xQA/xVR Architecture

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Description** |
| 1.0 | 5/1/12 | James Francis, Stephen Madson | Release Version |
| 1.7 | 8/14/13 | Ian Wehmeyer | Updates for 1.7 |

Document Approvers & Sign-Off

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Approver** | **Role** | **Document Accept/Reject** |
| 5/1/12 | Ric Merrifield | Release Manager | Accept |
| 5/11/12 | John Stiehl | Release Manager | Accept |

**Table of Contents**

1 xQA Introduction 4

1.1 History 4

1.2 Purpose 4

1.3 Related Documents 4

1.4 xQA/xVR Contacts 4

1.5 Definitions 5

2 xQA User Interface 7

3 xQA Web Service Interfaces 10

3.1 xQA Web Service Summary 10

3.2 Endpoints 11

4 xQA Data store 14

4.1 Data Store Architecture 16

5 xVR 17

# xQA Introduction

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

## History

In November of 2011 Disney and xConnect West began discussing a real time tool that would support the upcoming FastPass+ Tests. The initial goal of this tool was to track redemption percent, blue lane counts, currently in queue counts, and a Guest list. The secondary goal was to use our real time capabilities to test the FastPass+ system to ensure that readers were functioning as desired. Our vision has grown and now includes park entry, wait times, pre-arrival, recruitment, ping map health monitor, blue lane reason codes, and xVR. We have supported Tests from December 2011 to the current Guest Test 2 in August 2013. These include: FPT1, FPT2, FPT3, IPT, GT1, GT2.

## Purpose

This document describes the architecture and the design of the xQA (Quantitative Analysis) Dashboard user interface, web services, data store, and xVR. Explicitly, the dashboards consist of the xQA.war file deployed in a TCServer instance, and database tables and stored procedures hosted on SQLServer. xVR is a custom application built using Unity, a 3D game development software. The data is driven by database tables and stored procedures hosted on SQLServer.

## Related Documents

|  |  |
| --- | --- |
| **Document** | **Purpose** |
| xQA.xVR.Tour | Description of xQA and xVR with screen shots and details of data elements (PowerPoint) |
| 1.7 xBRMS Architecture (900-0057) | Description of the system, structure, and functionality of the xConnect xBRMS (Document) |
| 1.7 xBRC ICD (900-0058) | Describes how the xBRC communicates with other components of the Disney Next Generation Experience (Document) |

## xQA/xVR Contacts

|  |  |  |
| --- | --- | --- |
| Name | Role | Email |
| Michael Jungen | Director, Game Changers | michael.g.Jungen@disney.com |
| Bo Rankin | Industrial Engineer | john.rankin@disney.com |
| Nathan Ivey (Past) | Industrial Engineer | [nathan.j.ivey@disney.com](mailto:nathan.j.ivey@disney.com) |
| Jenn Fishpool (Past) | Industrial Engineer | jennifer.p.fishpool@disney.com |
| Ian Wehmeyer | PM – Data Analyst | [ianw@synapse.com](mailto:ianw@synapse.com) |
| Amar Terzic | Software Engineer – SQL | [amart@synapse.com](mailto:amart@synapse.com) |
| Robert Lantry | Software Engineer – Java & Unity Development | [robertl@synapse.com](mailto:robertl@synapse.com) |
| Stephen Madson | Software Engineer – Java Script/CSS | [stephenm@synapse.com](mailto:stephenm@synapse.com) |
| Craig Rosenberg | Software Engineer – Unity Development | craigr@synapse.com |
| John Stiehl (Past) | Program Manager – Swedish Chef | johns@synapse.com |
| James Frances (Past) | Software Engineer - Java | jamesf@synapse.com |

## Definitions

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Entitlements | selected or booked experiences |
| Offersets | a group of entitlements that were booked as a group. all standard entitlements are part of an offerset |
| Redemption | the redemption of an entitlement |
| Blue lanes | extraordinary attempts to redeem entitlements |
| Overrides | a blue lane occurrence that was overridden by a cast member such that a guest was granted access to the experience |
| Total Traffic | all Redeemed Entitlements and Blue Lane Overrides |
| Redeemable in Park | all Entitlements that are currently in the park |
| Visits | Black Bar: All Guests who have entitlements for that day; Green Bar: Guests who have redeemed entitlements |
| Pre-Arrival Metric | How far in advance of their valid entitlement date do Guests book. For instance, if a Guest logs into their account on August 10th and books and entitlement for August 17th we would show them in the -7 bucket |
| Engaged | Guest Distribution throughout the Parks |

