

Natural Business Services

Getting Started with Natural Business Services

Version 8.2.2

November 2023

This document applies to Natural Business Services Version 8.2.2 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2006-2023 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Document ID: NBS-GETTINGSTARTED-822-20231119

Table of Contents

Preface	v
1 About this Documentation	1
Document Conventions	2
Online Information and Support	2
Data Protection	3
2 Introduction	5
Create a Business Service	6
Create a Web Service for Your Business Service	6
3 Overview of Natural Business Services	7
4 Architecture of Natural Business Services	9
Server Components	11
Client Components	13
5 Roles and Prerequisites	15
General Information	16
Prerequisites	16
Types of Security Available with EntireX RPC	17
6 Supplied Demo Applications	19
Business Service Types	20
Features of the Demo Application	23
Additional Feature When Using Predict	24
7 Using Natural Business Services	27
Administer Business Services	28
Configure the Plug-in	28
Create a New Business Service	28

Preface

This documentation provides an overview of Natural Business Services and the architecture of the product and its components. It is intended for developers and others who are new to Natural Business Services and want to learn how to create and maintain business services.

This documentation is organized under the following headings:

Introduction	Introduces Natural Business Services.
Overview of Natural Business Services	Provides an overview of Natural Business Services.
Architecture of Natural Business Services	Describes the architecture of Natural Business Services and its components.
Roles and Prerequisites	Describes the roles of different users and helps new users understand which prerequisites are required for each business service activity.
Supplied Demo Applications	Describes the demo applications supplied with Natural Business Services.
Using Natural Business Services	Describes how to create a business service.



Note: For more information on Natural Business Services, see *Understanding Natural Business Services*.

1

About this Documentation

■ Document Conventions	2
■ Online Information and Support	2
■ Data Protection	3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Monospace font	Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.
<i>Italic</i>	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
Monospace font	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the symbol.
[]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at <https://documentation.softwareag.com>.

In addition, you can also access the cloud product documentation via <https://www.software-ag.cloud>. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

You can find helpful product training material on our Learning Portal at <https://knowledge.softwareag.com>.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at <https://tech-community.softwareag.com>. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at <https://github.com/softwareag> and <https://hub.docker.com/publishers/softwareag> and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at <https://empower.softwareag.com>. Many services on this portal require that you have an account. If you do not yet have one, you can request it at <https://empower.softwareag.com/register>. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Introduction

■ Create a Business Service	6
■ Create a Web Service for Your Business Service	6

Create a Business Service

Natural Business Services allows you to create and maintain business services. This is done using the client, namely NaturalONE's Service Development plug-in, which provides the Business Service wizard.

Each business service combines a group of methods related to a common business entity, such as a customer or order. Processing for the methods is supplied by either existing or wizard-generated Natural subprograms. The definition for each service is stored in the business service repository and identifies the associated methods.

During generation, the Business Service wizard:

- Populates the business repository with information about the methods used by the service
- Provides the domain, service, and version specifications
- Adds the method and service descriptions to the workspace

For information on how Natural Business Services creates business services, see *Understanding Natural Business Services*.

Create a Web Service for Your Business Service

Web services are created using NaturalONE's Service Development plug-in and the WS-Stack plug-in. A wizard is provided which creates Java classes and Web service definition files which can easily be deployed to a Web server.

3

Overview of Natural Business Services

A Natural business service provides a business perspective of a grouping of Natural subprograms. This perspective includes such things as the:

- Business service description
- Service methods and method descriptions

Retrieval algorithms are available to allow quick and easy searches of the business service repository. This allows a business analyst to quickly determine if a particular service currently exists or if one must be created. In addition, you can use NaturalONE to deploy business services from one environment to another and/or apply security at a domain, business service, and/or method level.



Note: The Natural subprograms used for your business services can be located in any library. The demo business services are located in the SYSBIZDE library on the server.

Each method defined for a business service is typically associated with a different Natural subprogram. If a method does not have an associated subprogram, a default subprogram for the entire service is required.

The attributes for each method are determined by the subprogram invoked for that method. These attributes can be used to describe the business service or as input, output, input/output, or state values for the service. They can be acted upon and/or changed based on which methods have been defined for a service. For example, the Customer service in the demo application contains the Update method and the Name and Phone Number attributes, which allows the service to be used to change the phone number for a customer.

You can quickly develop business services using the Business Service wizard supplied with NaturalONE. This wizard recognizes the style of service you require based on your answers to simple questions asked on the wizard panels.

Although you can create a business service for any Natural subprogram that does not contain user interface code (i.e., a Natural subprogram that was not generated using Natural Construct and does not contain user interface code), this type of subprogram would only have one method associated with it — `DEFAULT`, which executes the associated subprogram as it normally would in Natural. An example of this type of business service is the `ErrorMessageTesting` service in the `DEMO` domain.

