Supplied Demo Applications

Natural Business Services provides sample business services in the DEMO domain. All the Natural modules for these services are located in the SYSBIZDE library on the server. For more information about these services, see DEMO Domain.

Note:
If the sample services are not listed in the Business Service explorer, have your Natural Business Services administrator run the CSRLOAD program to load the repository.

In addition, client consumers (for example, Web services or client proxy classes) have been generated to demonstrate executing these services using Visual Studio and Java. Each business service was used to generate a Web service and then the Web services were used to generate the demo web application (web pages and menus).

This section describes the various business service types used in the demo application and highlights several features of the subprograms and proxies supplied in SYSBIZDE. The following topics are covered:

- Business Service Types
- Features of the Demo Application
- Additional Features When Using Predict

Business Service Types

Each business service has a type, which is determined by the wizard based on user input. On the client, the type is hidden from the user. On the server, it determines which Natural Construct model is used to generate the service. When the subprogram proxy is generated, the business service type is also stored in the business repository.

The business service types are:

Type	Service
blank	Arbsub (arbitrary subprogram)
1	Traditional
2	Object-Browse-Select without maintenance
3	Object-Browse-Select with maintenance
4	Object-Generic

A user may be able to identify which type a business service uses by its methods. The business service types are described in the following sections:

- Arbsub (Arbitrary Subprogram)

- Traditional
- Object-Browse-Select (Without Maintenance)
- Object-Browse-Select (With Maintenance)
- Object-Generic

Arbsub (Arbitrary Subprogram)

This business service (Type blank) assumes that the subprogram associated with it was not generated by Natural Construct. As Natural Business Services has no knowledge of the methods or required input and output parameters, all parameters are exposed to the client and the DEFAULT method is created. You can rename this method, as well as create other methods, by modifying the business repository information for this service.

An example of this type is the Calculator service, which has four methods: Add, Divide, Multiply and Subtract. As these methods would have behaved the same way as the DEFAULT method, overrides were added when the Web service was generated; the #FUNCTION parameter required a different value for each of the four new methods. For example, the Add #FUNCTION method required an override of “Add”.

A developer determines the value of #FUNCTION and decides what the subprogram will require to perform the function. For example, the Calculator service in the business repository uses the CALCY subprogram proxy in the SYSBIZDE library to call the CALC subprogram. If #FUNCTION is Add, CALC executes the appropriate code.

The main drawback to the Arbsub business service type is the amount of control given to the client. For example, if a Web service developer accidentally set the #FUNCTION override to Divide for the Add method, a user who only had permission to perform the Add method could inadvertently perform the Divide function.

Tip:
If this is a concern, use an Object-Generic business service type.

Traditional

This business service (Type 1) has two subprograms associated with it: an object maintenance subprogram (generated by the Object-Maint-Subp model) and an object browse subprogram (generated by the Object-Browse-Subp model).

The typical methods associated with this type are:

- Delete
- Exists
- Former
- Get
- Initialize

- Next
- Store
- Update
- Browse

An example of this type is the Customer business service, version 010101. The Browse method calls the ACUSTY proxy, which calls the ACUSTN browse subprogram. The other methods call the MCUSTY proxy, which calls the MCUSTN maintenance subprogram.

Object-Browse-Select (Without Maintenance)

This business service (Type 2) is created using the Object-Browse-Select-Subp model and behaves in the same manner as a Browse-Select subprogram. Internally, it is different since you only want to pass a limited number of rows across the wire at one time in a client/server environment.

The object browse select subprogram requires an object browse subprogram. Based on the keys for the object browse subprogram, the Object-Browse-Select-Subp model creates the FindBy… methods. The method names can be modified during generation and methods can be removed if they are not required. If the HISTOGRAM option is selected for a browse key in the object browse subprogram, the object browse select subprogram defines count methods for the key. The key values and a count are provided, instead of all the data for the record (using FindBy… methods).

