4.3.4 Identifiers 12](#_Toc337045290)

[4.4 API Testing 13](#_Toc337045291)

[4.4.1 Test Case Structure 13](#_Toc337045292)

[4.5 Stress Testing 13](#_Toc337045293)

[4.6 IDMS Database Testing 14](#_Toc337045294)

[4.6.1 Data Validation 14](#_Toc337045295)

[4.6.2 Schema Validation 14](#_Toc337045296)

[5 Restful Endpoints 15](#_Toc337045297)

[5.1 Critical IDMS Endpoints 15](#_Toc337045298)

[5.1.1 Guests 15](#_Toc337045299)

[5.1.2 xBands 15](#_Toc337045300)

[5.2 Legacy Calls 16](#_Toc337045301)

[5.2.1 Guest 16](#_Toc337045302)

[5.2.2 Meta 16](#_Toc337045303)

[6 Performance Targets 18](#_Toc337045304)

[7 Exit Criteria 19](#_Toc337045305)

# Introduction

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

## Purpose

The intent of the IDMS test plan is to provide a mid-level approach for testing IDMS and related services. The intended audience is anyone who may possibly be testing the component. From an agile “Kanban” approach this could be a QA team member, a developer, or anyone else with the appropriate technical expertise to interpret and execute the tests. The tools and test methodologies referenced in this document serve as suggestions for testing the component. Other methods may be utilized in place of or in conjunction with the suggested test strategy.

# Background

The ID Management System (IDMS) stores and maintains information about park guests, making it possible to link external systems with the xConnect components. IDMS utilizes a SQL Server database instance for data storage and a RESTful web service interface for data access. Although IDMS is not technically a System of Record (SOR) it does provide a systematic way of storing and retrieving guest information as well as looking up associations between guests, parties, xBands, and xConnect related IDs.

Many xConnect components rely on IDMS data for real-time operations. The accuracy and speed of the IDMS system is paramount to the overall performance of the xConnect system. From a quality assurance perspective, the priority for testing IDMS is:

1. Availability of RESTful endpoints
2. Accuracy of the data
3. Response time

# Scope

| **Feature** | **GxP Test** |
| --- | --- |
| Guests | X |
| xBands | X |
| Meta | X |
| Guest | X |
| Visit | X |
| Itinerary |  |
| Celebrations | X |
| xBand Import from Manifest | X |
| DME Identifiers |  |

## Dependencies

The ability for IDMS to function in the GxP Test is dependent upon the following services.

| External | For | Description |
| --- | --- | --- |
| UIE Cast Application | GxP Test | Guest creation and guest-xBand association |
| GxP SOR | GxP Test | gxp-link-id generation |

## Ownership

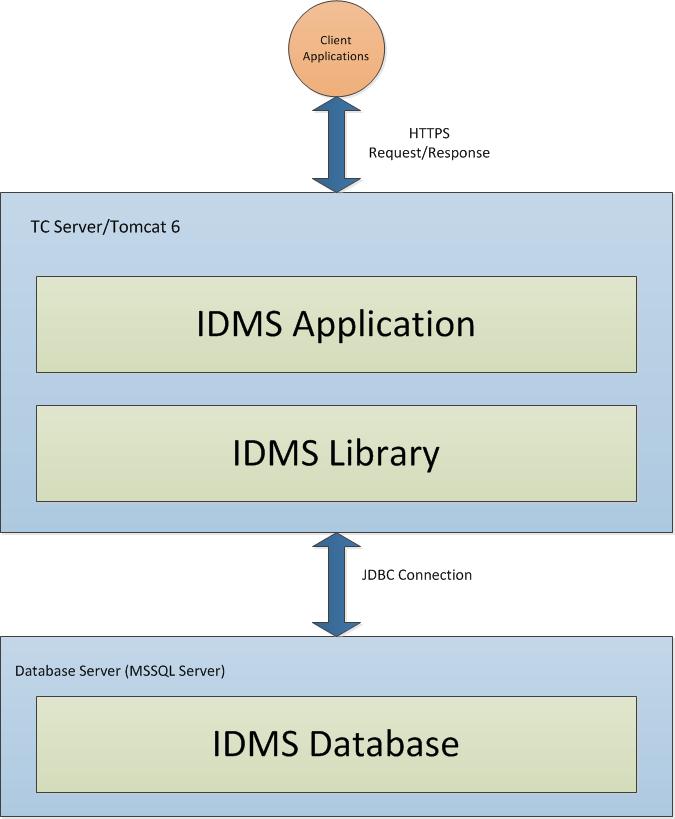
The contacts for each component reliant on IDMS are listed below.

