Synthesys for Systems Integrators

Overview
This document is for anyone who wishes to extend Synthesys, or integrate with it. It assumes that the reader had read the companion document 'Synthesys Technical Summary', and is also familiar with how to use Synthesys.

Document History
This document is new for versions 2.5 and 2.6 of Synthesys.

First draught	Luke Harris, 7th February 1999
This version	Luke Harris, 12th February 1999
Updated for 2.3	Luke Harris, 31st May 2001
Updated for 3.1	Luke Harris, 5th June 2003
Updated for 3.2	Luke Harris, 13th June 2005
Reviewed for v4.3	Luke Harris, 3rd January 2013

For clarifications and suggestions, please email me, Luke.Harris@noetica.com

Summary
Synthesys is an open platform which allows 3rd party customisation in a number of ways. The key ones are…

- Web Services
- Links to web pages.
- ODBC compliant database.
- ActiveX components in callflow editing.
- SynCTI, allowing control of the CTI aspects of Synthesys. See the document 'Synthesys CTI Integration' for details of this aspect of Synthesys.
- UMS, allows plug-in modules to communicate with SMS, email and other messaging services. See separate documentation for details on how to integrate with the UMS system.

This document provides examples and details of integration methods.

Case Studies

Credit Checking and supply
Synthesys is used in a system to provide a front end sales environment to guide a call centre agent through the process of getting all the information to approve the credit application, as well as making the sale to the customer.
The customer already had an underwriting and order processing system running on an IBM AS/400 system running DB2, Synthesys needed to integrate with this.

During the call, lookup information from the enterprise system is required. Since the system was located over a WAN, a service running on the Synthesys server regularly polls the Enterprise system to cache the information for ready use by workstations in the call centre. An ActiveX control dropped into the 'Take Calls' module displays this information to the call centre agent.

The Synthesys Export service is used to process each call when it was completed by the operator. This generates rows in a database table within Synthesys. Then a

specially written module checks for rows in this database, and when a new row is found, it generates the appropriate files to send on to the IBM Mainframe system to create a new customer and calls an API on the mainframe. If the mainframe link is unavailable, the module can retry the insertion later.

Each night, the Synthesys 'Selective Queuing and Import' facility connects to the DB2 database (using ODBC). It retrieves updated information from the AS/400 system to insert back into the Synthesys database. This serves two purposes. One is to make sure that all Synthesys CRM information matches that in the Enterprise system. Secondly Synthesys automatically queues certain customers for a callback from the call centre.

Note that it is also possible to direct the Synthesys CRM component to use live data from an enterprise system, again via ODBC. This solution is usually not selected though, since it can prove difficult to guarantee uptime and responsiveness in these circumstances.

Implementation time for this solution was approximately eight months in total.

Insurance Claims Processing

Synthesys was selected to front an insurance claim processing system. Synthesys was required to script the gathering of information from the customer, and to compare the gathered information with information held on the system. Finally if all checks were OK, Synthesys would send the gathered information to the enterprise system.

The information Synthesys needed was held in a number of different tables, and some data additionally was only available by running stored procedures. To solve this problem, we wrote a number of ActiveX controls and services, which encapsulate the logic to interrogate the insurance claims system for customer information and to either display it to the agent, or branch the script based on the information captured.

At the end of a call, the Export service directly populates a table in the Mainframe system; the mainframe system then generates a new claim from this data and starts processing it.

Implementation time for this was three months.

CRM and Order Processing

A large multinational have an enterprise system running via a WAN to an IBM and HP mainframes in Tampa, Florida. They wanted to move the call centre away from the old 'dumb terminal' solution to reduce the training costs.

The solution here is 'screen scraping', where Synthesys automatically navigates the mainframe screens by feeding keystrokes into them, and then reads the responses back. We then wrap up a number of complex processes on the mainframe such as 'Change of address' into a simple one step operation for the call centre agent.

We use the 3rd part package 'OnWeb' from NetManage to accomplish the screen scraping, since it provides a high level language used to define screens and transactions on the mainframe. We then use ActiveX controls to manipulate the OnWeb API and control interaction with the call centre agent.

Implementation time for this solution was four months.