An example of the Object-Browse-Select business service type is Order, version 020101. The FindByOrderWarehouseId method returns all data in Warehouse ID order. A related method, OrderWarehouseIdCount, returns only the specified warehouse ID and a count of the number of orders for that warehouse. This service uses the BORDY subprogram proxy, which accesses the BORDN object subprogram.

Object-Browse-Select (With Maintenance)

This business service (Type 3) is also created using the Object-Browse-Select-Subp model and takes advantage of more features of the model. The CustomerWithContactData, version 020101, business service calls both the object browse and object maintenance subprograms from the same object browse select subprogram. To allow this functionality, an object maintenance subprogram was added to the Object-Browse-Select-Subp model specifications. This created four new methods: MultiMaint, Update, Delete and Store, which can be applied to the entire business service (i.e., a group of rows).

The action applied to each row is indicated by the row state, for example: U for Update, D for Delete, and A for Add. Each row sent to the server for maintenance requires a row state. For the server to apply these actions, the business method must be MultiMaint, Update, Delete or Store.

In general, the client never needs to use the Update, Delete or Store methods because the MultiMaint method handles all of them. The methods are supplied for security purposes. For example, if an administrator applied security at the method level to revoke Delete privileges for a user, the business service will not allow the user to use a D (for Delete) row state.

The CustomerWithContactData business service, version 020101, uses the BCUST2Y subprogram proxy, which calls the BCUST2N object subprogram. The Object-Browse-Select service uses the ACUST2N object browse subprogram and the MCUST2N object maintenance subprogram.

Object-Generic

This business service (Type 4) allows the business service to access up to 10 subprograms and create up to 20 different methods. Each method can be clearly defined on the server by hard-coding certain values. This prevents security from being breached and allows one client interface to be used for multiple subprograms.

An example of this type is the CalculatorAdvance business service. This service invokes the BNUMY proxy, which calls the BNUM subprogram. BNUM accesses two subprograms: CALC and GCDN. As the data for these subprograms is similar, the exposed variables have been reduced (this is why some of the parameters are commented out in the generated PARAMETER-DATA user exit code). Before either subprogram is called, data must be moved from the exposed data area to the local data areas used to pass data into these subprograms. This is done in the generated MOVE-TO user exit code. Similarly, data must be moved back to the exposed data area before control is returned to the client. This is done in the generated MOVE-BACK user exit code.

While defining methods for the CalculatorAdvance service, the developer specified which subprograms to execute for each method, the execution order of these subprograms, and whether code should be executed before and/or after the subprogram. For example, the SolutionWithLowerNumbers method has code that is executed after the GCDN subprogram is executed. This method also executes the CALC subprogram. (Refer to the AFTER-CODE subroutine in the BNUM subprogram.) This code reduces the first and second number based on the greatest common denominator and then calculates the division between the two numbers.

The Subtract method executes code before the CALC subprogram is executed. Refer to the BEFORE-CODE subroutine to see how #FUNCTION is assigned. This process is similar to providing overrides for the Calculator service, except the Web service cannot change #FUNCTION for the Subtract method because it will always be overridden in the BNUM subprogram.

Features of the Demo Application

This section describes the features of the demo application, SYSBIZDE. The following topics are covered:

- Subprogram Proxies
- Web and Business Services
- SOAP Client

Subprogram Proxies

As data typically comes from a different platform and uses a different character set, each business service requires a subprogram proxy to retrieve data off the wire. Although the proxy is required, all business code is contained in the subprogram that is called by the proxy.

During generation of the subprogram proxy, the business repository is populated with the methods used by the service, as well as the domain/service/version specifications.

