QlikView

Using EDX in QlikView 11

A QlikView Technology White Paper

Published: November, 2011

www.qlikview.com

QlikView

Table of Contents

Introduction

Event Driven Execution (EDX) allows the running of QlikView batch jobs based on external events. The primary usage of EDX is to have an external scheduler or Extract, Transform, and Load (ETL) tool (for example, Informatica) run QlikView batch jobs as part of a larger batch process. There are also other use cases, for example, triggering a QlikView batch when a file arrives and user-initiated batches. This document describes the requirements for each of these three use cases.

The example code that is delivered together with this document uses the new EDX API, which means this document and the example code are useful both when using the EDX functionality for the first time and when migrating from previous versions of the EDX API.

Use Cases

Using an External Scheduler or ETL Tool

Many large organizations have other batch requirements than just QlikView. Such organizations often use a standard tool to run all batch processes, so that the IT training costs are minimized and common operations (for example, logging and re-launching) are handled in one central place. In addition, QlikView batch processing is almost always reliant on new data being available and in many cases this data is created by other batch processes. An external scheduler or ETL tool can be used to prepare the data and then run the QlikView batch to load it.

Most scheduler and ETL tools are designed to run external batch programs via a command line interface. The reasons for this include:

- Batch processes are long-running, so starting a separate process to handle them provides a simple way for the scheduler to monitor the batch processes.
- The scheduler can be configured to run the batch using a particular operating system account that provides the required security context.
- If the batch process fails, the standard operating system error handler can be used to pass this information back to the scheduler or ETL tool.

The main alternative to using a command line tool is using some sort of web services. Unfortunately, web calls cannot be used for long-running processes, since a timeout eventually occurs. A common solution to this problem is to start the process with a web request and then follow it up regularly with subsequent requests to find out if the original request has finished successfully or not.

File Arrival Detection

In many IT departments, the inter-system communication is not orchestrated by ETL or even Enterprise Application Integration (EAI) tools. Instead, files are pushed. For example, a non-QlikView batch job that extracts data from a production database is typically owned by the IT team running the database, which means the team is responsible for creating and configuring the batch job and ensuring that it runs correctly. On the QlikView side, a program that checks if a file has arrived must be running, so that the file can be used in a subsequent QlikView batch.

In this scenario, simply starting the QlikView batch process should be enough. There is no need to follow its execution. The program is typically a Windows service as it runs all the time.

User-initiated Action

The QlikView Automation API provides a number of functions that run QlikView batch scripts, for example, `Document.Reload()`. However, it is normally not a good idea to let users run batch jobs, since a QlikView batch job typically replaces the entire data set of an application used by many users. This means a single user can interfere with the work of all the others unless the QlikView batch is carefully configured. Because of this, the batch functionality is disabled in a QlikView Server deployment.

Despite this, there are legitimate use cases, for example:

- The marketing department at a pharmaceutical company has an organizational division between the analysts and the product managers. The analysts are in charge of loading a dozen external data sources of varying quality. Once the data relating to a particular product is ready, the running of a QlikView batch loads the data and publishes the application to the relevant product managers.
- A software company with thousands of employees wants each employee to have access to their sales data up to the last 30 minutes. The computer resources needed to support such data usage are massive. Since the salespeople only use the system once a week on average, it is decided to allow each user to run a QlikView batch to generate a QVW (for the user) whenever needed.

EDX in QlikView 11

In QlikView 11, EDX runs through the QlikView Management Service (QMS) API. This is a major change from QlikView 10, where EDX is realized by calls directly to a QlikView Distribution Service (QDS). The QMS API is a web service API that uses the Simple Object Access Protocol (SOAP). Client applications make HTTP (web) requests to QMS on port 4799. The system is secured by NT LAN Manager (NTLM) as well as special protective measures to avoid certain types of hijacking attacks known as “time limited service key”. This combination of security means the client application must be written in .NET and therefore the provided example code is .NET projects/solutions developed in Microsoft Visual Studio 2010.

The client application uses NTLM to authenticate a Windows account to QMS. QMS then checks which Windows groups the Windows account for the client application is member of to determine the function calls the user is allowed to make. Most of the QMS API requires membership in a local group called “QlikView Management API”, but to run EDX, a separate group, “QlikView EDX”, should be used. Both groups are local Windows groups on the server where QMS runs.

The client application makes calls to instruct QMS to start a task and in return receives an error code indicating success or failure, as well as an execution ID. The execution ID uniquely identifies the execution of the task as opposed to the task itself. The client application then periodically polls QMS to check the progress of the task execution. This poll request returns a data structure that contains, among other things, the execution status, start time, stop time (if already finished), and a list of new execution IDs. These subsequent execution IDs represent the execution of tasks that are triggered because the initial task has finished. The client application can then follow an entire set of inter-related tasks that together make up a full parallelized batch flow.

Lastly, the example code outputs log information and returns an error code to the operating system. The error code can be recovered in a standard way, for example, by using the `%ERRORLEVEL%` environment variable in a BAT file.

Changes Compared to Previous Versions

From a functionality perspective, the following has changed compared to using EDX in QlikView 10:

- Because communication is with QMS instead of QDS, a QDS cluster can be used to load balance EDX calls. If high availability is required for the batch environment, QMS is a single point of failure. Consider either virtualization of QMS or an active/passive system.
- The overall success or failure code of a QlikView batch can be returned. Previously, the status of the task could be fetched (that is, the definition of the batch job), but not the execution of the batch job.
- The execution of subsequent tasks can be followed.

Notes on Using the QMS API

Finally, some notes on using the QMS API in QlikView 11:

- The EDX password must not be null. Use an empty string, if no password is configured.
- The QDS ID is required in the function call that starts the EDX batch job. However, higher-level privileges than membership in “QlikView EDX” are required to get a QDS ID. The simple solution is to pass an empty GUID for the QDS ID, which means QMS will search for the task in all configured QDS instances. Almost all projects today only have one logical QDS configured, so this can be a useful solution.
- EDX can be used without a QlikView Publisher license. Simply pass the name of the QVW instead of a task name and the reload schedule of the QVW will run.