NaturalONE

Using NaturalONE

Version 9.1.3

October 2021

ADABAS & NATURAL

This document applies to NaturalONE Version 9.1.3 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 © 2009-2021 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: ONE-USING-913-20210930

Table of Contents

Preface xi
1 About this Documentation 1
Document Conventions 2
Online Information and Support 2
Data Protection 3
I 5
2 Starting NaturalONE 7
Starting the Software AG Designer 8
NaturalONE Perspective 9
License Key 9
Restore of the Natural Server View 9
Natural for Eclipse Projects 10
3 The NaturalONE Perspective 11
Opening the NaturalONE Perspective 12
Views of the NaturalONE Perspective 13
Editor Area in the NaturalONE Perspective 15
Showing a View of the NaturalONE Perspective 15
Resetting the NaturalONE Perspective 16
II Working with Natural Projects in Local Mode 17
4 Viewing the Contents of Your Natural Projects 19
General Information 20
Using the Natural Navigator View 21
Customizing the Natural Navigator View 22
Filtering Natural Objects and Libraries 24
Using the Details View 32
5 Managing Natural Projects 35
Creating Natural Projects 36
Types of Natural Projects 42
Changing the Project Properties 43
Quickly Viewing the Properties 64
Enabling a Project for NaturalONE 65
6 Managing Libraries in a Natural Project 67
Libraries in a Natural Project 68
Creating Libraries 75
Changing the Library Properties 78
Updating the Library Properties with the Settings from the Server 83
Copying, Pasting, Moving, Renaming and Deleting Libraries 83
7 Managing Objects in a Natural Project 85
Creating Natural Objects 86
Changing the Object Properties 89
Editing Objects 91
Saving Objects 91
Printing Objects 92

Executing Objects ... 93
Copying and Pasting Objects and Libraries ... 94
Moving Objects and Libraries ... 97
Renaming Objects and Libraries ... 98
Deleting Objects and Libraries ... 102
8 Launching Natural Applications ... 103
Defining a Different Start Library ... 106
Specifying Application Parameters ... 107
Launching Applications Using Shortcut Keys ... 107
9 Modifying Objects in the Natural Environment or in the Repository ... 109
General Information ... 110
Updating the Objects in the Natural Environment ... 110
Build Sequence ... 114
Canceling a Build ... 115
Flags in a Label Decoration ... 115
Resetting Flags ... 116
Consolidating Objects in Private-Mode Libraries ... 117
Rebuilding all Objects in the Natural Environment ... 118
Using the Compare Editor ... 119
Checking the Time Stamps in the Natural Environment ... 122
Excluding Objects from Processing in the Natural Environment ... 128
Committing the Objects to the Repository of the Version Control System ... 133
10 Understanding the Behavior of the Natural Builder ... 135
III Working with Natural Objects in Natural Server Mode ... 139
11 Accessing a Remote Development Environment ... 141
Mapping a Natural Environment ... 142
Dynamically Changing the CICS Transaction Name when Starting a Session ... 145
Contents of the Natural Server View ... 145
Filtering Libraries and Objects ... 147
Properties for the Different Nodes ... 153
System Information for a Natural Environment ... 153
Unmapping a Natural Environment ... 158
12 Managing Objects Directly on a Natural Server ... 159
Editing Objects ... 160
Listing Objects ... 161
Saving Objects ... 161
Checking Objects ... 162
Stowing Objects ... 162
Cataloging Objects ... 163
Executing Objects ... 163
Refreshing the Display ... 164
Copying and Moving Objects and Libraries ... 164
Renaming Objects and Libraries ... 167

Deleting Objects and Libraries ... 168
Unlocking Locked Objects ... 168
Working with DDMs ... 171
Working with Dialogs ... 172
Working with Resources ... 173
Working with Private-mode Libraries ... 173
13 Launching Natural Applications ... 175
14 Using the Natural Command Console for Mainframes ... 177
General Information ... 178
Opening the Natural Command Console for Mainframes ... 179
Active Environment ... 179
Server Connections ... 180
Commands in the Natural Command Console for Mainframes ... 180
IV Using the Natural Editors ... 183
15 General Information ... 185
Types of Natural Editors ... 186
Invoking a Natural Editor ... 187
Editing Data Areas ... 187
Viewing Dialogs, Classes and Adapters ... 188
Problems in Your Natural Sources ... 188
Parsing Dependent Objects ... 190
Unicode and Code Page Support ... 191
Bidirectional Language Support ... 192
Source Header ... 196
Line Numbers ... 196
16 Using the Source Editor ... 199
About the Source Editor ... 200
Associated Views ... 200
Using Content Assist ... 203
Using Context-Sensitive Help ... 204
Using Code Snippets from the Natural Community ... 205
Working with Tasks ... 209
Working with Bookmarks ... 212
Error Handling in the Source Editor ... 213
Going to a Specific Natural Line Number ... 213
Opening Referenced Objects and Jumping to Variable and Internal Subroutine Definitions ... 214
Externalizing Code Fragments ... 216
Inserting Call or Include Statements ... 219
Importing Data Fields ... 221
Generating Counter Fields ... 224
Adding and Removing Comments ... 226
Toggle Comments ... 226
Protecting Source Code Lines ... 227
Protected Lines in Sources Generated by Construct or Code Generation ... 228

Translating to Upper Case or Lower Case ... 228
Indenting the Source Code Lines ... 229
17 Using the Map Editor ... 231
About the Map Editor ... 232
Associated Views ... 233
Adding Controls to the Map ... 236
Selecting Controls in the Map ... 238
Changing the Reference Control for Selection ... 239
Managing the Controls in the Map ... 240
Defining Rules ... 241
Moving Data Definitions to Data Fields ... 243
Editing the Code of a Map ... 243
Changing the Properties for the Map ... 244
Changing the Properties for a Text Constant ... 246
Changing the Properties for a Data Field ... 247
Changing the Properties for a Rule ... 253
Defining Arrays ... 254
Viewing the Status Properties ... 256
Saving Maps ... 257
Stowing Maps ... 258
Validating Maps ... 258
Error Handling in the Map Editor ... 259
18 Using the DDM Editor ... 261
Creating a DDM ... 262
About the DDM Editor ... 265
Associated Views ... 266
Managing the Fields of the DDM ... 268
Field Attributes ... 269
V Using the Debugger ... 271
19 Debugging Natural Applications ... 273
General Information ... 274
Using Symbol Tables ... 274
Starting the Debugger ... 275
Commands in the Debug Perspective ... 276
Using the Debug Perspective ... 276
Specifying the Breakpoint and Watchpoint Properties ... 283
Going to the Next Statement ... 287
20 Using a Debug Attach Server ... 289
General Information ... 290
Starting the Debug Attach Server ... 292
Debugging a Natural RPC Application ... 292
Debugging an External Natural Application ... 293
VI Creating Application-Specific Messages ... 295
21 Creating Application-Specific Messages ... 297
General Information ... 298

Type, Name and Location of the Message Files ... 298
Creating Message Files ... 299
Opening an Existing Message File ... 301
About the Error Message Editor ... 301
Associated Views ... 303
Translating a Message File ... 306
Layout of a Message File ... 306
VII Using the Data Browser ... 309
22 Using the Data Browser ... 311
General Information ... 312
Creating a Report Template ... 312
About the Data Browser Editor ... 314
Selecting the Fields for the Report ... 314
Displaying the Properties for a Field ... 316
Setting Options for the Report ... 318
Creating the Report ... 321
Saving a Report Template ... 323
Saving a Report ... 323
Displaying the Properties for a Report ... 324
Opening an Existing Report Template ... 325
VIII Generating API Documentation with NATdoc ... 327
23 Generating API Documentation with NATdoc ... 329
Quick Start ... 330
Generating NATdoc ... 331
Location of the NATdoc Files ... 334
Previewing the API Documentation in the NATdoc View ... 335
Documentation Comments in the Source Code ... 336
Special Comment Files ... 338
Overview of NATdoc Tags ... 339
Where Can Tags be Used? ... 342
Using Custom Templates ... 343
Detailed Template Descriptions ... 345
IX Checking Natural Code with NATstyle ... 361
24 Checking Natural Code with NATstyle ... 363
General Information ... 364
Checking the Natural Code ... 364
Clearing the NATstyle Violations ... 366
Working with Result Files ... 366
Invoking NATstyle from Outside Eclipse ... 367
Overview of NATstyle Rules, Error Messages and Solutions ... 369
DTDs Used by NATstyle ... 378
X Using the Natural Profiler ... 383
25 Using the Natural Profiler ... 385
General Information ... 386
Prerequisites ... 386

Starting a Profiler Session ... 387
Viewing the Profiler Output ... 388
Managing the Profiler Sessions ... 392
Displaying Natural Profiler Resource Data ... 396
Application Programming Interface ... 398
Overview of Event Types ... 398
XI ... 401
26 Using the Natural Code Coverage ... 403
General Information ... 404
Quick Start ... 404
Prerequisites ... 405
Producing Natural Code Coverage Data ... 406
Viewing Natural Code Coverage Data ... 406
27 Using the Natural Coverage Plugin for Jenkins ... 409
Principle of Operation ... 410
Prerequisites ... 410
Installation ... 410
Configuration ... 411
Using the Plug-In ... 411
XII Using Natural Tools and Utilities ... 413
28 Using Natural Tools and Utilities ... 415
General Information ... 416
Starting Natural Tools and Utilities ... 417
SYSUTIL Utility ... 417
Rich GUI interface of SYSEXT ... 420
Rich GUI interface of SYSEXV ... 423
Message Retrieval ... 425
Extend the List of Tools and Utilities ... 427
Return Control to Natural Tools and Utilities ... 431
XIII Using the LastMsg View ... 433
29 Using the LastMsg View ... 435
XIV Using the XML Toolkit ... 439
30 Using the XML Toolkit ... 441
General Information ... 442
Generating the Output Files ... 443
Displaying the Settings of the Last Generation ... 446
XV Use SSL/TLS ... 447
31 Use SSL/TLS ... 449
XVI Deploying Applications ... 451
32 Deploying Natural Applications ... 453
General Information ... 454
Using the Deployment Wizard for Natural Applications ... 454
Controlling the Scope of Files to be Processed ... 468
Starting the Deployment from Eclipse ... 469
Starting the Deployment from the Command Line ... 470

Status Code Handling ... 471
Checking the Time Stamps in the Natural Environment ... 472
Reducing the Amount of Logging Information ... 475
Project References Handling ... 475
Versioning Repository Handling ... 476
33 Deploying Natural Applications with Jenkins ... 479
General Information ... 480
Basic Assumptions ... 480
Generating and Verifying Deployment Scripts with NaturalONE ... 480
JAR Files for Deployment ... 481
Versioning Repository Handling ... 481
Files Generated by Natural Deployment Scripts ... 481
Jenkins Jobs ... 482
34 Deploying Java Applications ... 491
General Information ... 492
Using the Deployment Wizard for Java Applications ... 492
Starting the Deployment from Eclipse ... 496
Starting the Deployment from the Command Line ... 496
Status Code Handling ... 498
35 Using a Master Deployment ... 499
General Information ... 500
Using the Deployment Wizard for a Master Deployment ... 500
Starting the Deployment from Eclipse ... 505
Starting the Deployment from the Command Line ... 505
XVII Setting the Preferences ... 509
36 Setting the Preferences ... 511
Showing the Natural-Specific Preferences ... 512
Natural ... 513
Appearance ... 528
Label Decorations ... 528
Data Browser ... 532
Debug ... 532
Debug Attach Settings ... 533
Display Options ... 533
Editors ... 534
DDM Editor ... 534
Map Editor ... 535
Object Templates ... 539
Source Editor ... 540
NATstyle ... 548
Natural Navigator ... 550
Parser ... 551
Profiler ... 555
Regional Settings ... 559
Runtime Execution ... 562

File Transfer 565
Natural I/O 566
SSL/TLS 568
Tomcat 570
XML Toolkit 570

Preface

This documentation explains how to work with NaturalONE, which is the Eclipse-based development environment for Natural. It explains the basic functionality for Natural application development. Special topics such as the creation of rich internet applications or web services are explained in other parts of the NaturalONE documentation.