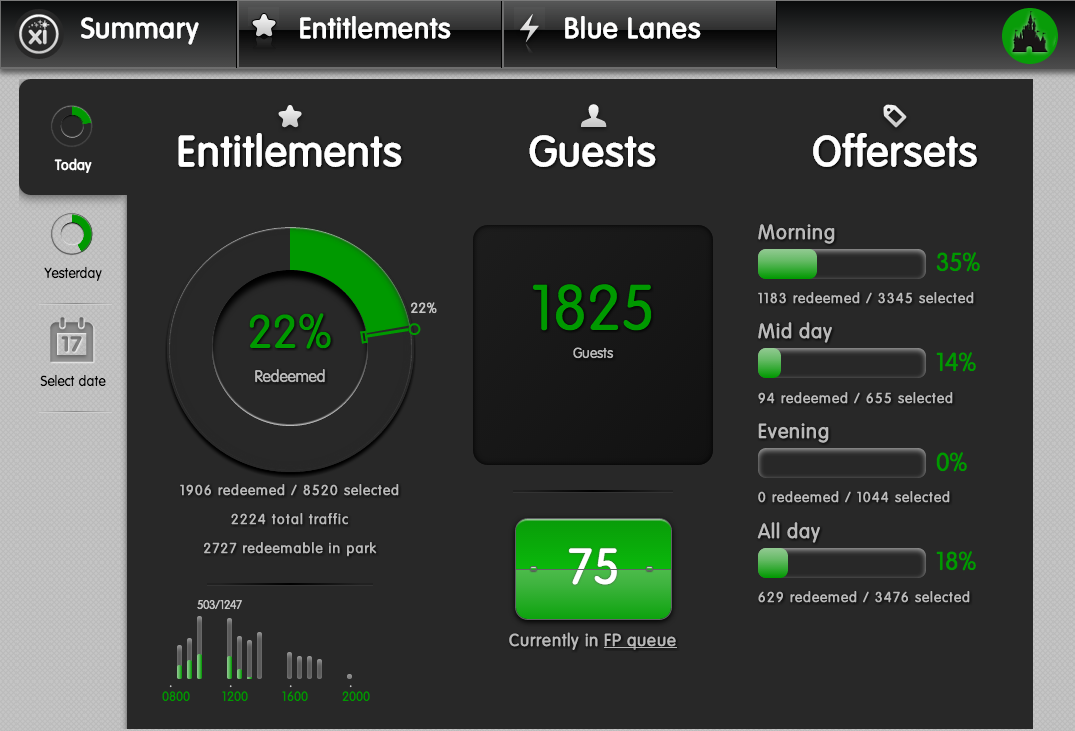
# xQA User Interface

The xQA Realtime Dashboards (xQA) are a set of html5 web applications deployed by a servelet application on TC Server.  They are designed to be viewed on an ipad or via the chrome browser on a Mac or pc. The goal of xQA is to provide relevant real-time information in a very consumable format that is both intuitive and aesthetically pleasing. Once loaded the xQA applications communicate with web service end points via Ajax to maintain a flowing user experience. xQA is very visual, allowing the users easily compare data, recognize trends, and respond to anomalies.

Currently there are 4 xQA Applications:

