



## Siebel Test Automation

### Executive Summary

Automation testing is finding favor in testing projects across various technologies and domains. It is becoming increasingly popular with Siebel applications because of its ability to reduce testing effort, which can contribute to a faster rollout/upgrade or lower maintenance costs for an already deployed application. This paper attempts to answer the following questions:

- What is the relative ease of automation testing of different areas of a Siebel application?
- What are the criteria for selecting appropriate functional test case candidates for automation?
- Given the fact that multi-country deployment of Siebel applications is fairly common, what are the salient points when considering automation testing for a Siebel deployment project?

Testing of Siebel applications is among the foremost challenges for clients planning on implementing Siebel in their IT enterprise. The fact that there are various modules of Siebel applications (Sales Force Automation, Marketing, Field Service) combined with different types of client instances (Web client, wireless Web client, handheld client, mobile Web client and dedicated Web client), creates a wide variety of testing scenarios for any Siebel implementation project. Post-implementation, there remains a challenge in terms of a disciplined testing effort for regular changes and upgrades. And because most large Siebel implementations span different geographies and languages, testing can be even more complex. A question that is normally faced during any Siebel testing project is, "Can we automate Siebel testing to a large extent to reduce the testing effort and cost and enable test case reuse for future projects?"

The selection of suitable areas of automation, given the various candidate areas from a typical Siebel implementation, is of paramount importance in order to optimize the automation script development exercise. It is the objective of this white paper to identify suitable areas of automation and provide guidelines on how the automated testing should be approached.

### Relative Ease of Automation of Siebel Test Areas

There are four broad testing areas for a Siebel application: Functional testing, analytics testing, data testing and interface testing. Careful consideration of each of the aforementioned areas leads to the following observations regarding automated test case suitability:

#### Functional Testing

This is carried out to validate the business application components of the system. Functional testing is conducted on units, modules and business processes in order to verify that the application is functioning as requested. This covers areas such as validating whether a module is performing correctly (e.g., activities are successfully created), Siebel workflows and assignment rules are working properly, and UI layout (i.e., look and feel, drop-down menus and test fields) is properly validated. Functional testing offers the maximum potential for automation of test cases. The following are a few of the areas where automation testing may be explored:

- Field validation in multi-value groups and pick applets
- List of Values (LOV) validation
- Modularizing test steps that repeat across multiple test cases (e.g., logon, screen navigation)
- Validation of multi-lingual label translations



### Data Migration Testing

Data testing is another appropriate candidate for test automation. Loading of data from external sources to a staging table, staging to EIM and EIM to database servers, can be tested for data integrity through the migration process. After loading the data in OLTP database, SQL queries can be used to check whether all the valid data is loaded properly in the database. For a large Siebel implementation where multiple instances of the same application are being rolled out, this would entail a huge effort. In order to test migration of data efficiently, and also with reduced effort, automation testing can be introduced. Modern day test automation tools provides support for directly querying databases without accessing the GUI, which facilitates automated testing of data.

### Interface Testing

This is typically not a good candidate for automation. Testing of the interface of a Siebel application with a third-party application can be fully automated only when the third-party application fully supports test automation. If otherwise, inbound/outbound data passing through the Siebel system interface needs to be generated and verified using driver and stub routines, resulting in extensive effort in certain situations.

### Analytics Testing

This is comprised of such things as report layout testing (e.g., checking the format and overall

appearance of the report) and report data testing (e.g., validation of data in the back-end and whether it is being represented correctly in the report). Analytics testing doesn't qualify for automation because activities such as format verification are difficult and cumbersome to implement through automation scripts. Moreover, the fact that data validation would involve setting up and executing SQL queries which often results in run-time changes and optimization, which renders the scripts to a great extent as unsuitable for reuse, thereby defeating the purpose of automation.

### Siebel Test Automation Approach

Figure 1 below highlights a suggested approach for automating testing of Siebel applications. This approach is based on the experience of the authors of this paper while working on multiple Siebel automation projects across domains such as pharmaceuticals and insurance.