This documentation is organized under the following headings:

Starting NaturalONE	How to start NaturalONE. Some preparatory steps which are required before you can start working successfully with NaturalONE.
The NaturalONE Perspective	How to open the NaturalONE perspective. Brief information on the contents of this perspective.
Working with Natural Projects in Local Mode	How to work with Natural projects, libraries and objects in the Eclipse workspace. How to update objects in the Natural environment (for example, on a Natural server in a mainframe environment).
Working with Natural Objects in Natural Server Mode	How to make a connection to a Natural server. How to edit and manage objects directly on a Natural server.
Using the Natural Editors	How to use the source editor, map editor and DDM editor. General information, for example, on how to edit data areas, on Unicode and bidirectional language support.
Using the Debugger	How to debug Natural applications. How to debug Natural RPC applications and external Natural applications using a debug attach server.
Using the Data Browser	How to generate data reports from Adabas or SQL databases which are available in your Natural server environment.
Using the XML Toolkit	How to generate aids for the processing of XML documents within Natural.
Creating Application-Specific Messages	How to write your own application-specific messages.
Generating API Documentation with NATdoc	How to generate API documentation in HTML format from doc comments in the source code.
Checking Natural Code with NATstyle	How to make sure that your Natural code adheres to your coding standards.
Using the Natural Profiler	How to collect trace data for selected events that are performed within Natural applications.
Using the Natural Code Coverage	How to enable and use the Natural Code Coverage to monitor the executed statements of a Natural application.
Using the Natural Coverage Plugin for Jenkins	How to use the Natural Coverage plugin to read *.ncvf* files and to visualize the information stored within.

Using Natural Tools and Utilities	How to start selected Natural tools and utilities from NaturalONE.
Using the LastMsg View	How to use the LastMsg View.
Using SSL/TLS	How to establish SSL/TLS connections with and without authentication.
Deploying Applications	How to deploy NaturalONE applications using the different deployment wizards.
Setting the Preferences	Information on the Natural-specific preferences.

Note: The descriptions and screenshots in this documentation are based on the Windows version of Eclipse, but they also apply to the Linux version of Eclipse.

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 *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	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

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at **https://documentation.softwareag.com**.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at **https://empower.softwareag.com/**.

You can find product information on the Software AG Empower Product Support website at **https://empower.softwareag.com**.

To submit feature/enhancement requests, get information about product availability, and download products, go to **Products**.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the **Knowledge Center**.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at **https://empower.softwareag.com/public_directory.aspx** and give us a call.

Software AG Tech Community

You can find documentation and other technical information on the Software AG Tech Community website at **https://techcommunity.softwareag.com**. You can:

- Access product documentation, if you have Tech Community credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

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.

I

- 2 Starting NaturalONE 7
- 3 The NaturalONE Perspective 11

2 Starting NaturalONE

- Starting the Software AG Designer 8
- NaturalONE Perspective 9
- License Key 9
- Restore of the Natural Server View 9
- Natural for Eclipse Projects 10

Starting the Software AG Designer

NaturalONE is part of the Software AG Designer.

To start the Software AG Designer in Windows

- Choose the following from the Windows Start menu (this is the default entry which can be changed during installation):

  All Programs > Software AG > Tools > Software AG Designer 9.*n*

  Important: When you start the Software AG Designer for the first time with an active Windows Firewall, several dialog boxes will appear, asking whether you want to keep blocking this program. You must choose the **Unblock** button in these dialog boxes.

To start the Software AG Designer under Linux

- Run the start script *naturalone* from a shell.

  This start script is located in the *bin* directory of your installation directory. By default, this is */opt/softwareag/bin*. All parameters that are specified on the command line are passed to the Designer.

  Or:

  Double-click the start script. The Designer will then be started using the default parameters.

When you start the Software AG Designer for the very first time, you will see a so-called "Welcome" page. If this page is not visible, you can open it by choosing the **Welcome** command from the **Help** menu of Eclipse.

From the welcome page, you can switch to the NaturalONE perspective and you can invoke the NaturalONE documentation.

From the welcome page, you can also install sample applications. When you click on a sample application, it is copied to your workspace and shown as a new project there. For further information, see *Sample Applications*.

NaturalONE Perspective

In order to work with NaturalONE, you have to open the NaturalONE perspective. For further information, see ***Opening the NaturalONE Perspective***.

License Key

NaturalONE is protected by a license key. When you install NaturalONE, you are prompted to specify the path to your license file. The license file will then be copied to the *common/conf* directory of your NaturalONE installation.

If the license has expired or no license file is found in the *common/conf* directory, some NaturalONE functionality is disabled. See also *Why are certain views or editors not available?* in *Frequently Asked Questions*.

NaturalONE may be installed with a number of optional components which derive their functionality from the license file. Therefore, if you move the NaturalONE license file to a different location, components such as EntireX are not able to find the license file and may refuse to work.

If you have installed a preliminary version for testing purposes, your license file contains an expiration date. When you decide to buy NaturalONE after the preliminary version has expired, it is not required that you install NaturalONE once more. You just have to copy the new license file which you receive for the licensed version into the *common/conf* directory so that it replaces the previous license file with the same name.

Restore of the Natural Server View

The **Natural Server** view is restored when the NaturalONE perspective is shown after the Software AG Designer has been started (or the first time you switch from another perspective to the NaturalONE perspective after the Designer has been started). When Natural environments have already been mapped (see ***Accessing a Remote Development Environment***) and the option **Restore Natural server view** is selected on the **Runtime Execution** page of the Natural preferences, this may take a while, especially when a large number of objects is stored on a server. A dialog box is shown in this case, indicating the restore progress. During the restore progress, you can already work with objects in other views. Up to a certain point, it is also possible to cancel the restore process.

As long as the **Natural Server** view has not fully been restored, the tree in this view appears gray, indicating that you cannot yet work with this view.

Natural for Eclipse Projects

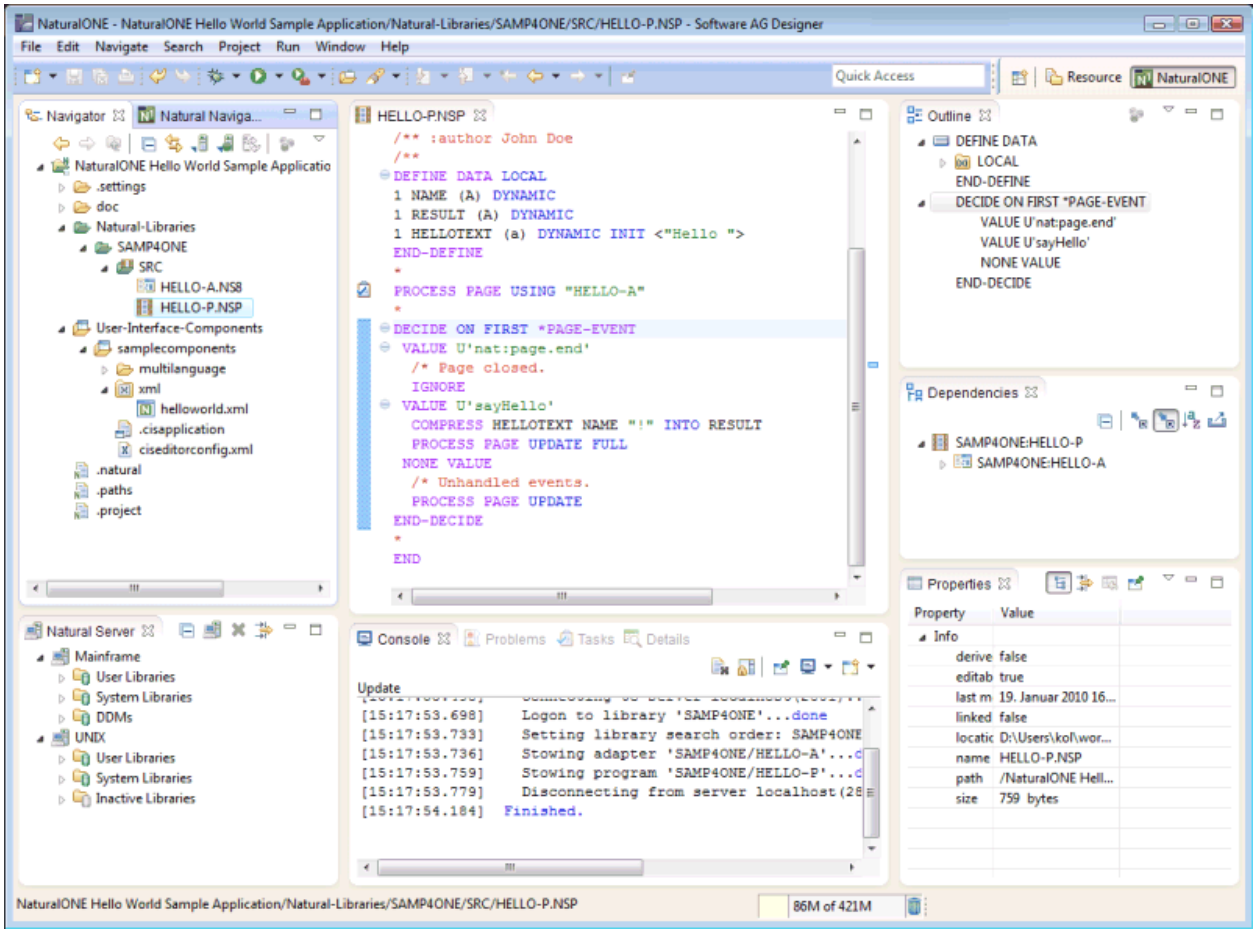
Natural projects which have been created with Natural for Eclipse (NFN) cannot be used immediately with NaturalONE. You have to enable them first. For further information, see *Enabling a Natural Project for NaturalONE*.

3 The NaturalONE Perspective

- Opening the NaturalONE Perspective 12
- Views of the NaturalONE Perspective 13
- Editor Area in the NaturalONE Perspective 15
- Showing a View of the NaturalONE Perspective 15
- Resetting the NaturalONE Perspective 16

Opening the NaturalONE Perspective

In order to use NaturalONE, you have to open the NaturalONE perspective. The following is an example of the NaturalONE perspective; it shows a Natural project where a Natural program has been opened for editing.



When the NaturalONE perspective is not active, you can open it as described below.

To open the NaturalONE perspective

1 From the **Window** menu, choose **Open Perspective > Other**.

2 In the resulting **Open Perspective** dialog box, select **NaturalONE** and choose the **OK** button.

The NaturalONE perspective is opened. The views which are part of the NaturalONE perspective are described in the section below.

Views of the NaturalONE Perspective

The NaturalONE perspective makes use of several views. Some of these views are Natural-specific views and other views are standard Eclipse views. In addition to the Eclipse views which are shown by default in the NaturalONE perspective, you can also use any other Eclipse view.

The following list briefly describes the most important views of the NaturalONE perspective (these are the views which are shown by default when you open the NaturalONE perspective for the first time):

- **Navigator**
  Shows the Natural projects defined in your workspace. Other types of projects (for example, Java projects) will also appear in this view. For further information, see ***Working with Natural Projects in Local Mode***.
- **Natural Navigator**
  When none of the toggle buttons is selected in the local toolbar of this view, the **Natural Navigator** view shows the same information as the standard Eclipse **Navigator** view. However, the **Natural Navigator** view offers enhanced support for Natural projects and allows you to switch on and off different Natural-specific options for displaying your Natural projects in the tree. For further information, see ***Using the Natural Navigator View***.
- **Natural Server**
  Used to make a connection to a Natural server. For further information, see ***Accessing a Remote Development Environment***.
- **Outline**
  The different types of Natural editors show different information in this view. For further information, see ***Using the Natural Editors***.
- **Dependencies**
  Shows the dependencies to other Natural objects that are referenced in the active editor window. For further information, see ***Using the Natural Editors***.
- **Properties**
  Shows the properties for a selected item. NaturalONE uses this view to display Natural-specific information.
- **Console**
  When console output is enabled in the Natural **preferences** (different types of console output can be enabled there), NaturalONE uses this view to display, for example, information from the Natural builder or output of the internal Tomcat server. You can also open a console in which you can enter Natural system commands; for further information, see ***Using the Natural Command Console for Mainframes***.

- **Problems**
  NaturalONE uses this view to display Natural-specific problems such as compiler errors, parser errors or NATstyle violations. For further information, see *Problems in Your Natural Sources* and *Checking Natural Code with NATstyle*.
- **LastMsg**
  NaturalONE uses this view to display additional information about the error situation which has occurred last. This functionality can be activated or deactivated by setting the preference **Retrieve error messages from runtime environment** in the **LastMsg** tab of the **Natural** preferences.

