SSISTProbe – A Data Extraction Test Automation Framework

Test Automation Framework

Filename:	Test Automation Framwork.doc
Revision:	V1.01
Last Save Date:	Sunday, March 02, 2008
Author(s):	Sathyan S Nair
File Location:	satnair@thoughtworks.com (OR) Sathyan.s.nair@gmail.com
Read Only (public) Link:	

Table of Contents

1. Test Automation Framework

1.1. Framework Introduction

Framework defines the organization's way of doing things – a Single Standard. Following this standard would result to the project team in achieving,

Test Library – Process Definition

- Test Library creation follows a standard design and development process with proper documentation
- Well-defined process would be established for Team communication, Library versioning and Artifacts creation

Standard scripting and Team Consistency

- Scripting standard maintained across the framework library creation, which includes business components, system communications, data check points, loggers, reporters etc.
- Project team is asked to follow the defined scripting standards
- Published standards across the project team would pre-empt the effort involved in duplicate coding, that is prevent individuals following their own coding standards

Encapsulation from Complexities

- Test engineers are encapsulated from the complexities and critical aspects of the code
- Engineers are exposed only to the implemented libraries and tests are executed by just invoking the libraries

Scripts and Data Separation

- Automation test scripts separated from input data store (for example: XML, Excel files)
- No modification is required to the test scripts
- Only input data gets manipulated, for testing with multiple input values