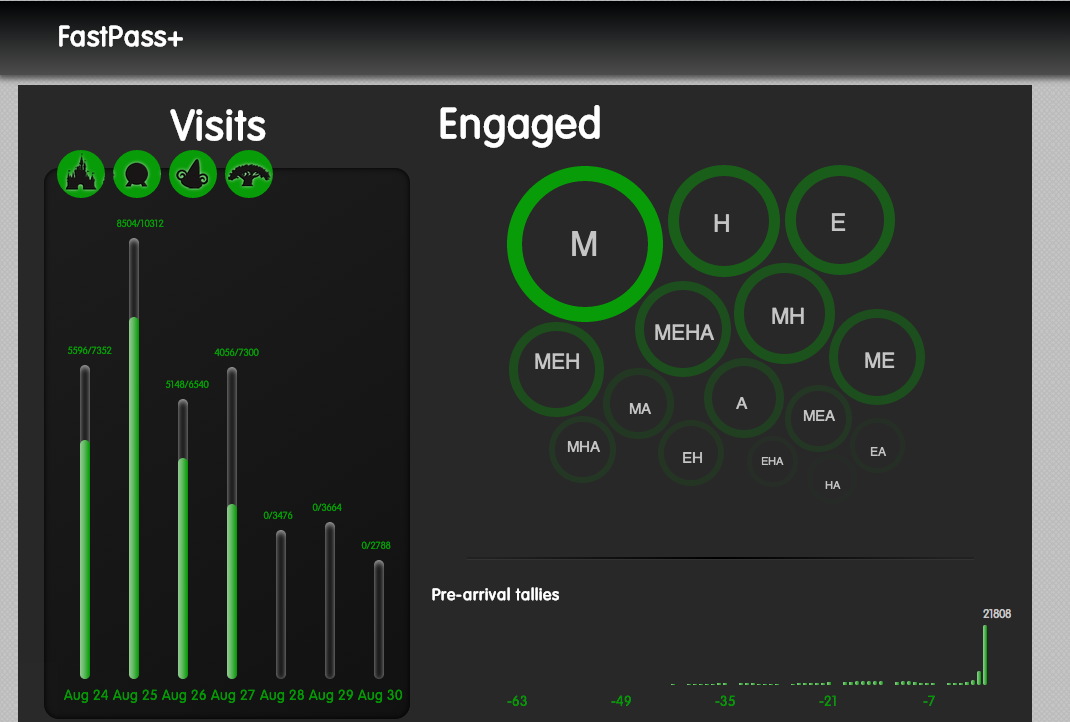
FP+ Dashboard: http://nl-flfa-00454.wdw.disney.com:8080/xi/

1. The FP+ Dashboard is a real time FastPass+ centric application that provides current FastPass+ data for Queues, Wait times, Entitlements, Offersets, and Blue Lanes. Data for MK, HS, Epcot, and AK are included.



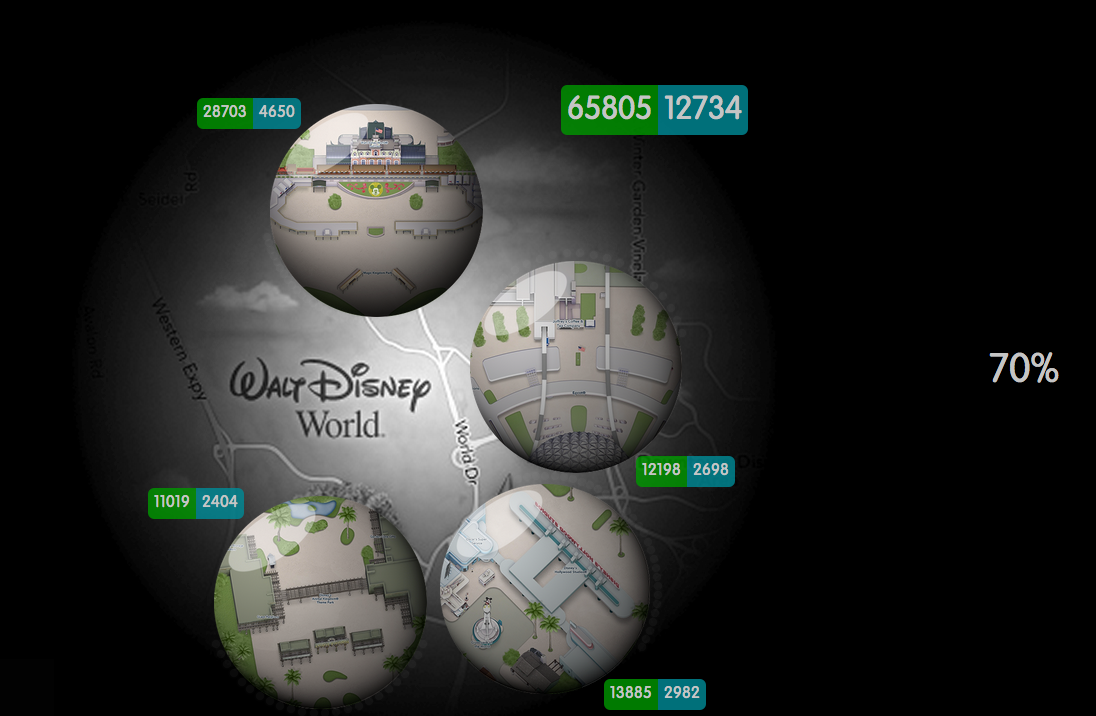
Recruitment Screen: http://nl-flfa-00454.wdw.disney.com:8080/xi/daily.html

1. The Recruitment Dashboard displays the total number of Guests that are due to arrive for a current day, their distribution throughout the Parks, and how early they booked their experiences.