| Compo-nent | Developer |  | QA Tester |  |
| --- | --- | --- | --- | --- |
| IDMS | Robert Lantry | robert.lantry@synapse.com | Rob Silvernagel | robs@synapse.com |
| GxP | Brad Sokola | bradley.r.sokola@accenture.com | GxP | GxP |
| xBRC | Arek Glabek | arkady.glabek@synapse.com | Tim Sharpe | tims@synapse.com |
| xBRMS | Iwona Glabek | iwona.glabek@synapse.com | Stephen Madson | stephenm@synapse.com |
| Cast application | Cecile Tron | ctron@uievolution.com | UIEvolution | UIEvolution |
| Kiosk | Frank Ivan | fivan@fourwindsinteractive.com | Four Winds | Four Winds |

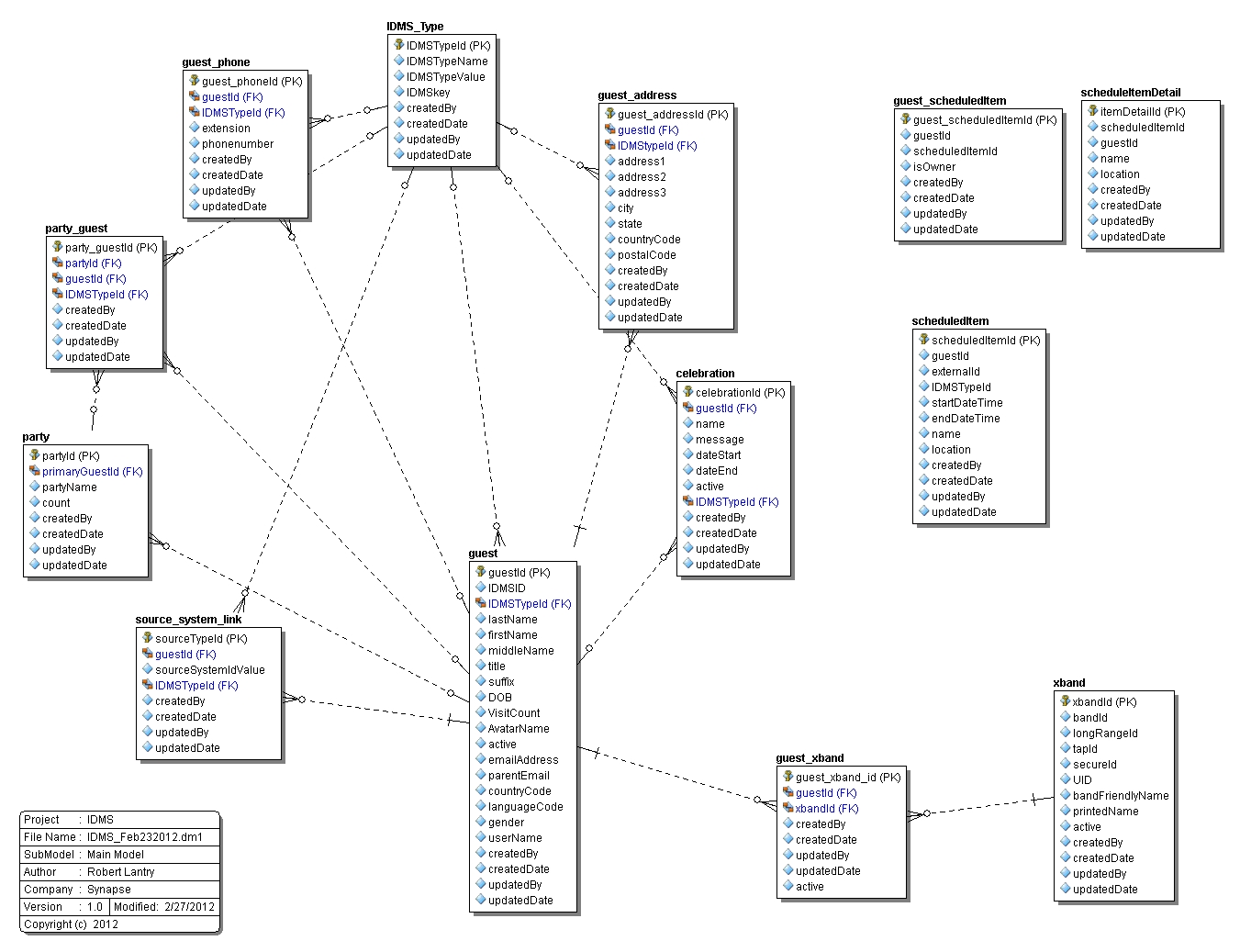
## Referenced Documents

| Document Name & Version | Issuance Date | Relationship |
| --- | --- | --- |
| IDMS REST API | 11/29/11 | Documents HTTP RESTful protocol |
| IDMS ICD | 4/27/2012 | Further defines the REST API |