  Any time a Natural command is issued, the potentially created error messages in the runtime environment are transferred to NaturalONE and displayed inside this view; for further information, see *Using the LastMsg View*.
- **Tasks**
  Shows the tasks (for example, for programming steps that still need to be done) that you have added to your source code. NaturalONE uses this view to display tasks of type "Natural Task". For further information, see *Working with Tasks*.
- **Details**
  Shows detailed information on the children of the project, library or folder that is currently selected in the **Navigator** view or in the **Natural Navigator** view. For further information, see *Using the Details View*.

The following Natural-specific views are not shown by default when you open the NaturalONE perspective:

- **NATdoc**
  Shows how the NATdoc-specific comments in the source code will appear in the API documentation. For further information, see *Previewing the API Documentation in the NATdoc View*.
- **Profiler Sessions**
  Shows an entry for each program for which you have started a profiler session. For further information, see *Using the Natural Profiler*.
- **Report Data**
  Shows reports that have been created from one or more DDMs. For further information, see *Using the Data Browser*.
- **RTL Visual Order**
  Shows the line which is currently selected in the active editor window in different screen directions: right-to-left (RTL) and left-to-right (LTR). For further information, see *Bidirectional Language Support*.
- **Time Stamp Conflicts**
  Shows all time stamp conflicts in the Natural environment. For further information, see *Checking the Time Stamps in the Natural Environment*.

- **Unlock Objects**
  Shows the objects which are currently locked on a Natural server. For further information, see *Unlocking Locked Objects*.
- **Code Coverage**
  Shows the percentage of executed code of a natural application. For further information, see *Using the Natural Code Coverage*.
- **Mainframe Navigation**
- **Server Logs Viewer**

Editor Area in the NaturalONE Perspective

One of the following Natural editors is shown in the editor area when you open a Natural object:

- source editor
- map editor
- DDM editor

For further information on the above editors, see *Using the Natural Editors*.

Other editors are shown in the editor area, for example, when you open the following items:

- Report template. For further information, see *Using the Data Browser*.
- Message file. For further information, see *Creating Application-Specific Messages*.
- Excludes file. For further information, see *Excluding Objects from Processing in the Natural Environment*.
- Profiler session. For further information, see *Using the Natural Profiler*.

Showing a View of the NaturalONE Perspective

If a view of the NaturalONE perspective is currently not shown, you can display it as described below.

To show a Natural view

1. From the **Window** menu, choose **Show View > Other**.
2. In the resulting **Show View** dialog box, expand the **Software AG NaturalONE** node and select one or all of the following views:
   - **Dependencies**

- **Details**
- **NATdoc**
- **Natural Navigator**
- **Natural Server**
- **Profiler Sessions**
- **Report Data**
- **RTL Visual Order**
- **Time Stamp Conflicts**
- **Unlock Objects**

Notes:

1. The views which are listed above pertain to the basic functionality of NaturalONE as described in this *Using NaturalONE* documentation. Other views in the **Software AG NaturalONE** node which are not listed above are used by other (optional) components of NaturalONE. See the corresponding documentation for further information.
2. When you expand the **General** node in the **Show View** dialog box, most of the standard Eclipse views (such as the **Navigator** view or the **Outline** view) can be selected.

3 Choose the **OK** button.

Resetting the NaturalONE Perspective

When you have closed one or more views, or moved a view to a different location, you can reset the perspective so that its default settings are used again.

To reset the NaturalONE perspective

1 Make sure that the NaturalONE perspective is active.

2 From the **Window** menu, choose **Reset Perspective**.

3 In the resulting dialog box, choose the **OK** button to confirm that you want to reset the perspective to its defaults.

II Working with Natural Projects in Local Mode

The so-called local mode is the preferred development mode in NaturalONE. In this mode, the sources on which you are working are organized in projects that are stored in the Eclipse workspace. See also *Different Modes for Developing Natural Applications* in the *Introduction*.

This part describes how to manage Natural projects in the Eclipse workspace. It covers the following topics:

Viewing the Contents of Your Natural Projects

Managing Natural Projects

Managing Libraries in a Natural Project

Managing Objects in a Natural Project

Launching Natural Applications

Modifying Objects in the Natural Environment or in the Repository

Understanding the Behavior of the Natural Builder

4 Viewing the Contents of Your Natural Projects

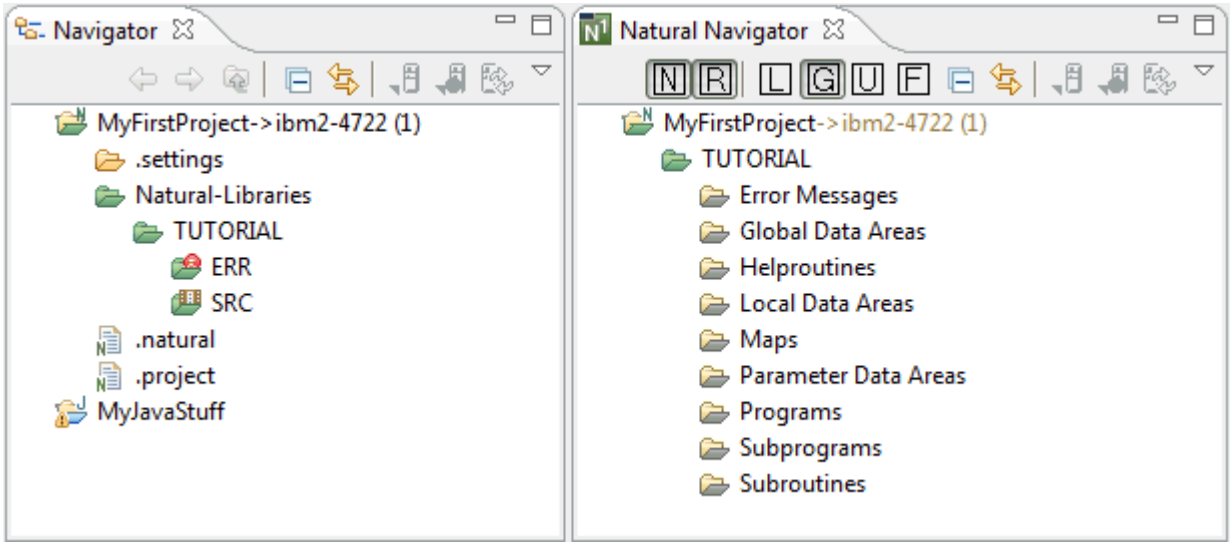
- General Information 20
- Using the Natural Navigator View 21
- Customizing the Natural Navigator View 22
- Filtering Natural Objects and Libraries 24
- Using the Details View 32

General Information

When you have chosen to work in local mode, you have to deal with Natural projects which are shown in the **Navigator** view and/or in the **Natural Navigator** view. Projects are essential parts of the Eclipse environment. In Eclipse, the resources (folders and files) in the workspace are always organized in projects.

The **Navigator** view is a standard Eclipse view which always shows a project as it is physically stored in the file system. The **Natural Navigator** view, however, allows you to manage, view and/or filter your Natural projects in several Natural-specific ways; see ***Using the Natural Navigator View*** for further information.

The following example shows how the projects that are shown in the **Navigator** view can be shown in the **Natural Navigator** view. According to the settings of the toggle buttons, the **Natural Navigator** view shows only Natural projects, the root folder "Natural-Libraries" is hidden and the objects are logically grouped by type.



A Natural project contains the contents of one or more Natural libraries, and also other folders and files which are added by other components of NaturalONE such as Ajax Developer. A Natural project is indicated by the following icon: .

By default, the environment to which a Natural project pertains is shown next to the project name. The number in parentheses which is also shown next to the project name indicates the number of Natural libraries in this project. This is part of the so-called **label decoration**.

Important information for this environment (such as the Natural parser settings or whether the environment is protected by Natural Security) is stored in the properties of the project. See ***Changing the Project Properties***.

Caution: Each Natural project includes standard project information which is required by Eclipse (*.settings* folder and *.project* file) and by Natural (*.natural* file). You must not change these items manually since this may result in damaging your project.

Using the Natural Navigator View

The **Natural Navigator** view is a Natural-specific view. If it is currently not shown, you can display it as described under ***Showing a View of the NaturalONE Perspective***.

Whereas the **Navigator** view, which is a standard Eclipse view, always shows your projects as they are physically stored in the file system, the **Natural Navigator** view can be configured to show your projects in a logical way. It allows you to switch on and off different Natural-specific options for displaying your Natural projects in the tree. In addition, it provides enhanced support, for example, for copying and pasting Natural objects (for detailed information, see ***Copying and Pasting Objects and Libraries***).

The local toolbar of the **Natural Navigator** view provides the toggle buttons described below. The view menu of the **Natural Navigator** view (drop-down menu at the top right of the view) contains the corresponding menu commands. When none of the toggle buttons is selected in the local toolbar of the **Natural Navigator** view, the **Natural Navigator** shows the same information as the standard Eclipse **Navigator** view.

Toggle Button	View Menu Command	Description
N	**Show Natural Projects Only**	When selected, only the Natural projects are shown. When not selected, all projects in your Eclipse workspace (including, for example, your Java projects) are shown.
R	**Hide Root Folders**	When selected, the root folders are not shown. When not selected, all root folders (for example, "Natural-Libraries") are shown in the Natural projects.
L	**Show Library View**	When selected, all objects in a Natural library are shown as they would be stored on the server. The special folders *SRC*, *ERR* and *RES* are not shown. When **library folders** have been defined, they are not shown. When not selected, the Natural libraries and all objects in a library are shown as they are physically stored in your Eclipse workspace in the file system. In addition, if library folders have been defined, their names are shown (if more than one library folder has been defined for a single library, more than one node is shown for the same library). This is similar to the representation in the **Navigator** view.
G	**Group Objects by Type**	When selected, all objects in a Natural library are logically grouped into different nodes, according to their object types. This is similar to the **Natural**

Toggle Button	View Menu Command	Description
		Server view. For example, your programs are grouped into a **Programs** node and your subprograms are grouped into a **Subprograms** node. The special folders *SRC*, *ERR* and *RES* are not shown. When not selected, all objects in a library are shown as they are physically stored in your Eclipse workspace in the file system. However, whether the special folders *SRC*, *ERR* and *RES* are shown depends on the setting of the **Show Library View** command (L toggle button). **Note:** When the option **Group new objects by object type** has been selected for a Natural project, the physical representation may be the same as the logical representation. See also ***Group Folders***.
U	**Show Objects with Update Flags Only**	When selected, only the modified Natural objects which still need to be updated in the Natural environment are shown for a Natural project. These are the objects which contain specific flags in the label decoration. For further information, see ***Flags in a Label Decoration***. When not selected, all objects of a Natural project are shown.
F	**Hide File Extension**	When selected, the file extensions of the Natural objects are not shown. When not selected, the file extensions of the Natural objects are shown (for example, ".NSP" for a Natural program).

When the **Natural Navigator** view shows the objects in a logical way (that is, when they are not shown as they are physically stored in the file system), the icons for the virtual libraries and group folders and are shown with a gray color.

Note: When the *.settings* folder, the *.project* file and the *.natural* file are not shown in the **Natural Navigator** view, the option **.*resources** has been selected in the **Available Customizations** dialog box (see ***Customizing the Natural Navigator View***).