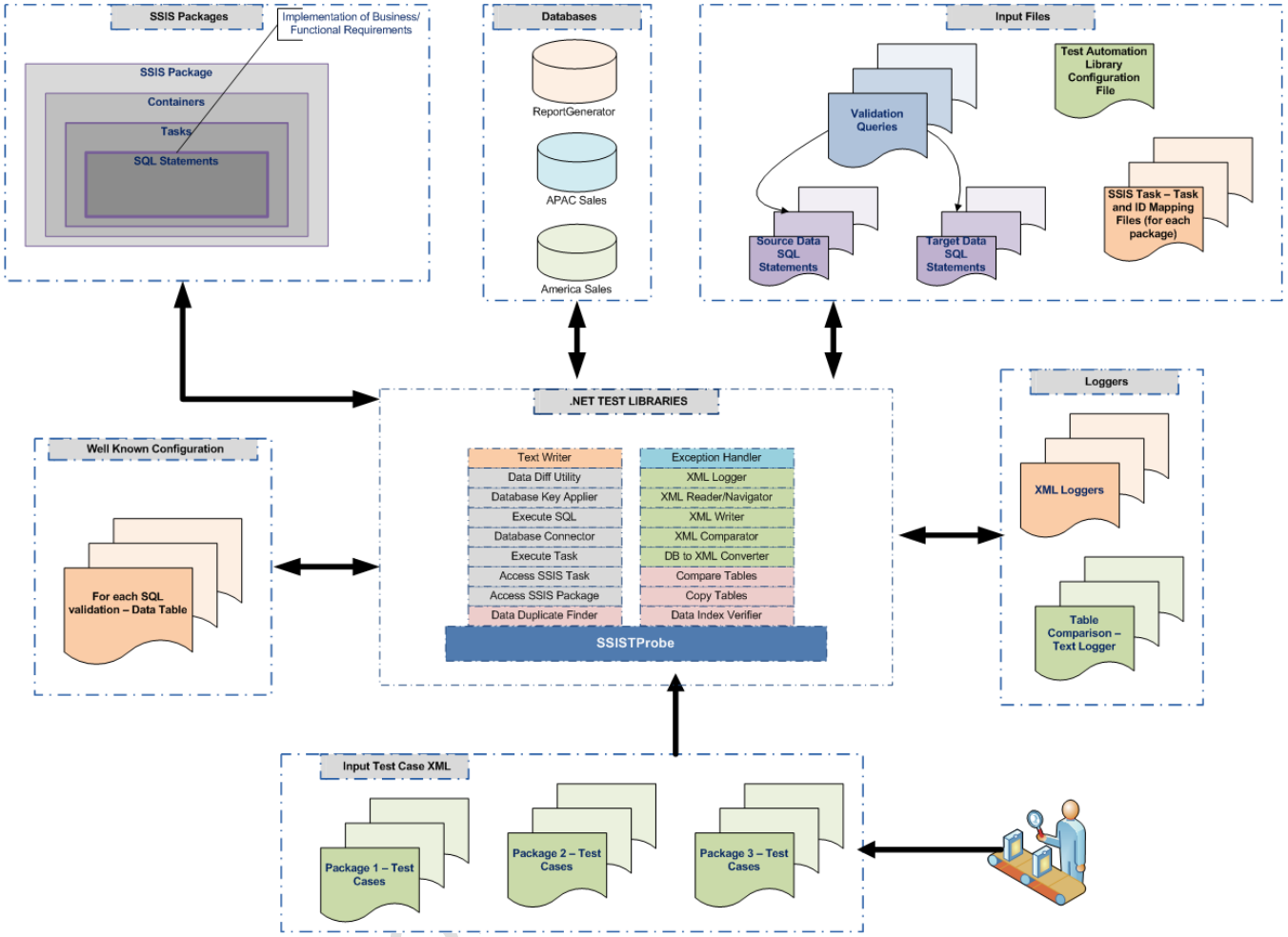
Implement and Maximized Re-Usability

- Establish the developed libraries across the organization/project team/product team, that is publishing the library and providing access rights
- Utilities/components shared across the team
- Usage of available libraries
- Minimized effort for repeated regression cycles

Extensibility and Maintenance

- Complete support for application's new enhancements and existing features modification. For example, re-usable library could be created only for the enhancement features with a minimal effort
- Standard process for script versioning
- Role based access rights. For example access rights such as addition, modification and deletion of scripts
- Project based – Utility/Component access

1.2. Test Automation Framework for SSIS



Overall framework is designed in a way that input XML test cases act as a driver for complete test case execution. Framework contains .net test automation libraries, which act as core key component in driving the complete test automation.

Framework follows a set of user-defined XML tags for specific actions. Framework contains a technical intelligence, which would invoke the relevant library methods based on the XML input tags. This approach would benefit the project team in the following ways,

- Business user having knowledge of XML will be able to create test cases
- Testers encapsulated from complexities
- 97% of coding effort reduced
- Minimal effort during regression cycles
- Able to execute test cases related to actions needed
- Standard Logging for easy analysis
- XML based input - an industry standard
    - Only XML and SQL validation scripts needs to be modified during regression cycles

High level components of the framework includes,

- Input files (Validation files, Config files etc)
- Input XML Test Cases
- .net Test Libraries
    - Database access
    - SSIS Package access
    - SSIS Task validation libraries
- Loggers

1.2.1. Input Test Files

Input test files consists of different kinds of files, which includes

- SQL Validation Files
- Config files
- SSIS TaskID Mapper
- Well-known configuration files

1.2.1.1.SQL Validation Files:

Validation files are basically SQL statements which contain the functional and business rule implemented. These files would be executed to validate the SSIS tasks from a functional and business rules perspective. Validation files needs to be created based on the following reference:

- Functional Requirements and Business Requirements
- Data Column-wise mapping
- Data Column casting
- Types of joins
- Different types of tables to be referred

SQL validation statements needs to be created with following guidelines (recommended):

- Simplify the complex queries present in the SSIS tasks
- Follow a standard naming convention (could be a user-defined naming convention, based on the package)
- Order SQL files in a specific folders, based on the sequence containers
- Create a SQL file for pulling source data (Data before SSIS specific task execution)
- Create a SQL file for pulling target data (Data after SSIS specific task execution)

1.2.1.2.Config files

Configuration file is a XML file, which contains specific configuration information, necessary for the test automation execution. This file would include

- SSIS
    - Pre-defined XML tags for specific test actions, related to SSIS packages
    - Utility file paths
    - Primary Tables file paths

1.2.1.3.SSIS TaskID Mapper

For each SSIS package, this file needs to be generated. Mapper would contain container/TaskID for all the containers/tasks available in the SSIS package. Mapper would help the project team to uniquely identify each task in the SSIS package by assigning a numeric value.

This approach would benefit the project team, by invoking and executing the SSIS tasks using a unique id, overcoming the implementation fact that multiple containers/tasks would have the same name.

1.2.1.4.Well-Known configuration

For each validation task, a data table is created. This data table contains the output data, for different happy and sad paths.

