

Requirements Specification for XXX 6th Form College Bespoke Integration Plugin (Distinction Systems QLS - CELCAT Timetabler 6)

Author: Reg Haversham

Review Team: [APC/JA]

History:

rel 1.0	6 July 2005	following CELCAT in-house peer review
rel 1.1	6 July 2005	following comments from ...
rel 1.2	18 Jul 2005	following receipt of STMAOS data

1. Overview

XXX 6th Form College uses QLS software for student management. CELCAT *Timetabler 6* is to be used for timetabling. Student enrolment data originating in QLS must be replicated in the *Timetabler* membership structure. The College statement of requirements is as follows:

“The plug-in needs to be able to add, transfer or withdraw students when they are entered for particular subjects in QLS. For example, if a student is enrolled on to an AS Level, e.g. Economics, this dynamically adds them to the group in the CELCAT timetable. The same applies for students transferring and withdrawing from a subject or group.” [College, May 2005]

This original statement was modified by ... as follows:

“The dynamic link requested between QL and CELCAT will not work without triggers on QL. We cannot add triggers to the DB so the next suggestion is a delayed scheduled task. This should work fine in practice.” [..., 27 May 2005]

The utility will not follow the standard Integration Plugin format, since it should run automatically as a scheduled task. Users will interact with the plugin via a bespoke interface that allows configuration of the task parameters such as database server names, etc. The time taken to perform the data transfer operations depends upon a number of factors including the amount of data to process, contention for database and network resources, etc.

The Utility should be able to operate as a scheduled task using a command-line parameter that suppresses the user interface and runs unattended.

Where appropriate, each requirement in this document is followed by a reference (in square brackets) showing the origin of the requirement.

1.1 Terminology

QLS	The Distinction Systems QLS software
Timetabler	The CELCAT <i>Timetabler 6</i> software
Utility	The bespoke utility to be developed
User	The user of the Utility
Table	A data table in the QLS or <i>Timetabler</i> database
Column	A Table column (sometimes referred to as a field)
Row	A Table row (sometimes referred to as a record)
AOS	A QLS “Area of Study”

1.2 Documentation & Sample Data

The customer-supplied documentation consists of the high-level statement of requirements (quoted above) and a subset of the College QLS database (qlsdat_data). The tables supplied are:

STCESSD

Area of Study sessional records

STMBIOGR

Student biographical details

STRROOMS

Room records. This table appears to be superfluous to the plugin design.

STRSTAFF

Staff records. This table appears to be superfluous to the plugin design.

STMAOS

Intersect table between STCESSD and STMBIOGR

2. Functional Requirements

In all cases, the Utility should record the origin of the *Timetabler* data, i.e. it should use the *Timetabler* CT_ORIGIN table and associated origin_id columns to store details of the QLS database used.

The Utility should communicate with the QLS database using an ADO connection and the MS Ole DB SQL Server Driver, and should require the User to login. The Utility should store login information (e.g. user name/password) so that it can operate as a scheduled task.

QLS is always considered to be the 'master' source of student, group and membership data. When the Utility runs, conflicting records in the relevant *Timetabler* tables should be overwritten using QLS data. For example:

- a. If the group membership is independently modified in *Timetabler*, the changes should be overwritten next time the Utility runs.
- b. If a student's name is modified in *Timetabler* it should be overwritten during the next run.
- c. If a group is removed from *Timetabler*, the group should be reinstated when the Utility is next run.
- d. If a student is independently added in *Timetabler*, the record should be removed during the next run, subject to the referential constraints of the *Timetabler* database.

A log should be kept to indicate activity and error conditions.

2.1 QLS to CELCAT Timetabler

The Utility should extract the following data from QLS, observing the specified column mappings.

Students

The QLS STMBIOGR table should be used as data source for the *Timetabler* CT_STUDENT table. The column mapping is shown below.

QLS Columns	Timetabler Columns
STMBIOGR.student_id	CT_STUDENT.original_id
STMBIOGR.student_id	CT_STUDENT.unique_name

STMBIOGR.forename + STMBIOGR.surname (concatenation style to be agreed)	CT_STUDENT.name
--	-----------------

Groups

The QLS STCSESSD table should be used as data source for the *Timetabler* CT_GROUP table. The column mapping is shown below.

QLS Columns	Timetabler Columns
STCSESSD.aos_code + '\$' + STCSESSD.aos_period	CT_GROUP.original_id
STCSESSD.aos_code + '\$' + STCSESSD.aos_period	CT_GROUP.unique_name
STCSESSD.full_desc	CT_GROUP.name (truncated to 64 characters as required)

Notes:

- Note the concatenation of aos_code and aos_period to form a unique identifier for the group. The dollar symbol should be used (or another suitable character) to separate the values.
- AOS records having STCSESSD.active_yn = 'N' are ignored.
- The Utility allows an appropriate academic period to be specified and this value is used to filter AOS records (on STCSESSD.acad_period).

Membership

The QLS STMAOS table should be used as data source for the *Timetabler* CT_GROUP_STUDENT table. The column mapping is shown below.

QLS Columns	Timetabler Columns
STCSESSD.aos_code + '\$' + STCSESSD.aos_period	CT_GROUP_STUDENT.group_id (via CT_GROUP lookup on original_id)
STMAOS.student_id	CT_GROUP_STUDENT.student_id (via CT_STUDENT lookup on original_id)

2.2 Determining QLS Changes

[Implementation note]

Since triggers cannot be added to the QLS database, the Utility should determine when group, student and enrolment data has changed in QLS. The options that can be considered during design are:

1. Perform a local join between QLS tables and CELCAT tables to determine where changes have been made since the last run of the plugin. This requires that both the QLS and CELCAT databases are stored on the same database server.
2. If QLS and CELCAT do not share a database server, configure the QLS server as a linked server on the CELCAT database server (a simple task), and then perform the appropriate joins.
3. Maintain a temporary connection-based table in the QLS database whilst the plugin utility is running. This will be used to store a snapshot of the relevant data at the last execution of the plugin and on subsequent runs is compared against the current QLS data to determine changes. This requires write access to the QLS database (but does not write to any QLS data tables of course).

4. Read all of the relevant data from the QLS database each time the plugin runs and compare it directly against data in Timetabler.

The plugin runs as a scheduled task and requires rapid operation, so option 4 is probably not feasible.

College instructs to use Option 1 above [..., 6 July 2005]

3. Phasing

Not applicable.