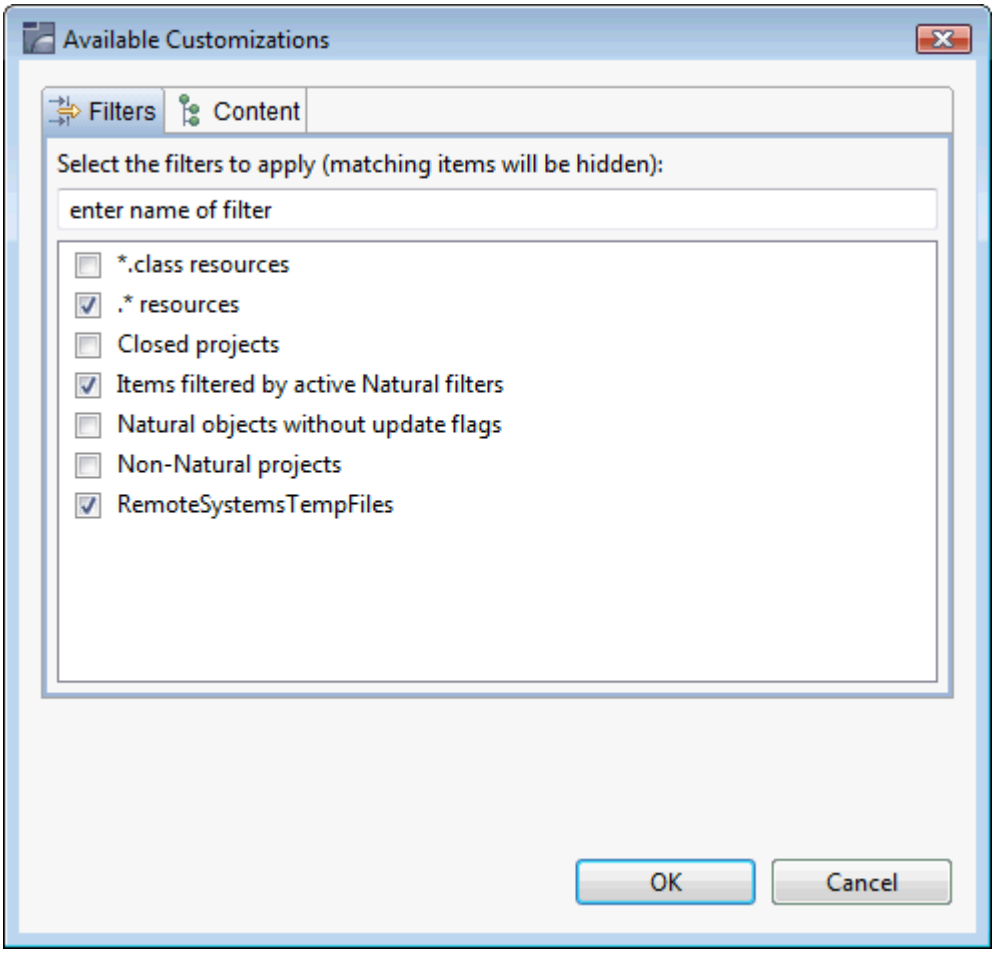
Customizing the Natural Navigator View

You can customize the **Natural Navigator** view so that the items that you do not want to see are not shown.

To customize the Natural Navigator view

1 From the view menu of the **Natural Navigator** view (drop-down menu at the top right of the view), choose **Customize View**.

The **Available Customizations** dialog box appears.



You can define the following filters, in addition to the standard Eclipse filters which are shown in this dialog box:

Option	Description
Items filtered by active Natural filters	When selected, the Natural filters that have been applied to a node are active and it is possible to set additional filters. The items that you do not want to see in the **Natural Navigator** view are not shown. This corresponds to the **Activate Natural filters** option in the **Natural Filters** dialog box. For further information, see *Filtering Natural Objects and Libraries*.
Natural objects without update flags	When selected, all Natural objects are hidden which do not have update flags. Only the modified objects which still need to be updated in the Natural environment are shown. This corresponds to the **Show Objects with Update Flags Only** command in the view menu (U toggle button in the local toolbar) of the **Natural Navigator** view.
Non-Natural projects	When selected, non-Natural projects (for example, Java projects) are hidden and only the Natural projects are shown. This corresponds to the **Show Natural Projects Only** command in the view menu (N toggle button in the local toolbar) of the **Natural Navigator** view.

2 Select the filters that you want to apply.

3 Choose the **OK** button.

Filtering Natural Objects and Libraries

Using a filter, you can reduce the number of items that are shown in the **Natural Navigator** view. Filtering involves several steps: First you define a filter and then you set the filter on a node of the **Natural Navigator** view (for example, on a project or library node). Detailed information is provided in the following topics:

- Managing the Natural Filters
- Defining a New Natural Filter
- Setting a Natural Filter
- Removing a Natural Filter

Note: The information below only applies for the **Natural Navigator** view. Eclipse also offers filtering functionality in its **Navigator** view: you can define resource filters in the properties of a project or folder. These resource filters will be written to the *.project* file. If you are using a version control system, keep in mind that you also have to keep the *.project* file in your versioning repository. For Natural projects, however, filtering information in the *.project* file is not desirable. It is therefore recommended that you use the filtering functionality of the **Natural Navigator** view, which is described below, because the Natural filters are not written to the *.project* file.

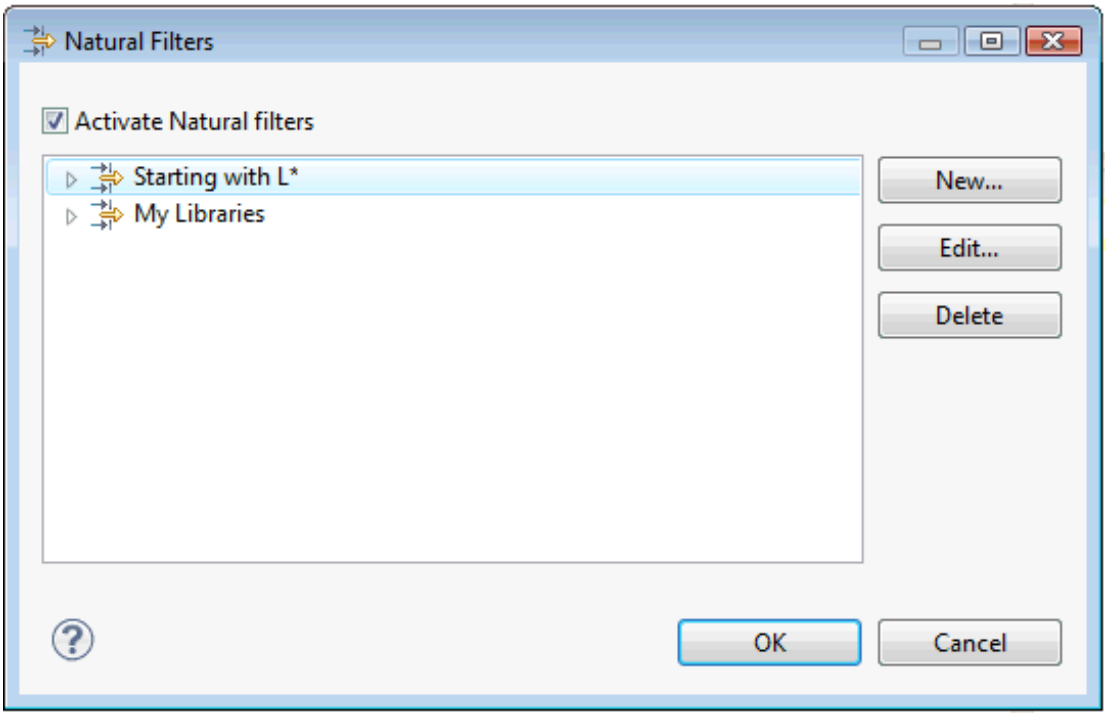
Managing the Natural Filters

If you want to define a new Natural filter, you first have to invoke the **Natural Filters** dialog box as described below. Using this dialog box, you can also edit and delete existing Natural filters, or you can deactivate all Natural filters that have already been set in the **Natural Navigator** view.

To manage the Natural filters

1 From the view menu of the **Natural Navigator** view (drop-down menu at the top right of the view), choose **Natural Filters**.

The **Natural Filters** dialog box appears.

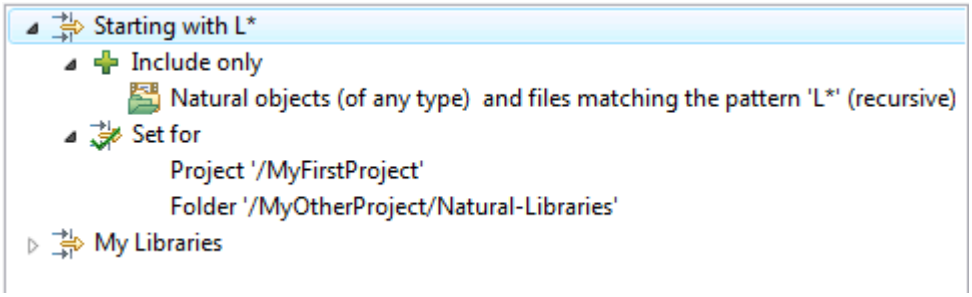


When the **Activate Natural filters** check box is selected (default), it is possible to add new filters, and to edit or delete existing filters. In addition, all Natural filters that have been set in the **Natural Navigator** view are active and it is possible to set further filters. See also *Setting a Natural Filter*.

Notes:

1. When you change the setting of the **Activate Natural filters** check box, the setting of the **Items filtered by active Natural filters** option in the **Available Customizations** dialog box is adapted accordingly (see *Customizing the Natural Navigator View*).
2. The **Natural Filters** dialog box can also be invoked from the **Set Natural Filters** dialog box (see *Setting a Natural Filter*). However, the **Activate Natural filters** check box is not available in this case.

When filters are already defined, they are shown in the **Natural Filters** dialog box. When you move the mouse pointer over a filter, a tooltip is shown providing information on this filter. When you expand the node for a filter, you can see more detailed information on the filter. When the filter is currently active in the **Natural Navigator** view, a **Set for** node is shown, informing you on which nodes this filter has been set. For example:



The following command buttons are provided:

Command Button	Description
New	Defines a new filter. See also *Defining a New Natural Filter*.
Edit	Edits the selected filter.
Delete	Deletes the selected filter. See also *Removing a Natural Filter*.

2 Choose the **OK** button to save the setting of the **Activate Natural filters** check box and to close the **Natural Filters** dialog box.

Defining a New Natural Filter

When you define a new Natural filter, you specify a name, a pattern (for example, that only items are to be shown which start with the letter "L"), and additional optional filter attributes. The filters that you define can then be set on a node of a Natural project in the **Natural Navigator** view.

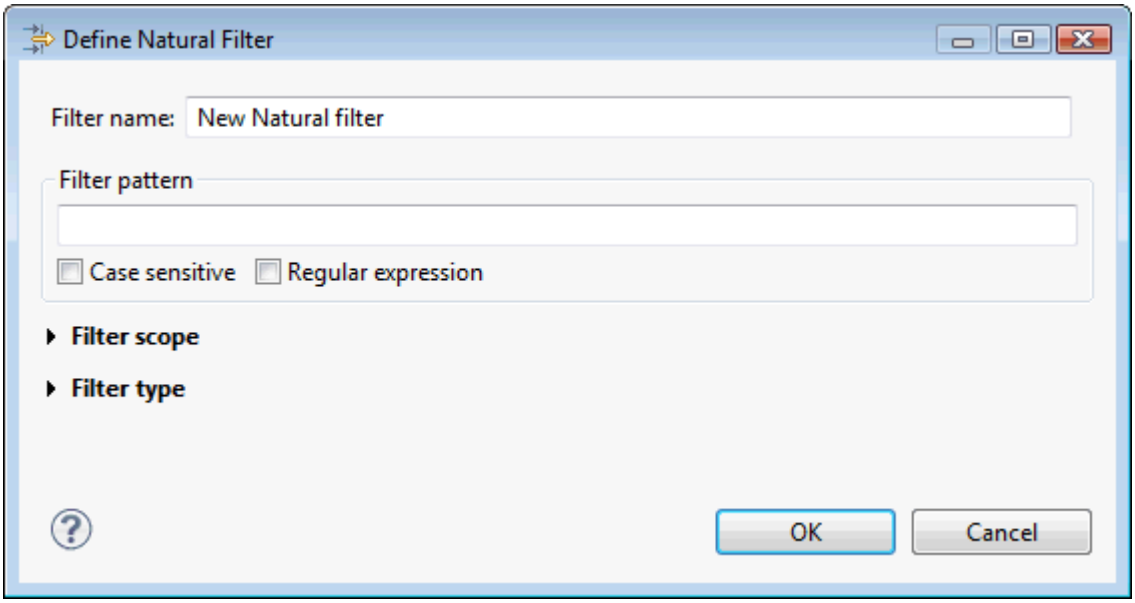
To define a new Natural filter

1 Invoke the **Natural Filters** dialog box as described above.

2 Make sure that the **Activate Natural filters** check box is selected. Otherwise, it is not possible to define a new filter.

3 Choose the **New** button.

The **Define Natural Filter** dialog box appears.