For more functionality, you can wrap several subprograms into one subprogram and create additional methods. An example of this type of service is the `CalculatorAdvance` service in the `DEMO` domain. When this service was created, the Business Service wizard recognized that the developer wanted to use more than one subprogram. The wizard prompted the developer to define the additional methods based on which subprograms to call and in what order to call them. The methods were enhanced by adding user exit code between the calls to these subprograms.

While this functionality is possible without using Natural Construct and Predict, you can enhance the functionality by generating business services based on files that have been defined in Predict. Standard methods, such as `Delete`, `Next`, `Update`, `Browse`, and `FindBy*`, can be automatically created based on input to the Business Service wizard and the file definitions set up in Predict.

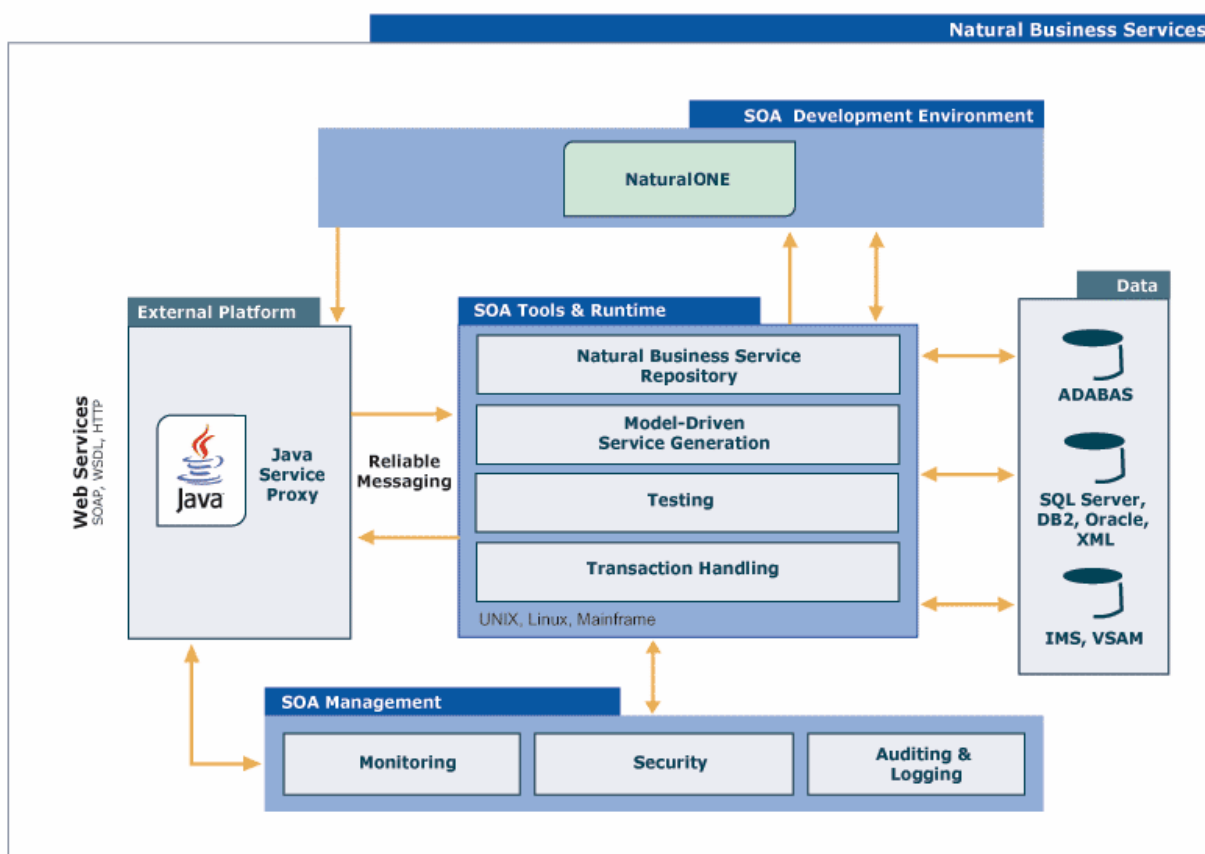


Note: When using Predict, please ensure that the Predict file descriptions and the generated Natural DDMs are connected. An existing connection ensures a proper representation of all supported database features in the generated Natural DDM.

4 **Architecture of Natural Business Services**

■ Server Components	11
■ Client Components	13

Using Natural Business Services, you can create all the components of a business service, including Natural object subprograms that perform maintenance and browse functions and GUI dialogs or web pages that communicate with the object subprograms. Communication between server and client components of an application is performed by a combination of EntireX and Natural RPC (or EntireX configured to use TCP/IP), as well as Natural Business Services middleware components. The middleware components encapsulate calls to EntireX on the client and server. The following diagram shows the architecture of character-based Natural applications and business service components:



This section describes these components according to the platforms on which the components run.