1.2.2. Input XML Test Case

Input XML test cases acts as a key driver for this test automation. Input XML test cases would be created by the tester, having XML knowledge and application knowledge. XML test cases need to be created, by following a list of mandatory guidelines as listed:

- General guidelines of XML file creation
- Each XML Test case will contain a parent tag. For example <SDS_Validation></SDS_Validation>
- Test methods will be created under the parent tag. For example <SDS_Validation><InsertAccount>.....</InsertAccount></SDS_Validation>
- Each test method must contain a log file path. For example <Log>C:\log.xml</Log>
- Appropriate tags relevant to test actions, must be listed. To see a listed of valid XML tags, please refer to the below attached document

On completion of the input XML test case creation, these Xml files would be invoked by a C# test automation project.

1.2.3. .Net Test Automation Libraries

Library design involves identifying requirements from multiple areas. At a high level, this includes (not limited to):

- Identification of necessary actions related to SSIS related to application functionalities
- Communication between the utilities/components (for example: data check-point components communicating to the logger etc)
- Customized user defined logs

Two libraries at high level implements the following sub-libraries:

1.2.3.1.Database libraries

Database libraries implement the functionality of accessing SQL databases/ tables, comparing two tables, moving tables, dynamic SQL table creation, table data correctness verification, verifying table indexes etc. SMO and SQL libraries (.net C# libraries) form as a base for these database libraries.

1.2.3.2.SSIS package access libraries

SSIS package libraries implement the functionality of accessing individual tasks present inside the containers. Libraries access the SSIS package using COM interface and relevant tasks are invoked based on the user input id's (from SSIS TaskID Mapper).