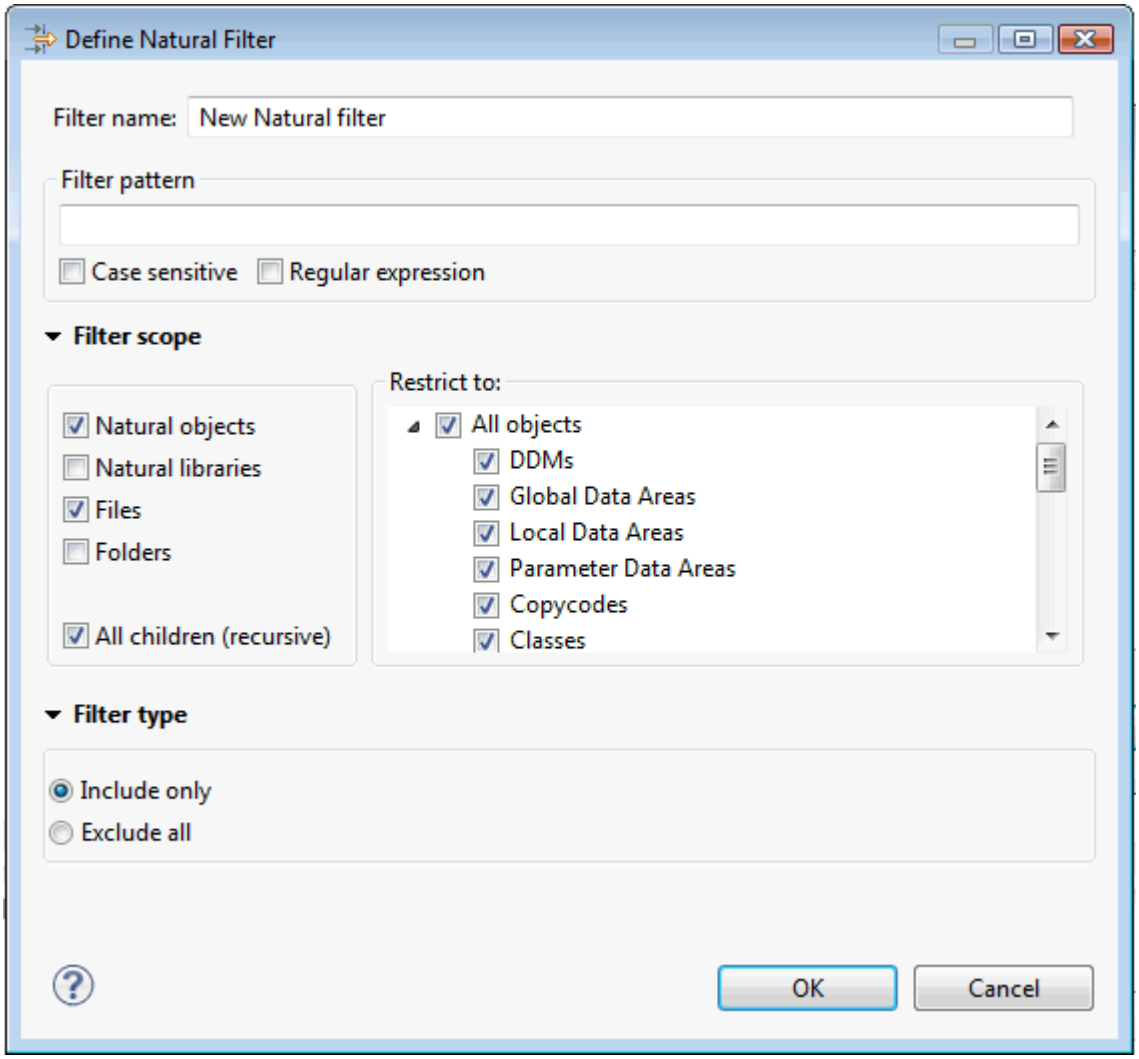


4 Specify the following information:

Option	Description
Filter name	The name for the filter. You can use any name (for example, "My Libraries").
Filter pattern	The pattern for the filter, that is, the names of all items that are to be filtered. You can use wildcards (? or *) within the names if you do not want to enter each name individually. The question mark (?) may be specified at any position inside the name. The asterisk (*) is only allowed at the end of a name. You can also use the escape character (/) for literals. **Note:** A tooltip for the wildcard and escape characters is shown when you move the mouse pointer over this text box.
Case sensitive	If selected, only items which match exactly the defined uppercase or lowercase letters in the pattern are shown.
Regular expression	If selected, the filter pattern is interpreted as a regular expression. In this case, you can press CTRL+SPACEBAR to invoke content assist for the regular expression.

5 Optional. Expand **Filter scope** and/or **Filter type**.

Further filter attributes are shown in the dialog box.



6 To define the filter scope, select one of more of the following check boxes:

Option	Description
Natural objects	If selected (default), the filter applies to the names of the Natural objects selected in the **Restrict to** group box. This group box is only visible when **Natural objects** is selected.
Natural libraries	If selected, the filter applies to the names of Natural libraries.
Files	If selected (default), the filter applies to the physical file names of Natural objects and non-Natural files.
Folders	If selected, the filter applies to the names of library folders and non-Natural folders.
All children (recursive)	If selected (default), all children of the selected node are filtered.

Make sure to define a reasonable filter scope. By default, the filter applies to the names of all Natural objects, the corresponding physical file names, and the names of all child nodes. If

you would select, for example, the **Natural libraries** check box in addition to the default setting, the filter would also apply to the names of the Natural libraries, which is redundant.

7 To define the filter type, select one of the following option buttons:

Option	Description
Include only	An **Include only** filter (default) allows only items that match the filter condition to be shown in the **Natural Navigator** view. If multiple **Include only** filters exist, the items shown will be those which match any of the existing **Include only** filters.
Exclude all	An **Exclude all** filter prevents all items that match the filter condition to be shown in the **Natural Navigator** view. If multiple **Exclude all** filters exist, the items excluded will be those which match any of the existing **Exclude all** filters.

If both **Include only** and **Exclude all** filters exist in a given folder or project, only items that match any of the **Include only** filters and do not match any of the **Exclude all** filters will be shown in the **Natural Navigator** view.

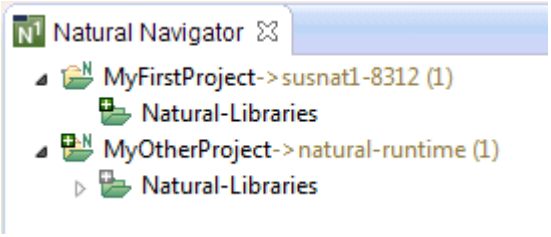
8 Choose the **OK** button to save your changes and to close the **Define Natural Filter** dialog box.

The following settings are stored after the dialog box has been closed: size and position of the dialog box, and collapsed or expanded state of the **Restrict to**, **Filter Scope** and **Filter Type** group boxes.

Setting a Natural Filter

Each Natural filter can be set on a node of a Natural project in the **Natural Navigator** view.

When a Natural filter has been set, the icons that are shown for the nodes contain additional plus signs. The background color of a plus sign indicates where the filter has been set. A green background indicates that the filter has been set on that node. A gray background indicates that the filter has been set on a parent node. Example:



Note: The label decorations for the Natural filters are controlled by the preferences under **General > Appearance > Label Decorations**. If you do not want to have label decorations for the Natural filters, just go to the above mentioned preference page and deselect **Natural Filters**.

To set a Natural filter

1 In the **Natural Navigator** view, select the node on which you want to set a Natural filter.

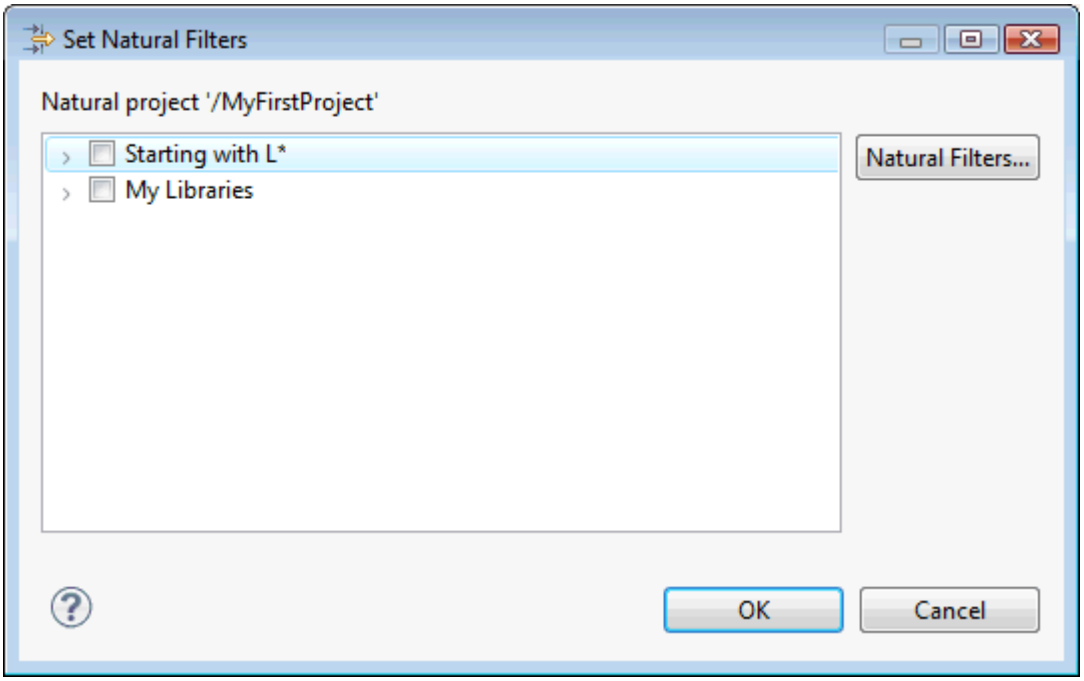
When you select a project node, you can reduce the number of Natural objects, Natural libraries, Natural library folders, files or folders in this node.

When you select a library node or a library folder node, you can reduce the number of Natural objects, files or folders in this node.

2 Invoke the context menu and choose **NaturalONE > Set Natural Filters**.

Note: This command is only available if the use of Natural filters has been activated in the **Natural Filters** dialog box (see *Managing the Natural Filters*) or in the **Available Customizations** dialog box (see *Customizing the Natural Navigator View*).

The **Set Natural Filters** dialog box appears. It lists all filters that are currently defined. The type and name of the node on which you are about to set the filter is shown at the top of the dialog box.



3 Activate the check box for each filter that you want to set.

As in the **Natural Filters** dialog box, tooltips are provided for the defined filters, and you can see more information when you expand the node for a filter. See also *Managing the Natural Filters*.

Note: When you choose the **Natural Filters** button, the **Natural Filters** dialog box appears in which you can define additional filters or change the existing filters.

4 Choose the **OK** button.

The content of the selected node in the **Natural Navigator** view changes so that only the items that match your filter are shown. These filter settings are always applied to the node, no matter which toggle buttons in the toolbar of the **Natural Navigator** view are currently selected.

Removing a Natural Filter

You can either deactivate a filter for a node or you can delete a filter so that it is no longer available.

To deactivate a Natural filter for a node

1 In the **Natural Navigator** view, select the node for which you want to deactivate a filter. Make sure to select the parent node. The icon on such a node has a plus sign with a green background.

2 Invoke the **Set Natural Filters** dialog box as described above (see *Setting a Natural Filter*).

3 Deselect the check box for each filter that you want to deactivate.

4 Choose the **OK** button.

To delete a Natural filter

1 Invoke the **Natural Filters** dialog box as described above (see *Managing the Natural Filters*).

2 Make sure that the **Activate Natural filters** check box is selected. Otherwise, it is not possible to delete a filter.

3 Select the filter that you want to delete.

4 Choose the **Delete** button.

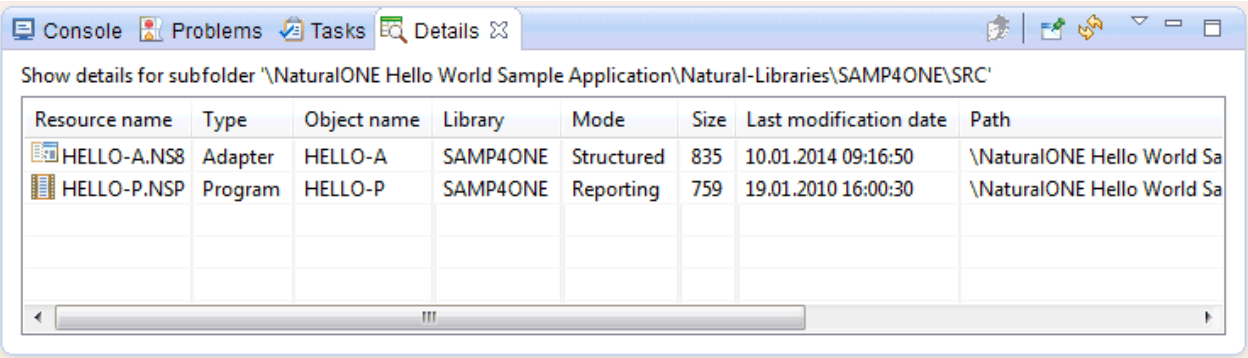
You are asked whether you really want to delete the filter. If the filter is currently used on a node of the **Natural Navigator** view, a corresponding message is shown.