## IDMS Architecture



## IDMS ID Relationship



## Sequence Diagram

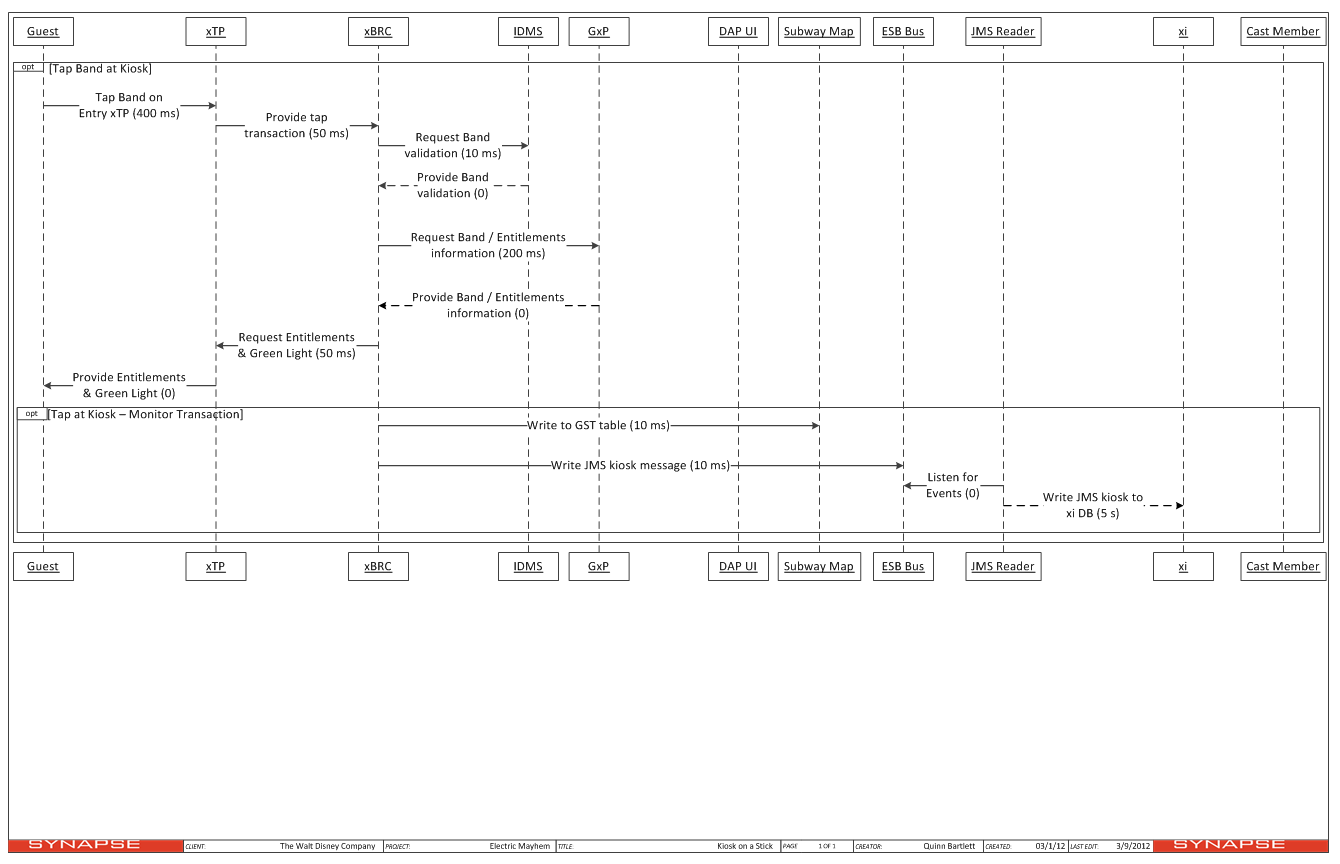


Figure 2 Sequence diagram for a kiosk tap, which is the same flow structure as a redemption tap.

# Testing Approach

One important aspect of testing IDMS is to ensure that the RESTful web services are available and functioning as intended. Another is validating that the database contains the appropriate data after any given set of operations are performed. This section outlines the approach to be used when testing these functions of IDMS.

## Use Cases - GxP Test

* A family of four creates an account at the airport using UIE’s Cast Application. IDMS is called to add guests to the system. A party is created and one of the guests is assigned as the party leader.
* A single user creates an account at the airport using UIE’s Cast Application. IDMS is called to add the guest to the system. A party is created and the guest is assigned as the party leader.
* A guest has changed addresses and needs to update their address in the system.
* A guest decides to visit the park for their child’s birthday. The guest adds a celebration to their experience. The Cast Application calls IDMS to create a celebration and associate it with the guest.
* A guest logs in to the GxP website using their email address. IDMS resolves the email address to the guestID.
* A guest taps their card at a park kiosk to determine if their Space Mountain entitlement is valid at this time. The secure ID is read from the card and used to look up the guest in IDMS.
* A guest loses their card and a replacement is assigned. A new guest is created in IDMS. When the guest is looked up again in IDMS the most recent guest with that email address is returned.

## Use Cases - Beyond the GxP Test

* A guest taps their band at park entry. An IDMS lookup identifies the guest by the secure ID before calling OMNI to verify park access.
* A guest’s xBand is detected by an xBR (long range reader) The xBRC uses the xBand’s UID to call IDMS and identify the guest.
* A guest sits down at a Great Food Fast table and the xBRC calls IDMS to identify the guest and look up their food order.