Park Entry: http://nl-flfa-00454.wdw.disney.com:8080/xi/entry.html

1. This application displays park entry events and blue lane events as they happen. It includes daily running totals. It also includes control variance to allow alerting when entry to blue lane ratios drop below the settable expected threshold.



Attraction Tie-Outs: http://nl-flfa-00454.wdw.disney.com:8080/xi/ping.html

1. This application compares ridership volume between different locations in the same attractions and allows alerting when a specific location's ridership volume to total ridership ratio drops below a settable expected threshold.



# xQA Web Service Interfaces

## xQA Web Service Summary

The xQA Realtime Dashboards are supported by a set of web services deployed as a servelet application on TC Server. The chain of events required to display dashboards is as follows:

Data are generated by the xBRCs and posted to a JMS Buss. A JMS Listener picks up the messages, decodes them and then writes them to the appropriate table(S) in the xQA repository. After capturing the messages, a series of ETL processes are run on the data to sort, clean and move the data to various views for latter access and data display.

The xQA dashboards call web-service endpoints, which in turn call stored procedures on the database that are written to view the post-ETL clean data – the data are returned and then moved down the pipeline, from database, to web-service (where it is converted to json data) and ultimately to web-pages where the data are displayed and described.

## Endpoints

The following are the xQA endpoints, with their corresponding database stored procedures.

**Blue Lane Resource**

|  |  |
| --- | --- |
| /bluelane/{parkId}/today | dbo.usp\_GetBlueLaneForAttractionETL |
| /blueland/{parkId}/yesterday | dbo.usp\_GetBlueLaneReasonCodesETL |
| /bluelane/{park}/todate | dbo.usp\_GetBlueLaneReasonCodesETL |
| /bluelane/{park}/{date} | dbo.usp\_GetBlueLaneReasonCodesETL |
| /bluelane/reasoncodes/{attractionId} | dbo.usp\_GetBlueLaneReasonCodesForAttractionETL |

**Daily Report Resource**

|  |  |
| --- | --- |
| /daily/today | dbo.usp\_GetDailyReport |
| /daily/yesterday | dbo.usp\_GetDailyReport |
| /daily/{date} | dbo.usp\_GetDailyReport |

**Entitlements Resource**

|  |  |
| --- | --- |
| /entitlements/{park}/todate | dbo.usp\_GetEntitlementSummaryETL |
| /entitlements/{park}/yesterday | dbo.usp\_GetEntitlementSummaryETL |
| /entitlements/{park}/today | dbo.usp\_GetEntitlementSummaryETL |
| /entitlements/{park}{date} | dbo.usp\_GetEntitlementSummaryETL |
| /entitlements/facility/{attraction}/today | dbo.usp\_GetEntitlementSummaryHourlyETL |
| /entitlements/facility/{facilityId}/yesterday | dbo.usp\_GetEntitlementSummaryHourlyETL |
| /entitlements/facility/{facilityId}/todate | dbo.usp\_GetEntitlementSummaryHourlyETL |
| /entitlements/facility/{attraction}/{date} | dbo.usp\_GetEntitlementSummaryHourlyETL |

**Executive Summary Resource**

|  |  |
| --- | --- |
| /execsummary/{park}/todate | dbo.usp\_GetExecSummaryETL |
| /execsummary/cal/{park} | dbo.usp\_GetRedeemedForCalETL |
| /execsummary/cal/{park}/{date} | dbo.usp\_GetRedeemedForCalETL |
| /execsummary/now | dbo.usp\_GetExecSummaryETL |
| /execsummary/{park}/now | dbo.usp\_GetSubwayQueueCountForAttractionETL |
| /execsummary/{park}/today | dbo.usp\_GetExecSummaryETL |
| /execsummary/{park}/yesterday | dbo.usp\_GetExecSummaryETL |
| /execsummary/{park}/{date} | dbo.usp\_GetExecSummaryETL |
| /execsummary/redemption/{park}/today | dbo.usp\_GetHourlyRedemptionsETL |
| /execsummary/redemption/{park}/yesterday | dbo.usp\_GetHourlyRedemptionsETL |
| /execsummary/redemption/{park}/{date} | dbo.usp\_GetHourlyRedemptionsETL |
| /execsummary/waittimes/{parkId} | dbo.usp\_GetWaitTimes\_ETL |