5 Choose the **Yes** button to delete the filter.

Using the Details View

The **Details** view is a Natural-specific view. If it is currently not shown, you can display it as described under *Showing a View of the NaturalONE Perspective*.

The **Details** view always shows detailed information on the children of the project, library or folder that is currently selected in the **Navigator** view or in the **Natural Navigator** view. Example:



You can sort a column by clicking on its column header, and you can move a column by dragging the column header to another location.

Using the **Pin Details** command in the view menu (or the corresponding button in the local toolbar), you can freeze the content of the **Details** view. This means, that the content of the **Details** view does not change when you select a different node in the **Navigator** view or in the **Natural Navigator** view.

By default, the **Details** view is updated automatically if one of the values shown in this view changes. You can switch off this behavior by deactivating the **Refresh Automatically** command in the view menu. In any case, you can always refresh the view manually by pressing F5 or by using the **Refresh** command in the view menu (or the corresponding button in the local toolbar).

The **Details** view only shows the columns which currently contain at least one value. If a column does not contain any value, it is not shown. For example, the **Size** column is not shown if only folders are contained in the view.

The following columns can be shown in the **Details** view.

Column Name	Description
Resource name	The Eclipse resource name, which is either a file name or a folder name.
Type	Either the type of a Natural object (for example. "Program") or the type of a folder (for example, "Library Folder").
Object name	The Natural object name. This column is only filled for Natural objects.
Long name	The long name of a Natural object. This column is only filled for Natural objects which have a long name (for example, DDMs).
Library	The Natural library to which a Natural object or folder belongs. For folders which are not assigned to a specific library, this column is not filled.
Private-mode library	The name of the private-mode library for Natural objects and folders when private mode has been defined for the project or library. This column is not filled for projects and libraries for which shared mode has been defined.
Mode	The programming mode of a Natural object. This can be either "Structured" or "Reporting".
Count	The number of direct children for libraries and folders.
Size	The size of a file. This column is not filled for libraries and folders.
Server encoding	The server encoding of a Natural object. This column is not filled for libraries and folders. It is also not filled for Natural objects for which a server encoding has not been defined.
Last modification date	The date and time of the last modification of the Eclipse resource.
Path	The path to the Eclipse resource. In the case of a virtual library or group folder (**Natural Navigator** view), the string "<virtual>" is shown.

You can invoke a context menu for each item that is currently selected in the **Details** view. For example, you can use the **NaturalONE > Update** command to upload and stow selected sources. Keep the following in mind when working with the context menu:

- **Selecting several items**
  When you select several items in the **Details** view, the context menu only contains the commands which are allowed for all selected items.
- **Open command (or double-click)**
  When you use the **Open** command with a folder or library (or when you double-click a folder or library), the children of that folder or library are shown in the **Details** view. When you then select an item in the **Details** view and choose **Show Details for Parent Folder** from the view menu (or choose the corresponding button in the local toolbar), the parent folder of the selected item is shown in the **Details** view.

  When you use the **Open** command with a file (or when you double-click a file), the file is opened in the appropriate editor.

- **Copy command**
  Using the **Copy** command, you can copy the visible columns of all selected rows to the clipboard, from where you can paste them into an application of your choice. The column headers are also copied. The contents of the individual columns are separated by semicolons (;).

5 Managing Natural Projects

- Creating Natural Projects 36
- Types of Natural Projects 42
- Changing the Project Properties 43
- Quickly Viewing the Properties 64
- Enabling a Project for NaturalONE 65

Creating Natural Projects

The name of a Natural project must meet the Eclipse naming conventions.

You can create a Natural project in different ways:

- Downloading an Existing Library or Object from a Natural Server
- Creating a New Project Using a Wizard
- Importing a Project from a Version Control System
- Importing an Existing Natural Project into the Workspace

Downloading an Existing Library or Object from a Natural Server

You can download entire libraries from a Natural server. A library is either placed in a newly created project or in an existing project, depending on the command that you use.

It is also possible to download only single objects of a library. These objects are then downloaded into a library which has the same name as the library on the Natural server (into a new or existing project, depending on the command that you use).

Only the sources are downloaded from the Natural server. Since a Natural application can only be executed (and debugged) directly in the Natural server environment, the generated programs (which are also called "GPs" or "cataloged Natural objects") remain on the server.

When you download into a new project, a number of required Natural profile parameter settings is also downloaded to the project. These settings are stored as the default settings in the *.natural* file. See also ***Changing the Project Properties***.

Notes:

1. In a UNIX, OpenVMS or Windows environment, it is possible to download an inactive library into a new project. However, when you upload the content of such a library later, it is always uploaded to the active system file and not to the inactive library from which it was downloaded.
2. When the option **Console output** is enabled on the **Runtime Execution** page of the Natural preferences, information about the download process is shown in the **Console** view.

To download a library or object

1 In the **Natural Server** view, map the required Natural server as described in the section ***Mapping a Natural Environment***.

2 Expand the node for the mapped Natural server.

3 Expand the node for the required system file (for example, **User Libraries**).

4 Select one or more of the following: a library, a node for an object type (for example, **Program**) or a single object.

Tip: Type the first letters of the library name to select the first library that starts with these letters. When you type all letters of a name quickly, without a pause, you can immediately select the corresponding library.

If you want to download several single objects at a time, open the library node(s), then open the node(s) for the object type(s), and then select the object(s).

If you want to download, for example, all subprograms of a library, select the corresponding node for this object type.

Notes:

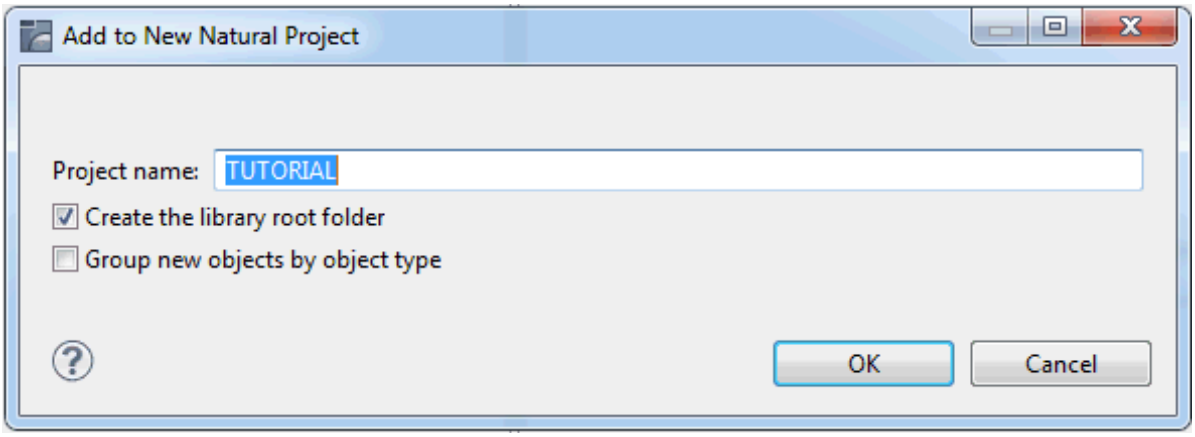
1. When Natural Security is not active on the Natural server, you can also select the node for a system file if you want to download all libraries contained in this system file at the same time.
2. When Natural Security is active on the Natural server, it is only possible to download objects for which editing and listing is allowed. Since the access to specific libraries may be restricted, it is not possible to download all libraries of a system file at the same time by selecting the node for the system file. In this case, you have to select the nodes for all libraries that you are allowed to use. Moreover, if editing or listing is not allowed for one object type in a library, it is not possible to download the complete library. In this case, you have to select the group nodes for the types which you are allowed to use.
3. When Natural Security is active on the Natural server, "copy from library" must be enabled in the `SYSMAIN` utility profile of Natural Security in order to download objects. See also *Protecting the Natural Development Environment in Eclipse* in the *Natural Security* documentation, which is part of the Natural documentation.

5 Invoke the context menu and choose either **Add to New Project** or **Add to Existing Project**.

With **Add to New Project**, a new Natural project is created and the **project properties** are set. The properties apply to the Natural server environment from which the objects are downloaded.

With **Add to Existing Project**, the project properties of the existing project are not modified even if they do not apply to the Natural server environment from which the objects are downloaded.

6 When you have chosen to download to a new project, a dialog box appears and you can specify a project name. The name of the (first) selected node is offered as the default name. When a project with this name already exists in your workspace, the dialog box informs you and you have to specify a project name that is not yet used.

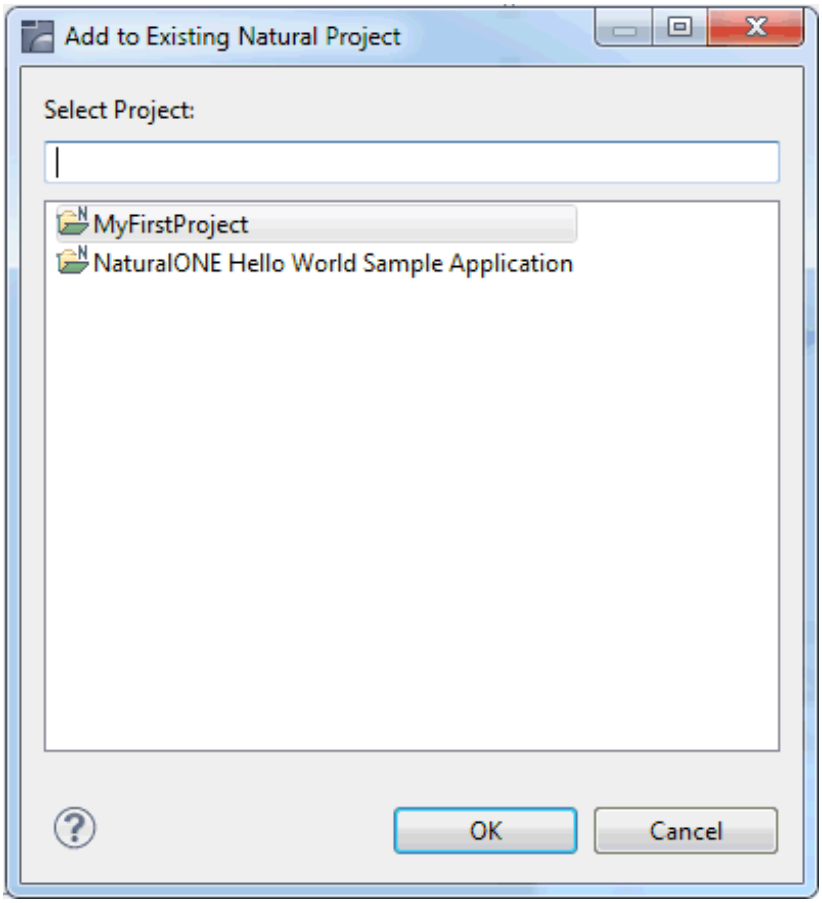


If you want to create the "Natural-Libraries" root folder in the new project, make sure that the option **Create the library root folder** is selected. This option is only shown when root folder support has been enabled in the Natural preferences. For further information, see *Natural > Project* in *Setting the Preferences*.

If you want to group the downloaded Natural objects into separate folders within the new project, according to their object types, make sure that the option **Group new objects by object type** is selected. For further information, see *Group Folders*.

Or:

When you have chosen to download into an existing project, a dialog box appears in which you have to select the project.



7 Choose the **OK** button to continue.

A dialog box is shown indicating the download progress. When you download a small number of objects, this dialog box is only briefly shown. However, when you download a large number of objects, you can choose to download the objects in the background so that you are able to continue working. A progress indicator is shown at the right bottom of the Eclipse window.

The downloaded libraries or objects are shown in the **Navigator** view and in the **Natural Navigator** view.

To download a library or object to an existing project using drag-and-drop

1 In the **Natural Server** view, select one or more libraries or objects as described above.

2 Drag the selected nodes onto an existing project node in the **Navigator** view or in the **Natural Navigator** view.

Creating a New Project Using a Wizard

Instead of downloading a library as a project from a Natural server, you can also create a Natural project manually.