Sending SMS messages
Two customers have rather different requirements here, both solved using our open UMS ('Unified Messaging Service') system.

A Swiss provider of paging and message services have there own Unix system to send paging messages. We wrote a UMS Service which talks directly to the Unix system over an RS232 link, to send both paging and SMS messages to that system and retrieve responses from it. Implementation time approximately six weeks.

An Australian company had a slightly different set up. Their provider of SMS services had a Web Service interface, so we could invoke the sending of SMS messages simply by sending a suitably formatted XML request over HTTP protocols. Implementation time ten days.

Fault Reporting
The call centre for a council used one enterprise system to record all the details of the properties they maintained, and another enterprise system to log faults for these properties and to organise engineer callouts.

In both cases the enterprise systems provides a proprietary API to interrogate the information held on the systems. We wrote a number of ActiveX controls to display housing information and also to locate and book time from an appropriate engineer to investigate the fault and resolve it.

Implementation time was two months.

Details of Integration Methods

Web Services

Synthesys currently provides two Web Services APIs, one providing access to the CRM subsystem, the other providing access to the Dialler queue. Contact Noetica for information on these and other forthcoming APIs.

Links to Web Pages

Synthesys allows context-sensitive links from a callflow or part of a callflow to an Intranet or Internet web page.

Although this sounds trivial, it can be used to put together powerful applications if used imaginatively. For example, the web page might contain a Java or ASP.NET application, which could interface with another system. Or the web page might be a search page, which could query an on-line database.

Examples
One of our customers uses web pages in two different ways :

1. They are selling a discounted energy package (gas and electricity) on behalf of a client. The client has a price calculator located on a web page. At the appropriate point in the script, the operator presses F1 to bring up a Synthesys web page, which redirects to the appropriate Internet page containing the client's price calculator.
2. They are also selling domain names for a technology client. Before calling someone, they bring up an Internet search page, and use it to search for

appropriate domain names to sell before using the Synthesys outbound module to continue.

ODBC compliant database

Synthesys installs an ODBC data source called 'Phoenix' on the server and all workstations. This is a link to the live Synthesys database. Rather than give an exhaustive (and uninformative) list of all tables, I here give a summary focussing on key elements in the database.

Phoenix_Sequence
The master table, with one entry per inbound or outbound callflow transaction. Each entry includes the time it was run, the agent, the campaign it was associated with, the callflow application. All sequences are identified by the Sequence_ID, which is generated when a call is first created.

Phoenix_Account and Phoenix_Campaign
Each callflow application is associated with a campaign record. For ease of use, campaigns are grouped together into accounts; these might be business customers of the call centre, or some other logical grouping.

Phoenix_Event
Each sequence has one or more events associated with it. These are the events that are displayed in the call tracker; the first event is always the inbound or outbound call that started the sequence of events; following events are follow up actions, such as calling an engineer, re-editing a transaction, or adding a note.

Other Stock Tables
There are tables for users on the system, machines on the system, logon sessions.

Custom tables, used for storing call data.
These are created in Phoenix when the callflow application is released. For backwards compatibility with old transactions, columns and tables are not deleted with new callflow releases.

Each campaign is assigned a five character prefix, such as AAA01. Then data from the main flow of a callflow is put in a table called AAA01_Main.

If the callflow branches, then a new table is used for each branch, and is named after the branch, so for example we might have a branch called AAA01_Sales.

Similarly, data for a subscript is also placed within a table, AAA01_Subscript for example.

Columns are given names 'SECTION_CONTROLNAME_PROPERTY'. So each column has a name based on the section it is found in, the name given to the control, and the name of a property of that control.
Since this can result in excessively long names, a short form of each name may be used. If the Property is 'VALUE', then this will be omitted from the column name. Also, if the Section is named the same as the Control, this also will be omitted.

To programmatically find out the names given to columns, the callflow application file can be examined (this is a simple to parse text file); the syntax 'COLUMNNAME=' is used to name the columns.

CRM Tables
Each CRM customer prefix results in a set of tables, the most interesting being the table CS_PFX_Customer, where PFX is the customer prefix. The CRM tables have the following structure :

Column Name	Description
object_index	An integer index, used internally by CRM.
P001	The customer ID.
P002…	Columns containing customer data.

