Historical Extract Manager Requirements Specification

1. Scope

1.1 Overview

A data warehouse is fed with data through a number of what are known as Extract, Transform and Load (ETL) processes. An ETL process will be referred to as a feed in this requirements specification.

A common first step in a feed is to stage the data so that the source system does not have to be hit again if the feed fails. This is very important if the source system is an OLTP database that may contain changing data.

In SQL Server 2005 Integration Services (SSIS), data from any data source can be stored in a raw form at very high speeds, so we do not need to establish a specific staging area schema on a source-by-source basis.

However, while allowing raw data to be stored in an arbitrary location, SSIS does not make any assumptions over where data is to be stored, leaving it up to the feed designer to determine where to store the data.

This component provides management over the raw data store, in order to support limiting the amount of data retained, storing data from multiple feeds and multiple data sources, and allowing other components to locate raw data stores. To do this, the component will store meta-data around each raw data file in a database table.

This component need only provide C# class library through which raw data files can be created, deleted and obtained – it does not need to integrate directly with SSIS, which is the purpose of future components.

1.2 Logic Requirements

1.2.1 All metadata is to be recorded to one or more database tables, through an ADO.NET database connection to be provided programmatically.

1.2.2 Provide a C# interface to obtain a filename for raw data storage

1.2.2.1 A common location for raw data file storage should be configurable through the Configuration Manager component

1.2.2.2 Generate and return a unique filename for the data file

1.2.2.3 Record at least the following information about the data file in the database:

Field	***Type***	***Description***
Creation	Date	The time the file name was requested
Feed Date	Date	This may not be the current time, and will have to be provided to this component through the designed API
Filename	String	The unique name generated
Path	String	The location of the file
Feed Name	String	An arbitrary string identifying the feed
Data Source	String	An arbitrary string identifying the source of the data

1.2.3 Provide a C# interface to allow old data files and metadata to be removed

1.2.3.1 A maximum retention period for data files in calendar days will be configurable using the Configuration Manager component

1.2.3.2 Accept a date parameter that will usually be the current date

1.2.3.3 Delete all data files that were created more than the configured number of days prior to the date provided (1.2.3.2)

1.2.3.4 Flag deleted files in the database

1.2.3.5 A maximum retention period in calendar days for the record of deleted files will be also configurable using the Configuration Manager component

1.2.3.6 Delete all records representing data files that were created more than the configured number of days prior to the date provided

1.2.4 Provide a C# interface to allow old data files to be identified

1.2.4.1 Accept a **Feed Date**, and a **Data Source** name as parameters

1.2.4.2 Return an array of fully qualified filename strings that identify the raw data files

1.2.4.3 Sort the array by **Creation Date**, newest first

1.2.4.4 Return filenames for records that match the **Feed Date** and **Data Source**, and that have not been deleted.

1.3 Required Algorithms

This design will use ADO.NET to access the database.

1.4 Example of the Software Usage

Two SSIS components will be created in a later competition that will leverage the Historical Extract Manager. These components may be dropped into a data pipeline within the SSIS designer, and will take care of providing the Historical Extract Manager's database connection, and sending requests to the C# interfaces defined in this component.

One of these components will check if it is being run for a historical feed date, and in that case it will source data from a raw data file identified using the API defined in requirements 1.2.4. Otherwise, it will source data from the live data source.

Data sourced from the live data source will be multicast to the other component mentioned which will use the APIs defined in requirements 1.2.2 in order to identify a location to store the raw data.

The third API defined in requirements 1.2.3 will be used from a front-end management console.

1.5 Future Component Direction

In the future this component could be generalized to record information about ETL feeds created in other data warehouse environments, like Oracle DW.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None

2.1.2 External Interfaces

The interfaces are described in the requirements section. No specific interfaces must be met.

2.1.3 Environment Requirements

- Development language: C#

2.1.4 Namespace

TopCoder.ETL.Archive.Manager

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Record retention for meta data
- Record retention for physical data files
- A path identifying the location for raw data files

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

No

3.2.2 TopCoder Software Component Dependencies:

Configuration Manager 2.0

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

.NET Framework 2.0

3.2.4 QA Environment:

- Windows Server 2003
- SQL Server 2005 Integration Services, including SP1 and SP2

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

XML documentation must provide sufficient information regarding component design and usage.