## Test Strategies

### Silt Data

The IDMS database under test will be restored to a known state and will contain ‘silt’ data required for the test being executed. Silt data can be loaded by running a SQL script that will import the silt data into the IDMS database. This data will contain guests, xBands, parties, and celebrations as well as the associations that link those together. At the time this document was written, the silt data was the same data used for the xConnect load test (BVT\_IDMS).

### Test Data

While silt data is necessary to ensure that there is data in the system that can be tested against, the test data defines the actual test cases that will be exercising IDMS to ensure proper functionality.

The test data will consist of five parts:

1. The hostname and port of the IDMS instance under test
2. The request, or the url of the RESTful endpoint being tested
3. The HTTP method (GET, POST, PUT, DELETE)
4. The request body, or payload of the request (if applicable)
5. The expected response

### Test Execution

IDMS tests are executed by making RESTful calls to the exposed IDMS endpoints and validating the responses. Many of the IDMS endpoints use the ‘GET’ method. These endpoints can be tested from an internet browser such as Microsoft Internet Explorer or Mozilla FireFox. The format for the URL is:

http://{IDMS\_host}:{port}/IDMS/{RESTful endpoint}.

The response body will contain a JSON object which will be displayed in the browser.

While most calls can be easily tested from the browser, there are tools available to test the other methods such as ‘POST’, ‘PUT’, and ‘DELETE’. The recommended tool is ‘Fiddler’ which is available at [www.fiddler2.com](http://www.fiddler2.com). Follow the same URL structure as previously mentioned. Select the appropriate method from the dropdown list. If the endpoint requires a request body, there is a text input box at the bottom of the Fiddler UI. After making a request, the request will appear in the ‘Web Session’ frame on the left side of the Fiddler UI. Select the request to see the response. The response headers will include an ‘HTTP 200’ to indicate success. Many times there will be a response body that also needs to be validated against the expected results.