### Identification of Test Case Candidates for Automation

When considering test automation of a Siebel application, keep in mind that it is never possible to achieve 100% automation coverage. Unless care is taken while choosing test cases which are suitable candidates for automation, it may result in loss of productivity. Dedicating time to automate test cases which will be executed for few and far between instances cannot justify the development effort required. For example, in a

## Siebel Test Automation Approach

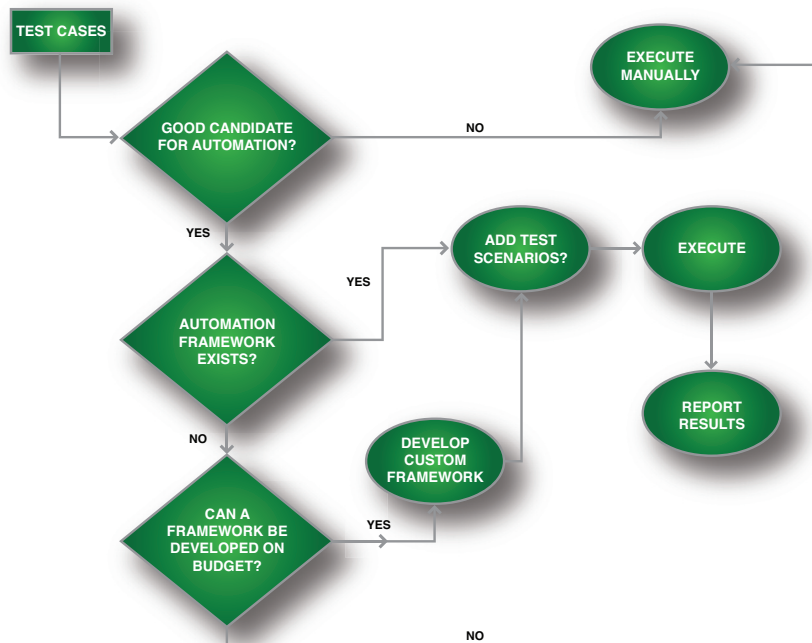


Figure 1

multi-country Siebel application rollout, automation of test cases involving country-specific localization requirements, which may be executed only once or twice in the entire testing phase, should be avoided. It is therefore imperative to develop selection criteria while considering test cases candidates for automation.

### Developing the Automation Framework

At the very abstract level, an automation framework is comprised of a set of standards, guidelines and utilities which are utilized for any test automation project. If a framework needs to be developed, a decision needs to be made on what type of framework should be used. For example, developing a keyword-driven framework takes a lot more time and technical expertise than developing a data-driven framework. On the other hand, creating new test cases using a keyword-driven framework is much faster and easier. It is critical therefore to understand the tradeoff and decide accordingly. Regardless of the type used, the framework should have support for most of the common tasks that need to be performed easily.

Apart from the built-in functionalities, the framework should also clearly define the coding standards, the script review and documentation process to be followed across projects. For any serious automation project, the life time of the automation suite is much longer than the duration of any specific automation engineer's involvement with the project. Adherence to proper coding standards and adequate documentation of standards reduces the learning curve for any new automation engineer joining the team.

Figure 2 highlights the automation framework that was used as part of a major project involving

testing of a Siebel SFA application which was implemented across twenty three countries in Europe for a Fortune 500 pharmaceuticals major.

### Estimation of Effort

Effort estimation for an automation project is a vast topic in itself and is beyond the scope of this paper. However, there are three critical components that need to be carefully considered for any automation exercise: effort estimation of the time required for developing the automation framework; effort estimation of the time required for developing and doing dry runs on the automation scripts; and, effort required for script maintenance.

### Configuring Siebel Environment for Test Automation

Automation support is not enabled in a Siebel environment by default. Before starting an automation exercise, it needs to be confirmed whether a License key is available for enabling automation. This license key, which is a 24 to 30 digit number, needs to be procured from Oracle and should be added to the Siebel server.