Web and Business Services

The SYSBIZDE demo application includes the following Web and business services:

Web Service Name (or business service name when different)	Type	Proxy	Subprogram
Calculator	Arbsub	CALCY	CALC
CalculatorAdvance	Object-Generic	BNUMY	BNUM CALC GCDN
CustomerCreditAnalysis	Traditional	MCUST3Y	MCUST3N
CustomerPlain Customer, version 010101	Traditional With browse	MCUSTY ACUSTY	MCUSTN ACUSTN
CustomerWithContactData	Object-Browse-Select With browse With maintenance	BCUST2Y	BCUST2N ACUST2N MCUST2N
CustomerWithContactDataTraditional CustomerWithContactData, version 010101	Traditional With browse	MCUST2Y ACUST2Y	MCUST2N ACUST2N
ErrorMessageTesting	Basic Arbsub	FLIPSTY	FLIPSTR
FlipString	Arbsub with defined method	FLIPSTY	FLIPSTR
GreatestCommonDenominator (also used with CalculatorAdvance)	Arbsub	GCDY	GCDN
Order	Object-Browse-Select With browse	BORDY	BORDN AORDN
OrderTraditional Order, version 010101	Traditional With browse	MORDY AORDY	MORDN AORDN
Product	Traditional With browse	MPRODY APRODY	MPRODN APRODN
StringManipulation	Object-Generic	BSTRINGY	BSTRINGN FLIBSTR CSUCASE

Web Service Name (or business service name when different)	Type	Proxy	Subprogram
Warehouse	Traditional	MWHY	MWHN
	With browse	AWHY	AWHN

SOAP Client

The easiest way to test a Web service is to create a SOAP message that uses the service. To do this, open the context menu for the Web service and select the SOAP Client testing tool. The SOAP Client provides a list of the available methods. After you select a method, the SOAP Client fills in the URL, method name, and a simple SOAP Request. If you require specific input values, you can modify the SOAP message to reflect this.

In some cases, you can use one method to create output for another method to use as input. For example, the output for a FindBy… method is typically the input for a MultiMaint method. In this case, you can indicate the rows to be modified by typing a value in the appropriate row states (for example, U for Update).

Natural Business Services uses services exposed by the SOAP Client. For example:

- GetMetaData, which retrieves the metadata for a service
- GetServiceList, which determines which services are used in a Web service node
- Login, which creates a security token for a user ID and password combination

To view an example of input that has been modified, open the Soap Client example called CalculatorAdvanceSolutionWithLowerNumbers (located in the SoapClient directory under inetpubs/wwwroot/NBSDemos). CalculatorAdvanceSolutionWithLowerNumbers divides the first and second numbers by their greatest common divisor. This produces the lowest numbers that equal the same result when they are divided. The greatest common divisor is also shown. For example, 30 divided by 24=1.25. The greatest common divisor is 6, so 30 can be reduced to 5 and 24 can be reduced to 4 and the result is 1.25.

Additional Features When Using Predict

This section describes additional features of the SYSBIZDE demo application when using the Predict data dictionary.

BDT Options

When data is displayed, formatting should be automatically performed. To do this, you can define BDTs (business data types) for fields in the Predict file definition. If a BDT is not assigned in Predict, the Web Service wizard “guesses” which BDT to use. For example, if the field format is numeric with two decimal places, the wizard guesses that the BDT type is a currency amount. An example of using this BDT is the Result field for the Calculator service in the DEMO domain. A “$” is automatically placed at the beginning of this field.

With this rule you may think the FIRST-NUM and SECOND-NUM fields should also have “$” in front of them. By default they do, but the data type was changed from BDTCurrency to BDTNumeric when the Web service was generated.

Tip:
To remove the “$” from the Result field, you can assign BDTNumeric to this field and regenerate the Web service.

ALLOW-LOWER-CASE Option

By default, an object browse subprogram converts all input data into upper case. The ALLOW-LOWER-CASE option allows users to enter data in lower case. It is useful for a field like Business Name, where the name is stored in mixed case.

To specify the lower case option:

1. Associate the ALLOW-LOWER-CASE keyword with the definition for the input field in Predict.

2. Regenerate the object browse subprogram.

For more information, see Natural Construct Object Models.