You only have to provide a project name. The project wizard creates a Natural project structure that contains the *.natural* file with the default settings of the project properties. All other project information may be entered at a later point in time. See also *Changing the Project Properties*.

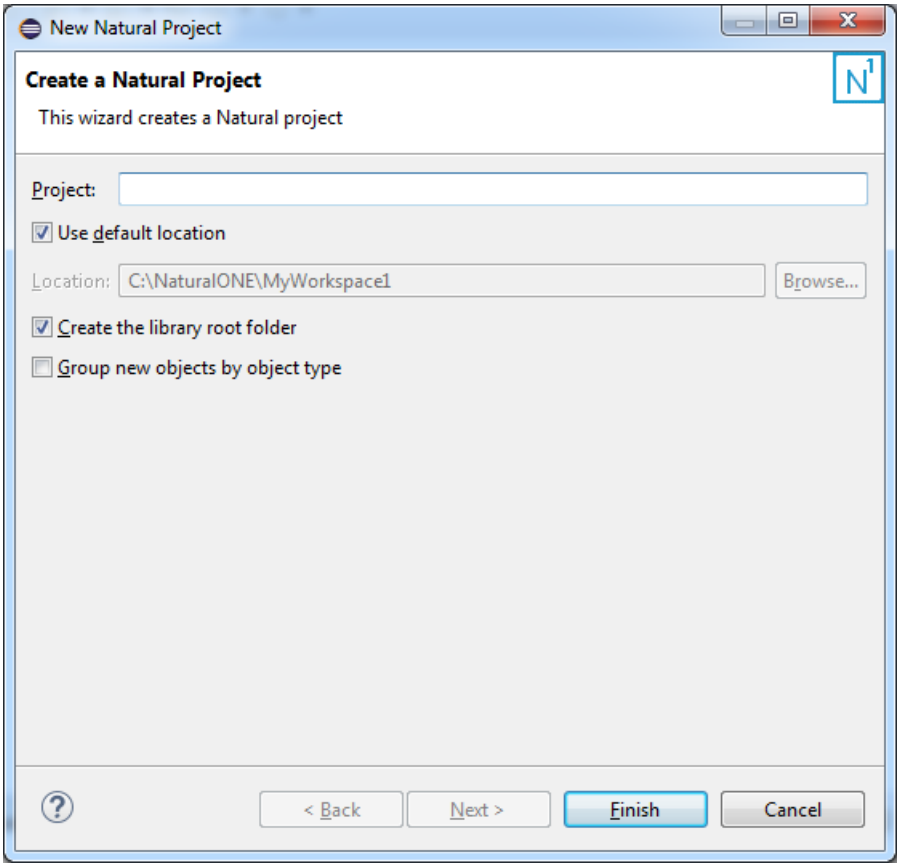
To create a project using a wizard

1 From the **File** menu, choose **New > Natural Project**.

Or:

In the **Navigator** view or in the **Natural Navigator** view, invoke the context menu and choose **New > Natural Project**.

The following dialog box appears.



2 Specify the following information:

Option	Description
Project	A unique name for the new project.
Use default location.	Natural projects are usually created under the Eclipse workspace root. With this option, you can change the location so that projects can be created outside of the Eclipse workspace. You can view the location of an existing Natural project by displaying the properties of that project.
Create the library root folder	If you want to create the "Natural-Libraries" root folder in the new project, make sure that this option is selected. This option is only shown when root folder support has been enabled in the Natural preferences. For further information, see *Natural > Project* in *Setting the Preferences*.
Group new objects by object type	If you want to group the Natural objects in this project into separate folders, according to their object types, make sure that this option is selected. For further information, see *Group Folders*.

3 Optional. Choose the **Next** button repeatedly if you want to change the default settings.

On the next pages of the dialog box, you can then specify the following information:

- Settings for the connection to a Natural server. By default, the local Natural runtime will be used. This enables you to upload projects without having to define any connection settings.

  The advantage of using a local Natural runtime is that you can immediately test and debug your Natural applications which are located in the **Navigator** view or in the **Natural Navigator** view, without the need of having Natural Development Server (NDV) installed. To deploy your applications, however, remote Natural server access via Natural Development Server (NDV) is required; this is also required if you want to test or debug applications which read or write data from/to Adabas.

  If you want to create a project which applies to a Natural server, you have to specify all required mapping information. See also *Mapping a Natural Environment*. When you are working in a new workspace and create a project, the user ID in the project wizard is initially set to the user ID that you have used to log on to your operating system. Each time you enter a different user ID in the project wizard, this user ID is persistently stored in Eclipse and will be provided as the default value the next time you create a new project.
- Environment settings (steplibs).
- Regional settings and character assignments.
- Bidirectional language settings (only displayed if **layout orientation** selected on the previous page is right-to-left)
- Parser limits and parser report parameters.
- Parser options.

See *Changing the Project Properties* for further information.

Note: Many default settings (such as the parser settings, project encoding and character assignments) are defined in the Natural preferences. For further information, see *Setting the Preferences*.

4 Choose the **Finish** button.

Importing a Project from a Version Control System

You use the standard Eclipse functionality to import a Natural project from a version control system.

Importing an Existing Natural Project into the Workspace

You use the standard Eclipse functionality to import a Natural project from a different workspace into the current workspace.

When you import a Natural project into your workspace or check it out from the repository of a version control system, the user ID on the **Runtime** page of the project properties may differ from your current user ID. On import or checkout, the user ID is always set to the default user ID and the password is set to blank. The default user ID is either the ID that you have used to log on to your operating system or - if already entered in the project wizard - the user ID that is persistently stored in Eclipse. It is your responsibility to enter the appropriate user ID and password for the defined server connection.

Types of Natural Projects

There are different types of Natural projects with different characteristics:

- **Regular Natural Project**
  This is the standard project type that is used to develop Natural applications. In the properties of each project, you can define other projects as project references (this is standard Eclipse functionality). If you do this, it is important that you also define the corresponding libraries in the referenced project as **steplibs** in the regular project.
- **Referenced Natural Project**
  A referenced Natural project is designed to hold common Natural source objects for use in the regular project. For example, common data structures or copycodes can be defined once in a referenced project in order to have these objects available for other projects. In the properties of the regular project, the referenced project must be defined as a project reference; the builder of the regular project then gets the information from the referenced project.

  Example: You have a library containing Natural subprograms which provide a set of general functionality. These subprograms are used in all of your projects, but you have not downloaded the corresponding sources into your workspace. Since the subprograms that are referenced in

your programs are not available to your projects, the **Dependencies** view marks them as "Unknown". To avoid this, you can download all subprograms once into a dedicated project in your Eclipse workspace. Let us call this project "ExternalObjects". In all other regular projects that make use of these subprograms, you can then define the project "ExternalObjects" as a referenced project (in the project properties). In addition, you have to define the library containing the subprograms as a steplib in all projects which make use of the referenced project. When this has been done, the **Dependencies** view no longer shows a subprogram as "Unknown". It now shows the name of the referenced project.

Changing the Project Properties

The Natural project properties are made up of information and profile parameters that are important for the development of a Natural application.

When a project is created using the **wizard**, the project properties are preset and can be changed in the wizard. The properties are then written as the default settings into the *.natural* file. For some properties, default values can be defined in the Natural **preferences**.

When a project is created by **downloading** a library or object from a Natural server into a new project, the properties from the server are written as the default settings into the *.natural* file.

The settings stored in the *.natural* file are the so-called “default settings”. They are important when you work with a version control system. When you change any settings in the project properties and just choose the **OK** or **Apply** button, these changes are stored persistently under the control of Eclipse, but they are not automatically written to the *.natural* file. The content of the *.natural* file is only changed when you do this explicitly by also choosing the **Store new Defaults** button (see the description of the **Natural** property page **below**).

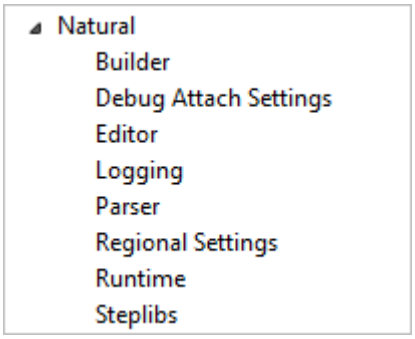
When you change the settings for a project, your changes are uploaded to the appropriate Natural environment the next time you **update** the Natural environment with an object in this project. The uploaded changes include the default settings in the *.natural* file, plus your user settings which are stored persistently under the control of Eclipse. In this case, the user settings always superimpose the default settings.

To change the project properties

1 Select the project in the **Navigator** view or in the **Natural Navigator** view.

2 Invoke the context menu and choose **Properties**.

3 In the tree of the resulting dialog box, expand the **Natural** node.



Different pages are provided for setting different types of properties.

Notes:

1. The **Debug Attach Settings** node is only shown when a debug attach server has been enabled in the Natural **preferences**.
2. The **Steplibs** node is only shown when the project pertains to a server environment which is not protected by Natural Security.

4 Select one of the property pages in the tree and set the required options as described in the topics below.

5 Choose the **OK** button to save your changes and to close the dialog box.

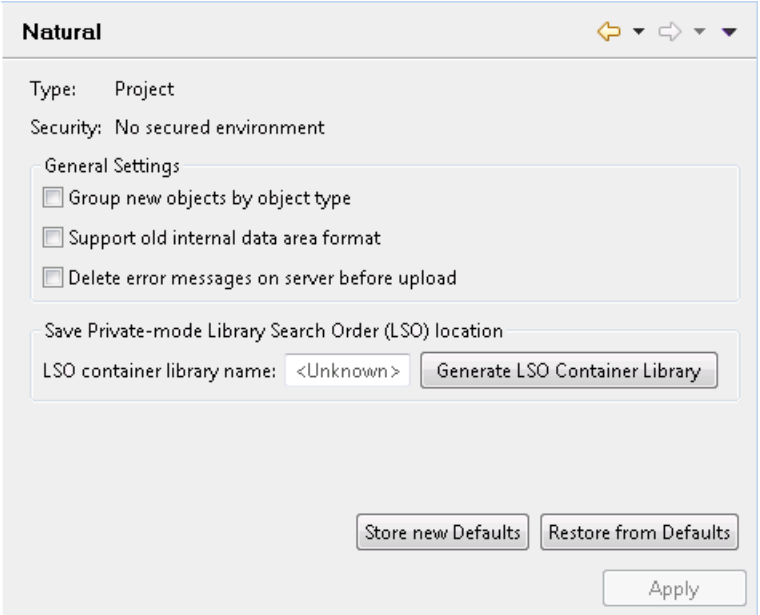
The following property pages are available for Natural projects:

- Natural
- Builder
- Debug Attach Settings
- Editor
- Logging
- Parser
- Regional Settings
- Runtime

- Steplibs

Natural

This property page is shown when you select the **Natural** node.



One of the following messages can be shown next to **Security**:

Message	Meaning
Environment is secured	This environment is protected by Natural Security.
No secured environment	This environment is not protected by Natural Security.

Note: The environment is defined on the **Runtime** page.

You can modify the following settings:

Group new objects by object type

If you want to group the Natural objects in this project into separate folders, according to their object types, make sure that this option is selected. For further information, see *Group Folders*.

Support old internal data area format

This option applies to the internal format of data areas in a Natural for Windows, UNIX and OpenVMS environment.

With Natural Version 6.1 for Windows and UNIX and Natural Version 6.3 for OpenVMS, a new internal format was introduced for data areas which supports, for example, dynamic and large variables.

When data areas are uploaded to the Natural environment, the new internal data area format is used by default. It is strongly recommended that you keep this default (that is, do not select this check box).

Data areas with the new format are not downward-compatible. Therefore, it is not possible to use them with Version 5.1 and below. Select this check box only, if you require data areas in the old format that are to be used with Natural Version 5.1 or below.

Note: The default setting for this option can be changed in the Natural preferences. See ***Natural > Options*** in *Setting the Preferences*.

Delete error messages on server before upload

When selected, all error messages are deleted in the appropriate library on the server before the error messages from the project are uploaded.

When not selected (default), all error messages are uploaded to the server. Any error messages which are no longer available in the project are not deleted on the server. This may cause inconsistencies.

See also ***Creating Application-Specific Messages***.

Note: The default setting for this option can be changed in the Natural preferences. See ***Natural > Options*** in *Setting the Preferences*.