### Adding Test Scenarios/ Re-usable actions/ Function Libraries

Once the framework is ready and all configurations are in place, specific test scenarios can be added. These test scenarios leverage the available support from the framework to quickly automate specific test steps. Generic scenarios for a specific flavor of Siebel fall under this category.(e.g., a medical representative creating an unplanned call with a doctor in Siebel ePharma application or an insurance agent creating a new quote for an individual client in the Siebel eFinance application.) Multiple scenarios

## Automation Framework Developed for a Large Pharmaceuticals Client

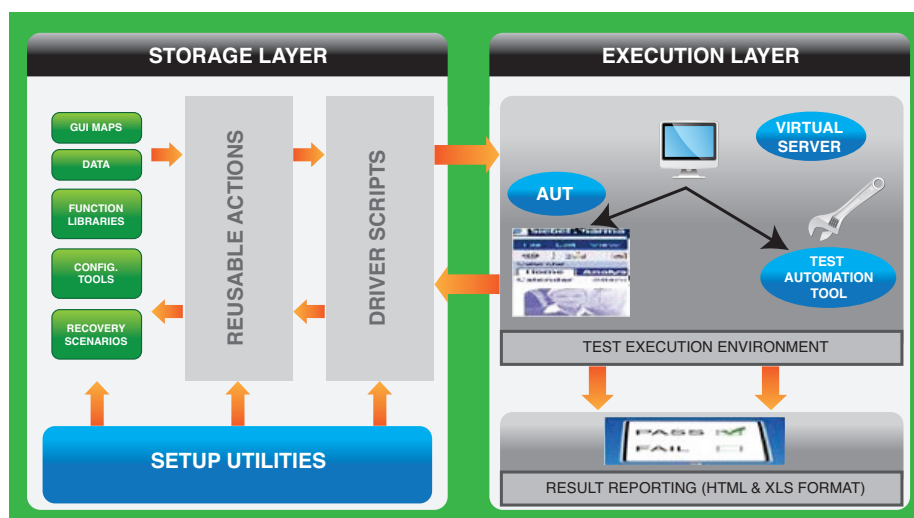


Figure 2

are then combined to create application specific test cases.

### Automation Execution

The biggest challenge faced by automation teams while executing automated test cases in a Siebel environment is the hardware resource requirement of the Siebel application as well as the automation tools. It is suggested to have dedicated computers / virtual servers with a sufficient amount of memory available for execution of automated test cases. For example, in the case of a multi-country Siebel deployment, country-specific setups needs to be maintained in different machines, as having multiple language packs installed on same machine may have unforeseen impact on application behavior (i.e., submitting data on a form applet may result in unexpected error message, hence causing the test to fail). When multiple machines are not available due to budget constraints, care should be taken to have the different language packs installed in different folders.

### Reporting

Reporting of results of automated test runs needs to be carefully planned. The same scripts may run multiple times in different cycles or for multiple countries. Hence, results need to be mapped correctly with the appropriate test run. While this may sound trivial, lack of care in designing the reporting structure may cause loss of test results in the long run. A sample reporting format is shown in the Figure 3:

## Reporting Structure for a Large Pharmaceuticals Client

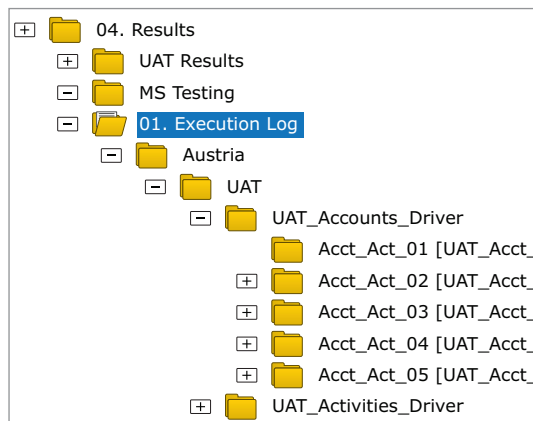


Figure 3

### Functional Testing: Criteria for Identification of Test Cases for Automation

Having explored the suitability of automation testing in various Siebel areas it is evident that func-

tional testing of the application is one area where considerable automation may be achieved. However careful consideration has to be made when selecting candidate test cases for automation. The following guidelines will assist in selection of suitable candidate test cases for Automation:

#### Number of Test Cycles and Potential Re-usability of the Test Case

This is the single most important factor in choosing a test case for automation. If a test case needs to be executed once or twice, the development effort required for automating the test case may exceed the effort required to execute the test case manually. A useful rule of thumb to consider is that a test case needs to be executed at least thrice during its lifetime to consider it a suitable candidate for automation. From a Siebel perspective it therefore makes sense in automating test areas such as Screen/View/LOV validation, logon/password sync and also test cases where basic opportunity/activities are created by filling in certain mandatory information.