**Guests**

|  |  |
| --- | --- |
| /guests/facility/{attraction}/date | dbo.usp\_GetGuestsForAttraction |
| /guests/facility/{attraction}/today | dbo.usp\_GetGuestsForAttraction |
| /guests/facility/{attraction}/yesterday | dbo.usp\_GetGuestsForAttraction |
| /guest/{guestid}/today | dbo.usp\_GetGuest |
| /guest/{guestid}/yesterday | dbo.usp\_GetGuest |
| /guests/{guestid}/{date} | dbo.usp\_GetGuest |
| /guests/search/{date}/{rowcount} | dbo.usp\_GetGuestsForSearch |
| /guests/search/{date} | dbo.usp\_GetGuestsForSearch |
| /guests/search/now/{rowcount} | dbo.usp\_GetGuestsForSearch |
| /guests/search/now | dbo.usp\_GetGuestsForSearch |

**JMS**

|  |  |
| --- | --- |
| /jms/venue/{venuename}/messages | No database SP |
| /jms/venue/{venuename}/gueststate | No database SP |
| /jms/venue/{venuename}/messagehistogram | No database SP |
| /jms/venue/{venuename}/messagerange | No database SP |
| /jms/venue/{venuename}/messagecount | No database SP |

**LRR (Long Range Read)**

|  |  |
| --- | --- |
| /lrr/tieout/{parkid} | dbo.usp\_LocationTieOut\_ETL |
| /lrr/queuecounts/{parkid} | usp\_GetSubwayQueueCountForAttraction\_LRR\_ETL |

**PEEvents**

|  |  |
| --- | --- |
| /PEEvents/facility/{attraction} | dbo.usp\_GetPEEventTotalsForPark\_ETL |
| /PEEvents/facilityios/{attraction} | dbo.usp\_GetPEEventTotalsForPark\_ETL |
| /PEEvents/PE/totals | dbo.usp\_GetPEEventTotalsForPark\_ETL |
| /PEEvents/facilities | dbo.usp\_GetPEEventTotalsForPark\_ETL |

**QueueResource**

|  |  |
| --- | --- |
| /queueresource/{park}/now | dbo.usp\_GetSubwayQueueCountForAttractionETL |
| /queueresource/subway/{attractionid} | dbo.usp\_GetSubwayQueueCountForAttractionETL |
| /queueresource/subway/guests/{attractionid} | dbo.usp\_GetSubwayGuestsForReaderETL |

**Recruitment**

|  |  |
| --- | --- |
| /recruitment | dbo.usp\_RecruitBubblesETL |

# xQA Data store

Concept: Accelerated ETL

The idea behind accelerated ETL is to process and publish new data from the ESB in real time. It is based on a same concept as a data warehousing with a difference that data pulls happen in intervals of 3-5 seconds.

ETL itself consists from the following stages that make up all necessary steps in a process of preparing data for xQA screens:

* New data capture (delta processing)
* Pre-processing, cleansing and loading new data
* Reporting and caching
* Publishing data for xQA screens consumptions

There are three main ETLs defined by its purpose:

Entitlement ETL

* This is guest/entitlement centric ETL. It primarily collects GXP data published on ESB and provides real time metrics about selected and redeemed entitlements including blue lanes on the exec summary screen. It also supports recruitment screen.

Long Range Read ETL

* This ETL collects in real time all reader activity such as taps and long range reads. Queue counts and wait times are some of the metrics calculated by this ETL.

Park Entry ETL

* This ETL collects park entry activity including guest taps and blue lanes and provides real time metrics on park entry dashboard.