Server Components

This section describes the server components for Natural Business Services. The following topics are covered:

- [Required for Development](#)
- [Required at Runtime](#)

Required for Development

The following table lists the components required for development purposes:

Component	Description
Natural subprograms	<p>Subprograms written in Natural that do not contain user interface code (for example, WRITE, DISPLAY, PRINT, INPUT, and REINPUT statements) or navigation code (for example, PF-key processing). They can be existing Natural subprograms or they can be wizard-generated in NaturalONE. Existing subprograms can be wrapped together so one server subprogram accesses more than one subprogram. The Business Service wizard can wrap the subprograms it generates, as well as use Natural Construct models internally to generate subprograms that perform maintenance and browse functions on the server. The wizard chooses the appropriate model based on criteria the user has selected. These models are: Object-Browse-Subp, Object-Maint-Subp, Object-Browse-Select-Subp, and Object-Generic-Subp.</p> <p>Note:</p> <ol style="list-style-type: none"> 1. The Object-Browse-Select-Subp model has the same type of functionality as the Browse-Select model, but is designed for a client/server environment where only n rows are processed at a time. 2. The Object-Generic-Subp generates a business service that uses more than one pre-existing subprogram. <p>The same set of business objects can be accessed from character-based Natural applications, client/server applications, and web applications. This ensures that the integrity of business data is preserved, independent of the presentation layer, and existing code can be preserved.</p>
Character user interface (optional; only used at sites that access the business service from a 3270 client)	Non-distributed Natural applications created with Natural Construct accessing subprograms directly (for example, subprograms generated by the Object-Maint-Dialog model).

Required at Runtime

The following table lists the components required for runtime purposes:

Component	Description
Business Service Administration subsystem	<p>Server subsystem that allows system administrators, application administrators, and developers to set up and manage system and application environments.</p> <p>Note: Although this subsystem also provides access to the Business Service repository, we recommend that you use NaturalONE to access the repository. Only system environment options will be discussed from the server perspective.</p>
EntireX	<p>Runtime component that transfers messages between Windows or the web server and the Natural environment. EntireX can be configured to use either native TCP/IP or Entire Net-Work as the transport layer.</p> <p>EntireX performs the following runtime functions:</p> <ul style="list-style-type: none">■ Encrypt and decrypt data (set in the Broker attribute file)■ Compress and decompress data (set in the Broker attribute file)■ Translate data (handled automatically and has defaults you can customize) <p>Note: As this component is separate, it can be used without Natural Business Services and may already be installed. It is a required component for the Natural Business Services system environment.</p>
Natural RPC server	<p>Server that provides a common interface and EntireX services for Natural subprograms in the application. The main functions of the Natural RPC server are to:</p> <ul style="list-style-type: none">■ Receive requests from the client through EntireX■ Optionally decompress and/or decrypt and translate the request message from the client's character set (ASCII) to the server's character set (either ASCII or EBCDIC)■ Check security to ensure that the client is permitted to issue the request■ Optionally compress and/or encrypt the message to be returned
Business Service repository	<p>Directory structure containing the business service metadata, such as domains, descriptions, methods, method descriptions, as well as security access to these services and methods.</p>

Client Components

You can create business services with NaturalONE's Service Development plug-in. The plug-in links to Natural Business Services from NaturalONE. Using the plug-in, you can configure the business service connections, search for business services, and invoke the wizards.

Component	Description
Business Services menu	Menu used to perform tasks such as the following: <ul style="list-style-type: none"> ■ Create new business services, domains and steplib ■ Add metadata to Centrasite
Natural Server view	Tree view of the business repository. You can perform tasks such as the following: <ul style="list-style-type: none"> ■ View current business services, domains and steplib ■ Add (download) definitions to an existing NaturalONE project in the workspace
Preferences	You can define preferences such as: <ul style="list-style-type: none"> ■ Define generation, upload and download settings ■ Set up Centrasite connections ■ Define UI interactions depending on product installations
Business Service wizard	Creates new business service definitions in the local workspace.
Domain and Steplib wizards	Creates domain and steplib definitions in the local workspace.
Java wizard	Creates Java classes to access your business service from the client.
Web Service wizard	Creates Java classes and Web services definitions.
Security view	Used to inspect and update NBS security definitions.

5

Roles and Prerequisites

■ General Information	16
■ Prerequisites	16
■ Types of Security Available with EntireX RPC	17

This section describes the roles of different users and helps new users understand which prerequisites are required for each business service activity.

General Information

With all the functionality and components of Natural Business Services, it is important to understand which components are required for your scenario. Every development activity requires the Business Service repository and the Administration subsystem and every consumer requires an existing business service. All business services are created using NaturalONE's Service Development plug-in.