### Identifiers

There are various ways of identifying a guest or xBand in the IDMS. The guestID serves as the primary key for a guest and uniquely identifies guests in the system. External systems have their own means of uniquely identifying guests. Tests that require a guest ID should also test the various external IDs in addition to the IDMS guest ID.

| External | ID Type | Example call |
| --- | --- | --- |
| GxP | gxp-link-id | http://localhost:8080/IDMS/guest/id;gxp-link-id=2468/xbands |
| XBMS | XBMS | http://localhost:8080/IDMS/guest/id;XBMS=314159/xbands |
| IDMS | xid | http://localhost:8080/IDMS/guest/id;xid=1357/xbands |
| IDMS | xband | http://localhost:8080/IDMS/guest/id;xband=98765/xbands |
| UIE | [email] | http://localhost:8080/IDMS/guest/searchEmail/test@domain.com |

## API Testing

The IDMS API consists of a set of RESTful endpoints. Manual testing of the API can be conducted using tools such as Fiddler or WireShark. Automated testing will be implemented via node.js or soapUI.

A list of the valid endpoints is contained in the ‘IDMS ICD’ (also see ‘IDMS REST API Calls’). A terse reference is available in this document (see [RESTFUL ENDPOINTS](#_RESTFUL_ENDPOINTS)).

### Test Case Structure

* The test data will consist of a request to IDMS as well as a method (GET, POST, PUT, DELETE) based on the specifications for the endpoint. The test data can be exercised either manually (via Fiddler or WireShark) or in an automated manner (using node.js scripts).
* The request will consist of the test data and the method
* The test result is the response that IDMS returns after receiving the request
* The test data should denote instances where the test results requires santization (i.e. removal of headers, decoding, etc)
* The sanitized test result will be compared to the expected test result and a pass/fail will be generated.

## Stress Testing

Selenium Grid can be used to generate a high number of concurrent IDMS connections. The tests will be scalable and various performance metrics can be gathered with each load increase. The scalability will come from varying the number of Selenium remotes that are generating IDMS requests. Each remote will send IDMS a new request after the previous request’s response is received. Performance metrics will be used to determine how much latency is introduced as the load increases from 2->5->10->20->30->40 remotes/threads. Each test will run for a minimum of 5 minutes.

## IDMS Database Testing

### Data Validation

The data validation tests will tie in with the system load tests such as ‘day in the life of the park’ tests. These tests verify that IDMS is providing the needed endpoints and data to fulfill requests

### Schema Validation

Schema validation involves validating that the schema is correct and that table columns are set to use the appropriate data types. Columns containing string data should be using a data type of ‘nvarchar’, which supports Unicode characters. If ‘varchar’ is used, then extended characters will be stored incorrectly in the database and will result in data loss.

# Restful Endpoints

IDMS employs an API or a set of RESTful endpoints that allow clients to access/interact with IDMS data. The base URL for IDMS is: http://{IDMS\_HOST}:{PORT}/IDMS. A more complete reference can be found in the IDMS ICD and Design document.

## Critical IDMS Endpoints

### Guests

| Key | Description | HTTP Action | Supported for Gxp Test? |
| --- | --- | --- | --- |
| /guests | Returns all of the guests in the system | GET | No |
| /guests/id/{guest id} | Returns an xViewGuest object by guestId | GET | Yes |
| /guests/{guest id} | Returns an xViewGuest object by guestId | GET | Yes |

### xBands

| Key | Description | HTTP Action | Supported for April? |
| --- | --- | --- | --- |
| /xbands | Returns all of the xViewXBand objects in the system. | GET | No |
| /xbands/id/{Secure ID} | Returns an XViewXband object by xBandId (IDMS internal ID). | GET | Yes |
| /xbands/{Secure ID} | Returns an XViewXband object by xBandId (IDMS internal ID). | GET | Yes |