#### Criticality of the Test Case

The test case should exercise a critical functionality in terms of either technical complexity or business importance. High-priority test cases should be chosen for automation first, while lower priority ones should be automated if time permits.

#### Complexity of the Script Development Effort

Even if a test case is exercising a very critical module/functionality of the application, it may turn out that automating the test case requires more effort than is needed for the project. For example, with the Siebel ePharma application, creation of a meeting by using the drag-n-drop to Siebel Calendar object is a common functionality that is used very frequently by the users. But incorporating the support for automating this functionality may require considerable effort and not worth the time.

#### Test Data Type

**Input Test Data Type:** The type and format of test data required to automate a test case is an important factor to consider. To have a form filled automatically, it may be needed to create random character strings to be entered to the text entry fields and specific dates in predefined format (DD/MM/YYYY or MM/DD/YY) may need to be input in required fields.

#### Output Data Type

The test case may need to extract/compute some specific data from application which may be required for the execution of a subsequent test case. Commonly three different methods are used

for storing such data. Each method has their advantages and disadvantages.

**1. Storing data in external file** - This is the most commonly used method. Output data is computed and stored in text (.txt) or spreadsheet (.xls) format in some shared location. While using this method, care should be taken that the files are not stored in the local drive of a system. If the output data is stored in local drives and the subsequent test cases are scheduled to be executed on a different machine, these machines will not have access to the local drive of another machine. Also, for security reasons write access may not be provided in the test machines where the automation is executed, and the tests will fail.

**2. Passing data as output parameter** - A test may pass data as output parameter to another test and the second test may receive those data as input parameter. This method is fastest as the data is readily available in memory and is not fetched from secondary storage. But to successfully use this method, all tests that require interchange of data need to be executed in the same session without a break. Once a session ends, the data is lost. This also prohibits partial execution of subsequent test cases, which may be required for retesting of test cases.

**3. Passing data as runtime data table** - This method is technically similar to method one described above, but the special data format is provided by some leading automation tool vendors. This feature may not be present in all automation tools. Runtime data tables are used when the same test case is executed against a multiple set of data and the output needs to be tracked for each set.

## Automation Testing in a Multi-Country / Multi-Lingual Deployment Scenario

Large Siebel deployment projects often involve multi-country roll-outs wherein the application has to be rolled out in multiple languages. Such rollouts are also characterized by a common set of functionality applicable across all countries (core requirements) in addition to certain unique country specific requirements (localization requirements). In such a multi-country deployment scenario the following salient points need to be considered:

### Prioritization of Test Cases for Execution

It is advisable to initially do an assessment on the criticality of the requirements for each country. Post this exercise, a prioritization of the test cases for automation should be conducted for each country.

## Changes in Automation Approach

Normally, automation tools identify test objects used in the application by means of the text displayed on the GUI of the object. For example, a button with caption "Submit" will be recognized by the text "Submit" displayed on it. When the application is deployed in multiple languages, this becomes a serious problem for the reusability of the scripts across different languages. This difficulty is easily overcome in Siebel applications, as all the objects used in a Siebel application have a unique repository ID which can be utilized to uniquely identify the object. This ID remains same even when the GUI text of the object changes for different languages. The framework designer must enforce the use of repository ID as the object identifier in all automation scripts if they are intended to be used for multiple languages.

Reporting structure of the automation framework should also be amended to store the country-specific results separately. This adds one layer of complexity on the reporting structure of the automation framework, but offers long-term benefits by providing a clear picture of the test execution result for individual countries.

### Language Validation Approach

Availability of comprehensive reference data is a pre-requisite to accurate translation validation. The reference data should be arranged in a tabular form, as indicated in Figure 5.