Each ETL is a single monolithic SQL stored procedure and it is invoked by Windows service running on a SQL server.

xQA database is divided into following logical groups:

* JMS landing tables
* ETL master tables
* Cache tables
* Reporting tables
* Archive tables

JMS landing tables are used to extract content from JMS messages pulled from the ESB. These tables contain raw data and this is the only place where data integrity is enforced.

ETL master tables, cache tables and reporting tables are used by ETLs. These tables contain normalized, cleansed, processed, rolled up and aggregated data.

Archive tables are populated by nightly maintenance job and contain all expired data from landing, master and reporting tables. Expired data is defined as a data older than 24 hours.

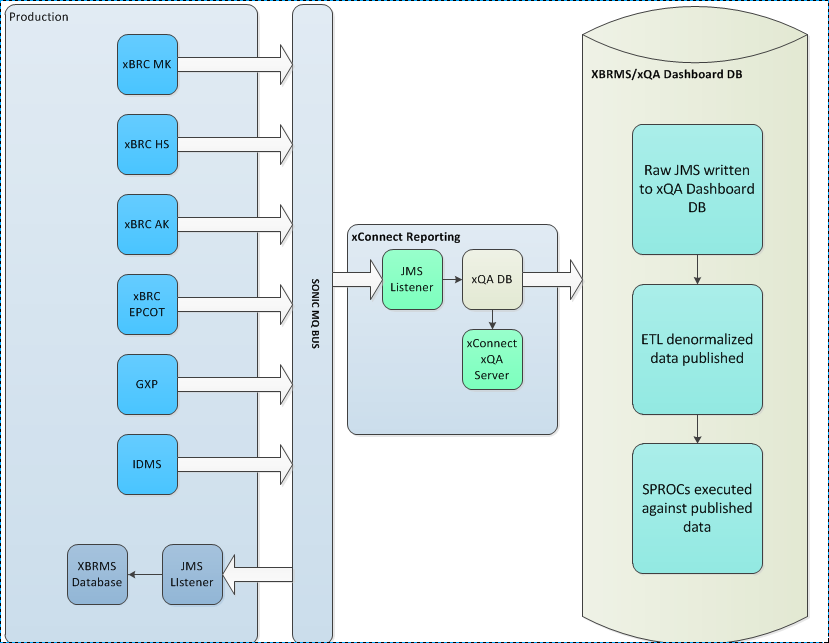
Each logical group is linked to a set of SQL stored procedures.

Business and processing rules are contained in SQL stored procedures. To minimize data and reporting discrepancies and to allow for easy data reconciliation all of the business logic behind xQA is contained within SQL stored procedures.

Database endpoints used by xQA are defined as SQL stores procedures.

Database is self-maintainable and that’s accomplished through a set of nightly archiving processes driven by a SQL Server Agent Job. Data older than 30 days is being sent to a text files, stored on a file server and deleted from the database.

## Data Store Architecture



# xVR

The xVR system attempts to create a rough proximity map of guests as they experience an attraction. It gets a series of messages from the xQA Web-service that give it guest reader event messages and attempts to map those messages to 3D avatars that are walking through the attraction.



xVR uses Unity3d 4.0 (and later) to build a client app that talks to the xQA web service. Unity breaks down component scenes as “levels” and, based on a user’s selection, opens the appropriate attraction level and then begins communicating with xQA.

Unity also supports the concept of level re-use and “prefabs” that are units of code and objects made in such a way as they can be dropped into scenes without much additional coding. This savings in time and effort allows us to generally solve 3D space problems and share them easily between different types of attractions.

Queue behavior is actually a very difficult problem to solve in 3D space but fortunately Unity provides a technology called “NavMesh” which supports path-finding and obstacle negotiation. However, strictly speaking, queue behavior still has to be finely coded and behaved. Navmesh is much analogous to providing us with a launch platform for a rocket, but then getting the rocket to stop, wait in line, wait behind other rockets, and resume patterned movement is the part we are left to solve. This has been an ongoing challenge with xVR.



Also, Unity is cross-platform and usually, with only small amounts of change, a unity application will run on PCs, Macs, iOS and Adroid devices as well as consoles such as xBox and PlayStation. It also has, recently, provided Linux playback ability.

xVR provides a “generic” level container that allows us to view any park attraction that has been provisioned for the xConnect system, for which we don’t have a full 3D model prepared. This generic model simply provides a walking queue with entry, merge and load points that map to the messages from xQA.