At a high level, the minimum tasks required to produce a viable business service are:

1. Install Natural Business Services on the server.
2. Install NaturalONE - including EntireX and the Service Development plug-in - on the client.
3. Ensure connectivity exists through EntireX and Natural RPC.
4. Ensure security is set for users and the development environments.
5. Create the business service.
6. Using NaturalONE, consume the business service through the creation of Java and Web service clients.
7. Invoke the business service through one of the consumers.

Prerequisites

The following table describes the prerequisites required for each business service activity.

Activity	Natural Business Services	NaturalONE	EntireX	Java Runtime
Create a business service with NaturalONE		X	X	X
Create a Java application or Web service	X	X	X	
Execute a Java application	X		X	X
Execute a Java Web service	X		X	X

Types of Security Available with EntireX RPC

The following types of security are available when using the EntireX communication middleware:

Security Type	Description
NONE	Uses no security.
NSC	Uses Natural Security to validate users and check library access. (Natural Business Services definitions are used to check authorizations.)

As most sites have both a production and development environment, the decision about which type of security to use can be postponed for a while (although you must eventually choose a security type). See also *Defining Users and Security Groups* in *Natural Business Services Administration*.

For more information on using Natural RPC with Natural Security, see *Natural RPC (Remote Procedure Call)* in the Natural documentation for the appropriate platform.

6

Supplied Demo Applications

■ Business Service Types	20
■ Features of the Demo Application	23
■ Additional Feature When Using Predict	24

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* in *Natural Business Services Administration*.



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

This section describes the various business service types used in the demo application and highlights several features of the subprograms supplied in SYSBIZDE.

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.

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 CALC subprogram. If #FUNCTION is set to "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 ACUSTN browse subprogram. The other methods call 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 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 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 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

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

Web Service Name (or business service name when different)	Type	Subprogram
Calculator	Arbsub	CALC
CalculatorAdvance	Object-Generic	BNUM CALC GCDN
CustomerCreditAnalysis	Traditional	MCUST3N
CustomerPlain	Traditional	MCUSTN
Customer, version 010101	With browse	ACUSTN

Web Service Name (or business service name when different)	Type	Subprogram
CustomerWithContactData	Object-Browse-Select	BCUST2N
	With browse	ACUST2N
	With maintenance	MCUST2N
CustomerWithContactDataTraditional	Traditional	MCUST2N
CustomerWithContactData, version 010101	With browse	ACUST2N
ErrorMessageTesting	Basic Arbsub	FLIPSTR
FlipString	Arbsub with defined method	FLIPSTR
GreatestCommonDenominator (also used with CalculatorAdvance)	Arbsub	GCDN
Order	Object-Browse-Select	BORDN
	With browse	AORDN
OrderTraditional	Traditional	MORDN
Order, version 010101	With browse	AORDN
Product	Traditional	MPRODN
	With browse	APRODN
StringManipulation	Object-Generic	BSTRINGN
		FLIBSTR
		CSUCASE
Warehouse	Traditional	MWHN
	With browse	AWHN

Additional Feature When Using Predict

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

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*.

7

Using Natural Business Services

■ Administer Business Services	28
■ Configure the Plug-in	28
■ Create a New Business Service	28

Administer Business Services

Administrators can use NaturalONE's Service Development plug-in to perform various administrative and security-related tasks. These include:

- Edit service definitions locally and upload them to the server.
- Define or edit domain and steplib definitions and upload them to the server.
- View security settings per domain, make changes and synchronize changes to the server.

Configure the Plug-in

Before you can create a business service, you must confirm that the environments are set up correctly. To do this:

- Define Natural Development Server (NDV) connections to a Natural environment where Natural Business Services (NBS) is installed.
- Browse the environment and verify that NBS nodes are available in the **Natural Server** view.
- Define RPC environments using EntireX preferences in NaturalONE. The RPC environment must match a Natural RPC server that is installed in your NBS server environment
- Configure a Web server which should have the WS-Stack installed (see WS-Stack documentation). Alternately, use the internal WS-Stack Web server installed with NaturalONE.

Create a New Business Service

A business service consists of a collection of methods related to a common business entity. The best way to create a business service is to use the Business Service wizard in Natural ONE. You should follow these steps:

- Ensure that you have a domain and steplib created and properly configured.
- The main library you are using should already be available in the local NaturalONE project and workspace. Your NBS steplib definition should contain this library.
- Make sure you have some subprograms created that you want to service-enable. As a minimum, these subprograms should have defined data area definitions.
- Start the Business Service wizard to define a service. Give the service a name and select a domain.
- Once created, use the Business Service editor to define different methods. You should indicate which subprograms are used for each method by either using the default subprogram for the entire service or defining each method with different subprograms.

- Upload the service definition (and related subprograms if you have not done so already) to the server.
- The service is now ready to be used.