From the point of view of any modern test automation tool, any text displayed in a Siebel application is a run-time property of an object of the application. For example, consider the scenario of validating the translation of the caption of a button. Let's assume that in the application under test, a "Submit" button is displayed in "Customers" applet of "Customers" view. The caption of the button will read "Submit" in English, "Einreichen" in German and "Soumettre" in French version of the same application. To test this, first a table is created as shown in Figure 4. The table depicts complete details of the navigation path to the button (i.e. which screen, which view and which applet the button belongs to). Along with the navigation information, the correct translations are also stored in the table. The automation tool first navigates to the relevant area of the application and then validates the displayed text using this reference table. Using this approach, validation of translation can be done in a very generic way so that the same script can be reused for multiple applications.

## Stress on Re-usability

As long as repository ID is used as the identifying property of the Siebel objects used in the automation scripts, the same scripts can be used for all countries. Care should be taken for situations where a particular Siebel object is present for one country but is absent for another country due to localization requirements. This type of scenarios must be handled individually for each country.

## Table for Translation Testing

Screen Repository Name	View Repository Name	Applet Repository Name	Field Repository Name	English	German	French
Customer Detail Screen	Customers View	Customers	Submit Button	Submit	Einreichen	Soumettre
Account Screen	Account View	Accounts	New Visit	New Visit	Besuchen Sie New	Nouvelle visite

Figure 4

## Conclusion

An automation test strategy should be developed based on the objective of your automation exercise (e.g., reduce testing cost, increase test coverage etc). If an automation testing exercise is to be attempted for a Siebel test automation project, it is advisable to start with suitable test candidates from a functional and data testing perspective and then move to analytics or interface testing areas. Adequate care should also be taken when selecting test candidates for automation within these areas and decisions should be driven by repeatability, relative ease of automation and criticality of the test case.

## About the Authors

Parantap Samajdar is a CRM Testing consultant with more than six years of experience in test automation and performance testing. He holds a Bachelor of Engineering Degree in Computer Science. Parantap can be reached at [parantap.samajdar@cognizant.com](mailto:parantap.samajdar@cognizant.com).

Indranil Mitra is a Principal Consultant and has extensive experience in test analysis and management gained from multiple CRM testing projects. He has more than 10 years of experience and holds a Bachelor of Engineering Degree in Instrumentation and Electronics. Indranil can be reached at [Indranil.Mitra2@cognizant.com](mailto:Indranil.Mitra2@cognizant.com).

Ankur Banerjee works within Cognizant's CSP Testing Pre-Sales and Delivery team. He has more than 10 years of experience across various domains and technologies. Most recently he handled test management of a large Siebel upgrade project for a major insurance client. Ankur holds a MBA (Systems) and a Bachelor of Engineering degree in Civil Engineering. Ankur can be reached at [ankur.banerjee@cognizant.com](mailto:ankur.banerjee@cognizant.com).

## About Cognizant

Cognizant (Nasdaq: CTSH) is a leading provider of information technology, consulting, and business process outsourcing services, dedicated to helping the world's leading companies build stronger businesses. Headquartered in Teaneck, New Jersey (U.S.), Cognizant combines a passion for client satisfaction, technology innovation, deep industry and business process expertise, and a global, collaborative workforce that embodies the future of work. With over 50 delivery centers worldwide and approximately 104,000 employees as of December 31, 2010, Cognizant is a member of the NASDAQ-100, the S&P 500, the Forbes Global 2000, and the Fortune 1000 and is ranked among the top performing and fastest growing companies in the world.

Visit us online at [www.cognizant.com](http://www.cognizant.com) for more information.



### World Headquarters

500 Frank W. Burr Blvd.  
Teaneck, NJ 07666 USA  
Phone: +1 201 801 0233  
Fax: +1 201 801 0243  
Toll Free: +1 888 937 3277  
Email: [inquiry@cognizant.com](mailto:inquiry@cognizant.com)

### European Headquarters

Haymarket House  
28-29 Haymarket  
London SW1Y 4SP UK  
Phone: +44 (0) 20 7321 4888  
Fax: +44 (0) 20 7321 4890  
Email: [infouk@cognizant.com](mailto:infouk@cognizant.com)

### India Operations Headquarters

#5/535, Old Mahabalipuram Road  
Okkiyam Pettai, Thoraipakkam  
Chennai, 600 096 India  
Phone: +91 (0) 44 4209 6000  
Fax: +91 (0) 44 4209 6060  
Email: [inquiryindia@cognizant.com](mailto:inquiryindia@cognizant.com)