The CRM tables can be used for directly maintaining a link with an external database. For example, a batch process could insert or update data in the appropriate CRM table ready for calls to take place the next day; then at the end of the day, any amendments made to CRM records can be copied back to the master external database.
Note that you should think carefully whether the 'New Customer' and 'Modify Customer' buttons should be enabled in CRM if you are synchronising data in this way.

Adding data to the Synthesys database
Creating non-Synthesys tables is allowed, although care should be taken that there is a mechanism for removing old data from such tables when they are no longer required. We would not recommend modifying Synthesys tables directly unless you have had discussions with Synthesys software architects at Noetica, since it might jeapordise system stability.

ActiveX controls

This is a flexible way to improve data input to Synthesys as well as link to back end systems. ActiveX controls can query external databases, do on-line credit checking, can start CTI activity (such as passing a call to an IVR system, or start voice recording). They can also be used to start and control other applications.

New ActiveX controls are introduced to the system by using the 'Gallery', accessible from the Callflow Editor. Before adding an ActiveX to the system, it should be copied to the server Synthesys\bin directory; it will then be automatically installed on workstations and registered by the Synthesys Workstation software.

The gallery can then be used to inform Synthesys about properties of that control. Any properties that should be recorded in the database should be marked as [DB]. Additionally, for string properties, a length should be entered for the column size.

Although off-the-shelf controls can be used in this fashion, Synthesys provides two extra mechanisms that can be useful. One is a 'Maintenance' method; any control exposing a method called 'Maintenance' will have the 'Maintenance' menu item available in the Callflow Editor. Normally, property pages are used to control the form of a control, the maintenance dialog controls its data content.

The other useful feature is a number of ambient properties that the Callflow Editor and Manager make available as ActiveX containers.

DISPID_AMBIENT_USERMODE can be used to distinguish between Callflow Editor and Manager. Other stock ambient properties are available. The following Synthesys ambient properties are available for tighter integration with Synthesys…

Name	*DISPID*	*Description*
DISPID_AMBIENT_CAMPAIGN	5000	Campaign prefix, such as AAA01
DISPID_AMBIENT_SECTION	5001	Section Name.
DISPID_AMBIENT_CONTROL	5002	Control Name.
DISID_AMBIENT_ DATASOURCENAME	5003	ODBC data source name; Phoneyx in design mode, phoenix in release mode.
DISPID_AMBIENT_SEQUENCE_ID	5005	Sequence ID (Run Mode)
DISPID_AMBIENT_CUSTOMER_ID	5006	Current customer ID (Run Mode)
DISPID_AMBIENT_USERID	5007	ID of logged on user.

Probably the most useful are the sequence ID and the customer ID; they can be used to tie values stored externally to Synthesys with the Synthesys database.

Synthesys Spider

The Synthesys spider is an RPC mechanism which is often used by ActiveX controls. Let us say that we need to access a DB2 database held on an AS400 machine. Then we would write two applications.

The AS400Service would run on the server, and connect (perhaps using ODBC) to the database. It would then have provided a RPCs, called (say) AS400Service_Search. This would retrieve data from the database, and might also cache the results.

AS400ActiveX would present a user interface, and would communicate with the AS400Service_Search using the Spider RPC mechanism whenever it needs to retrieve data from the AS/400. In this way, the client doesn't need a direct link to the back end system, which simplifies client configuration.

Examples of controls

Some of these are relatively trivial, some have taken months of development work.

- As indicated above, we have a collection of controls that access data stored in an external database.
- A control for a Video Conferencing company, allowing studios in different parts of the world to be booked, and fees to be calculated.
- A control for a newspaper company, which would find the appropriate local newsagent for delivering the newspaper, using a variety of algorithms depending on circumstance.
- A control which can calculate appropriate sales discounts.
- A control which does 'auto correct' style spell checking, allowing replacements of common abbreviations to speed agent typing.
- A 'clipboard' control, which can be used to communicate data with external programs (for example CTI interfaces).
- A ticket control, which is used to allocate tickets for theatres and shows.