## Legacy Calls

### Guest

| Key | Description | HTTP Action | Supported for April? |
| --- | --- | --- | --- |
| /guest | Returns all of the guests in the system | GET | No |
| /guest/{guest id}/xbands | Returns a list of xBand objects assigned to a user | GET | Yes |
| /guest/{guest id}/identifiers | Returns a GuestIdentifierCollection of GuestIdentifier objects | GET | Yes |
| /guest/{guest id}/identifiers | Creates a source system identifier for a guest by {guest id}. This is provided so that external systems can add a source system ID to a guest | POST | Yes |
| /guest/{guest id}/identifiers | Updates a source system identifier for a guest by {guest id}. This is provided so that external systems can post changes to IDs for a particular guest. | PUT | Yes |

### Meta

| Key | Description | HTTP Action | Supported for April? |
| --- | --- | --- | --- |
| /meta | Accepts an **IDMSTypeListItem** object to create a new IDMSType in the system | POST | No |
| /meta/name/{name} | Returns an **IDMSTypeListItem** object based on the name value of the IDMSType. Ie, “Home Address” | GET | No |
| /meta/key | Returns a list of strings for all of the IDMS Type Keys in the system. | GET | Yes |
| /meta/{id} | Returns an **IDMSTypeListItem** object based on the IDMSTypeId (IDMS internal ID) | GET | Yes |
| /meta/key/{key} | Returns a list of **IDMSTypeListItem** objects that are of the type of {key}. | GET | ? |

| Area | Key | Description | HTTP Action | Supported for April? |
| --- | --- | --- | --- | --- |
| Guest | /guest | Returns all of the guests in the system | GET | No |
| Guest | /guest/{guest id}/xbands | Returns a list of xBand objects assigned to a user | GET | Yes |
| Guest | /guest/{guest id}/identifiers | Returns a GuestIdentifierCollection of GuestIdentifier objects | GET | Yes |
| Guest | /guest/{guest id}/identifiers | Creates a source system identifier for a guest by {guest id}. This is provided so that external systems can add a source system ID to a guest | POST | Yes |
| Guest | /guest/{guest id}/identifiers | Updates a source system identifier for a guest by {guest id}. This is provided so that external systems can post changes to IDs for a particular guest. | PUT | Yes |
| Meta | /meta | Accepts an **IDMSTypeListItem** object to create a new IDMSType in the system | POST | No |
| Meta | /meta/name/{name} | Returns an **IDMSTypeListItem** object based on the name value of the IDMSType. Ie, “Home Address” | GET | No |
| Meta | /meta/key | Returns a list of strings for all of the IDMS Type Keys in the system. | GET | Yes |
| Meta | /meta/{id} | Returns an **IDMSTypeListItem** object based on the IDMSTypeId (IDMS internal ID) | GET | Yes |
| Meta | /meta/key/{key} | Returns a list of **IDMSTypeListItem** objects that are of the type of {key}. | GET |  |

# Performance Targets

IDMS response times will vary based on a number of factors, including network traffic, external components, and the endpoint being requested. In order to ensure that IDMS meets these performance targets a full xConnect system should be exercised under a reasonable load for at least 15 minutes. This includes at least one xBRC connected to an xTP making at least 10 redemption calls to GxP per minute. At the end of this test the IDMS performance metrics can be viewed by opening a browser and browsing to the URL: http://{IDMS host}:8080/IDMS/metrics. Ensure that the average response times fall below these thresholds:

| **Endpoint** | **Maximum Acceptable Average Response Time (in ms)** | **Critical for Redemption?** |
| --- | --- | --- |
| GetCelebrationByGuestXId | 10 | Yes |
| AddCelebrationToGuest | 100 | No |
| SaveGuest | 100 | No |
| PostGuestIdentifier | 50 | No |
| GetXBandsByIdentifier | 5 | Yes |
| GetGuestByEmail | 5 | No |
| GetGuestProfileById | 5 | No |
| GetGuestIdentifiers | 5 | Yes |
| GetXbandBySecureId | 5 | Yes |

# Exit Criteria

1. BVT tests are all passing with the exception of known and acknowledged failures that are documented in the release notes
2. SIT tests are all passing with the exception of known and acknowledged failures that are documented in the release notes
3. Performance targets are met