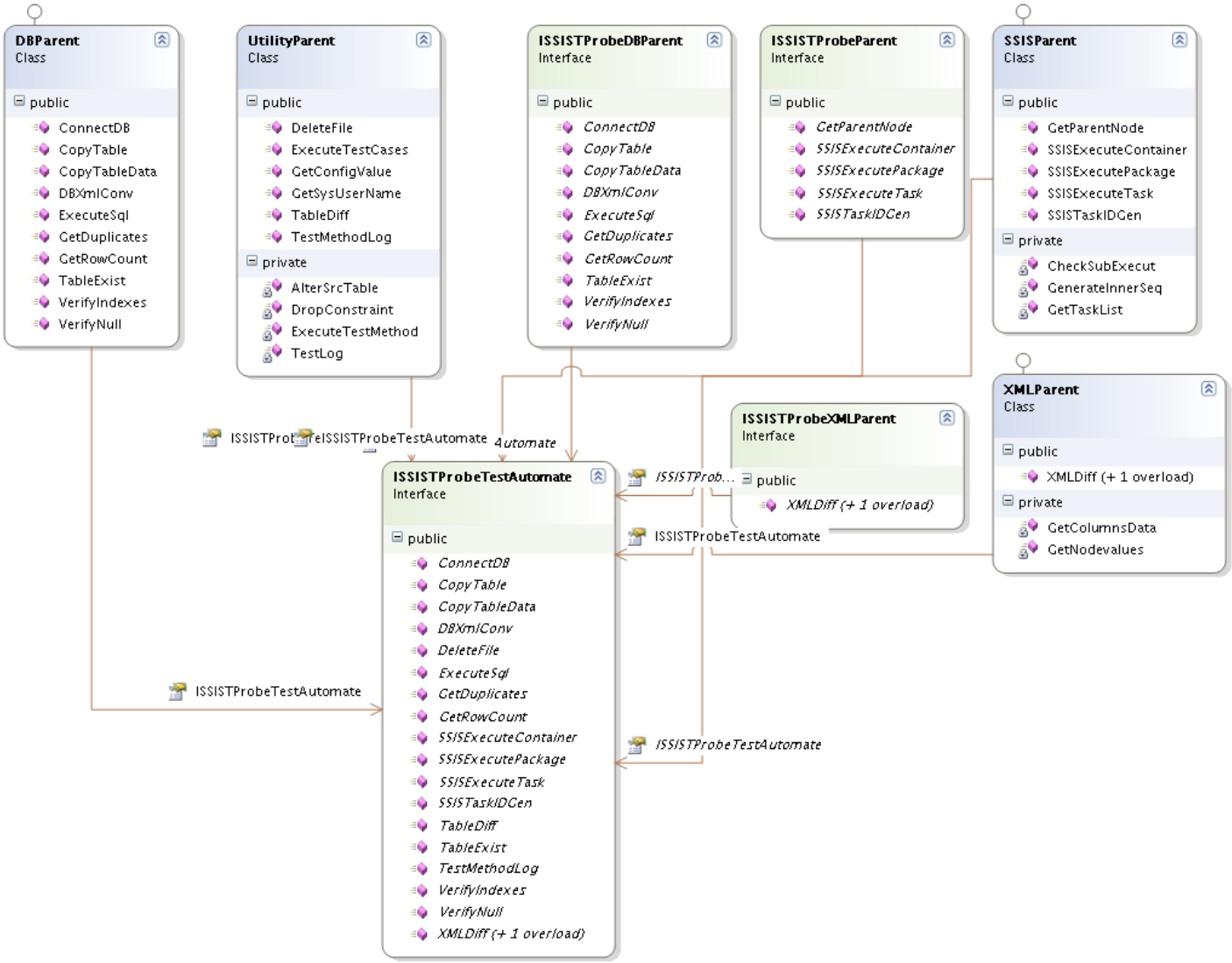
1.2.3.3.Loggers

Loggers act as common library, which handles all the exceptions and user-defined messages to be logged from different libraries. Based on the type of message (i.e., Exception, Error, User-defined message) dynamic XML tags will be generated.

1.2.3.4.Cross Tab- Column Algorithm Library:

Converting a cross-tab XML data to columnar SQL data table will be implemented using this library. An intelligence algorithm will be developed in this library to do the conversion of data. This library will be used for data comparison from the report builder UI against the database data.

SSISTProbe – This library is stacked with majority of the test automation functionalities. At a high level, libraries are created for SSIS package access, DB access, XML Converters, Data Diff Utilities etc. Using this library, a complete SSIS package could be test automated.



1.2.4. Logs and Reports:

Reports related to each test automation task execution would be logged in a custom defined formatted way. Each action related to the test method will be logged with appropriate attributes. Time of each task execution will also be logged. For example,

Result Report for a particular task Execution:

```
<LogInfo>
  <LogMessage />
- <LogMessage>
- <Transfer_tblSDSAccountNote_Diff>
  <UserMessage>FunctionName =SrcMake; Excepted Result=; Arrived Result=602990 ;</UserMessage>
  <DateTime>6/5/2007 9:18:00 AM</DateTime>
  <UserMessage>FunctionName =SrcExecQuery; Excepted Result=; Arrived Result=120598
     ;</UserMessage>
  <DateTime>6/5/2007 9:18:18 AM</DateTime>
  <UserMessage>FunctionName =SrcExecQuery; Excepted Result=; Arrived Result=-1 ;</UserMessage>
  <DateTime>6/5/2007 9:18:23 AM</DateTime>
  <UserMessage>FunctionName =ExecuteTask; Excepted Result=; Arrived Result=Success
     ;</UserMessage>
  <DateTime>6/5/2007 9:19:52 AM</DateTime>
  <UserMessage>FunctionName =TgtExecQuery; Excepted Result=; Arrived Result=120598
     ;</UserMessage>
  <DateTime>6/5/2007 9:20:10 AM</DateTime>
  <UserMessage>FunctionName =TgtExecQuery; Excepted Result=; Arrived Result=-1 ;</UserMessage>
  <DateTime>6/5/2007 9:20:10 AM</DateTime>
  <UserMessage>FunctionName =CompareTables; Excepted Result=; Arrived Result= ;Testcase
     =Pass</UserMessage>
  <DateTime>6/5/2007 9:21:49 AM</DateTime>
    </Transfer_tblSDSAccountNote_Diff>
- <RG_Truncate_tblSDSContact_Diff>
  <UserMessage>FunctionName =ExecuteTask; Excepted Result=; Arrived Result=Success
     ;</UserMessage>
  <DateTime>6/5/2007 1:54:44 PM</DateTime>
  <UserMessage>FunctionName =VerifyTable; Excepted Result=0; Arrived Result=0 ;Testcase
     =Pass</UserMessage>
  <DateTime>6/5/2007 1:54:45 PM</DateTime>
    </RG_Truncate_tblSDSContact_Diff>
    </LogMessage>
  <LogMessage />
  <LogMessage />
```

Data Comparison Log

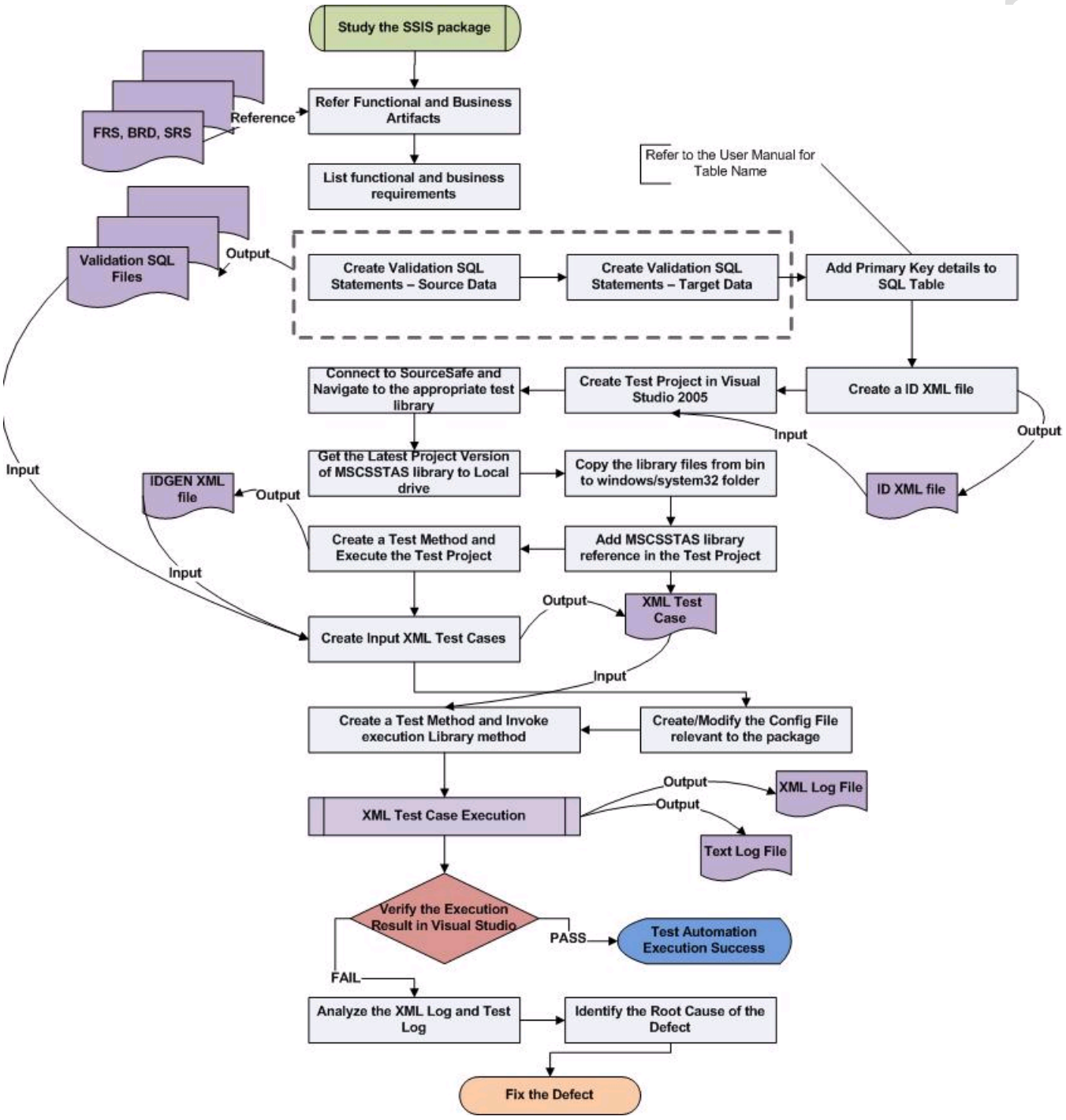
Text based logger implement the functionality of logging the diff data along with the primary key, during data table comparison. A sample Log information is provided below:

Table [ReportGeneratorT].[dbo].[src_Final_SDSRgsubsidiary] on sgprbuitsql03\test and Table [WorkArea12].[dbo].[tgt_Final_SDSRgsubsidiary] on sgprbuitsql04 are identical.
The requested operation took 3.5434288 seconds.
Table [WorkArea12].[dbo].[src_Final_SDSRGContactAccountType_Distinct] on sgprbuitsql04 and Table [WorkArea12].[dbo].[tgt_Final_SDSRGContactAccountType] on sgprbuitsql04 are identical.
The requested operation took 23.3612005 seconds.

2. Library Usage Workflow

High level workflow of library usage is provided below:

2.1. SSISTProbe Usage



3. Moving Forward

In future based on the new features added to the application, minimal modification effort would be required to accommodate the following. The items which may require enhancement, only for added features would include:

- C# test libraries
- Config files
- Additional XML test cases