Store new Defaults / Restore from Defaults

The *.natural* file contains the default settings of the Natural project properties. The following command buttons are available:

- **Store new Defaults**
  If you choose this command button, the current user settings of the project properties are stored as the new default values in the *.natural* file. A dialog appears in which you have to confirm the action.

  When you commit the new version of the *.natural* file to the repository of your version control system, the new default values are available to all developers involved in the project.

- **Restore from Defaults**
  If you choose this command button, the current user settings of the project properties are discarded. They are overwritten with the default values that are stored in the *.natural* file. A dialog appears in which you have to confirm the action.

  Since the *.natural* file can be versioned, any newly defined project settings can thus be used consistently by all developers involved in the project.

If the project uses private-mode libraries, and the library search order (LSO) container has not yet been created, the following command button is available:

- **Generate LSO Container Library**
  If you choose this command button, an LSO container library is created on the server and the name of the LSO container library name is shown. This name is required when running a Natural batch application in private mode. The naming convention for the LSO container library is the same as for a private-mode library defined in the Natural preferences. See ***Natural > Options*** in *Setting the Preferences*.

If the library search order (LSO) container has already been created, the following command button is available:

- **Delete LSO Container Library**
  If you choose this command button, the LSO container library is deleted.

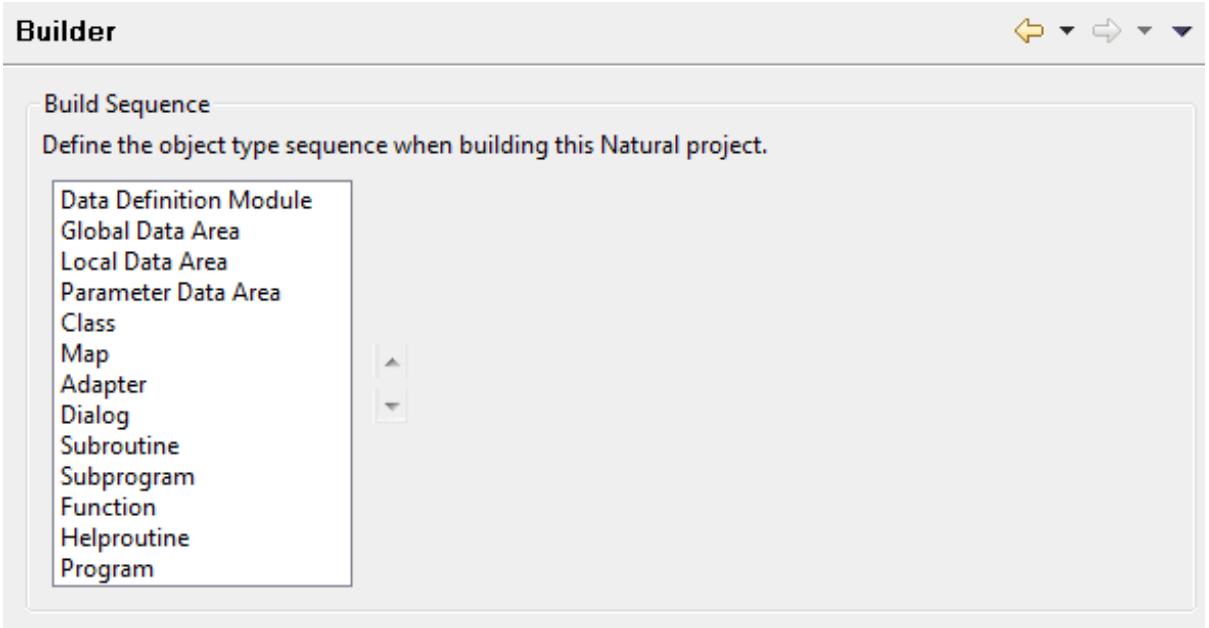
Notes:

1. Before you can use private-mode libraries in batch, you have to save the private-mode library search order. For more information, see *Using Private-mode Libraries in Batch* in *NaturalONE in a Nutshell*.
2. The **Store new Defaults** and **Restore from Defaults** buttons do not apply to the LSO container library settings.

Builder

This property page allows you to define a project-specific build sequence. This sequence is used by the **Build Natural Project** command to upload and stow objects in the Natural environment. If required, you can use the up-arrow and down-arrow buttons to change the build sequence for the current project.

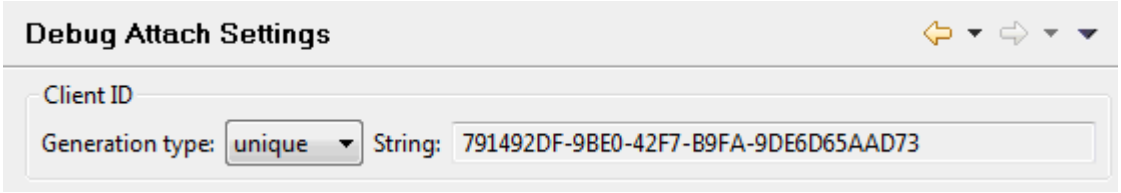
The default sequence for new projects is defined in the Natural preferences. See also ***Natural > Build Sequence*** in *Setting the Preferences*.



Debug Attach Settings

This property page (and its corresponding link in the tree) is only shown when a debug attach server has been enabled in the Natural preferences. See ***Debug Attach Settings*** in *Setting the Preferences*.

The debug attach server uses a client ID to manage its attach records. A new unique client ID is generated for each project each time you start NaturalONE. This client ID is shown on this property page.



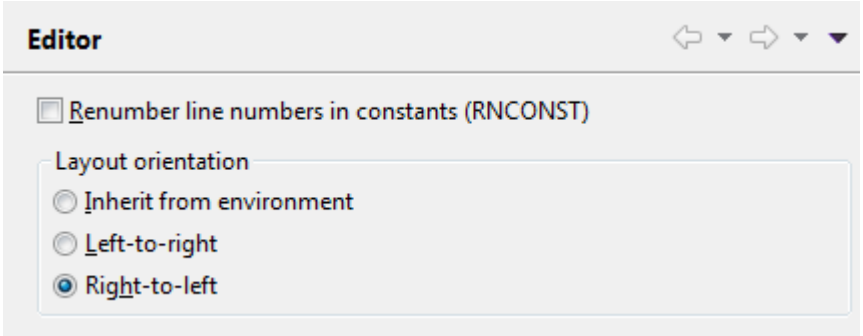
Generation type

It is recommended that you do not change the generation type. You should leave it with **unique**.

However, if you want to debug an external Natural application or, if necessary, for testing purposes, you can define a custom client ID. You should only do this if you want to start a particular Natural runtime server manually. In this case, select **custom** from the drop-down list box and specify a client ID in the text box.

Editor

This property page allows you to change editor settings.



You can modify the following Natural profile parameters on this page:

Option	Corresponding Natural Profile Parameter
Renumber line numbers in constants	`RNCONST`
Layout orientation	`PM=I`

For detailed information on these profile parameters, see the Natural documentation for the appropriate platform.

If **Renumber line numbers in constants** is enabled, the line number references in constants are automatically renumbered while you are editing a source and when uploading the source to the Natural server (irrespective of its `RNCONST` setting). See also *Line Numbers* for more information.

Note: The setting of the above option is only taken into account when a source is being opened. If you change the setting while a source is already open, you have to close and re-open the source so that the changed setting becomes active.

The **layout orientation** that you define with the following option buttons affects only the **Layout** page (that is, the graphical display) of the map editor and the **Report Data** view in the data browser.

- **Inherit from environment**
  When selected, the above components have no intrinsic orientation and simply inherit the orientation of the Eclipse environment.
- **Left-to-right**
  When selected, the above components are displayed from left to right (LTR), regardless of the orientation of the Eclipse environment.

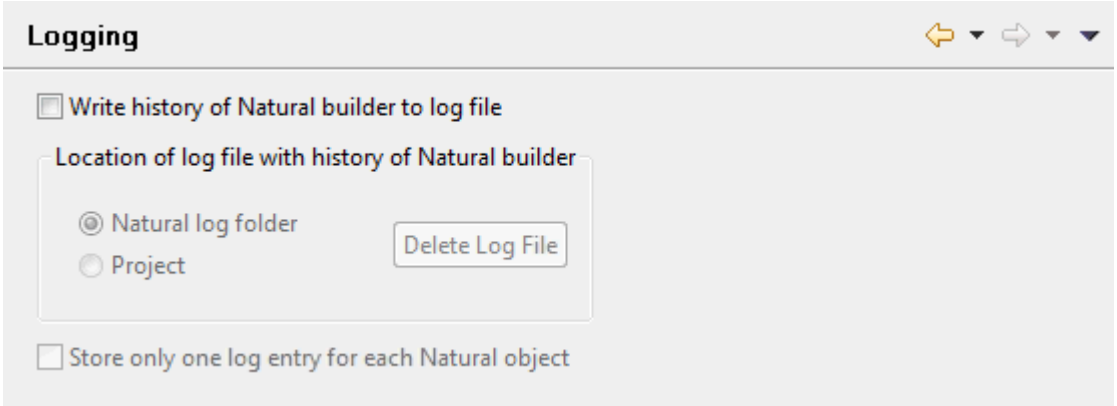
- **Right-to-left**
  When selected, the above components are displayed from right to left (RTL), regardless of the orientation of the Eclipse environment.

Note: The settings of the above options are only taken into account when a source is being opened. If you change the settings while a source is already open, you have to close and reopen the source so that the changed settings become active.

See also *Bidirectional Language Support*.

Logging

This property page allows you to write the history of the Natural builder to a log file.



Write history of Natural builder to log file

When this check box is enabled, all other options on this page are also enabled and information is written to a log file each time you change, delete or add a source. The log file is created as soon as the first action occurs which requires an update in the server environment. These are the actions for which flags will be shown in the label decorations when your projects are not automatically built (see also *Flags in a Label Decoration*).

New history information is appended to the log file. As long as you do not enable or disable the option **Store only one log entry for each Natural object** (see below), the content of the log file is not deleted.

The following is an example of a log file:

```
[2012/03/29][10:25:14.817],MYLIB-A,COPY1,C,SAVE
[2012/03/29][10:25:14.817],MYLIB-A,PROG1,P,CAT
[2012/03/29][10:25:27.330],MYLIB-B,LDA1,L,STOW
[2012/03/29][10:25:27.330],MYLIB-A,PROG1,P,CAT
[2012/03/29][10:25:27.330],MYLIB-A,PROG3,P,CAT
[2012/03/29][10:25:35.641],MYLIB-A,PROG1,P,STOW
[2012/03/29][10:25:40.072],MYLIB-B,HLPRT2,H,STOW
[2012/03/29][10:25:40.072],MYLIB-A,PROG3,P,STOW
[2012/03/29][10:25:46.945],MYLIB-B,SUBP1,N,SCRATCH
```

For each operation, the log file provides the following information, separated by commas:

- timestamp in the format [yyyy/mm/dd][hh:mm:ss.SSS]
- library name
- object name
- one-letter abbreviation for the object type (for example, `P` for a program)
- type of operation (`SAVE`, `CAT`, `STOW` or `SCRATCH`)

Location of log file with history of Natural builder

Using the following option buttons, you can define where the log file is to be stored:

- **Natural log folder**
  When enabled (default), the log file is stored in your Eclipse workspace. When you look at your workspace in the file system, you can find it in the *.naturalone/log* folder. The name of the log file is *<project-name>_builderHistory.log*.
- **Project**
  When enabled, the log file is stored in the root of the current Natural project. You can find it in the **Navigator** view or in the **Natural Navigator** view. The name of the log file is *builderHistory.log*.

When a log file exists at the selected location (either in the Natural log folder or in the project), the **Delete Log File** button is enabled. You can use this button to delete the log file at the selected location. When you choose this button, a dialog appears, asking whether you want to delete the log file. It is not possible to undo the deletion.

Store only one log entry for each Natural object

Caution: When you change the setting of this option and if a log file exists at the selected location, a dialog box appears, informing you that the current contents of the log file will be deleted. You are asked whether you want to continue.

When enabled, each Natural object is listed only once in the log file. The log file is written in the same format as described above. However, its sort sequence is different. The entries are no longer sorted according to the timestamp (that is, new information is not always appended at the end of the file). Instead, the following sort sequence is used: library name, object type, object name. This is similar to the sort sequence which is used by `CATALL`.

The following is an example of such a log file:

```
[2012/04/05][10:26:52.633],MY_LIB1,SUBPGM2,N,STOW
[2012/04/05][10:27:03.345],MY_LIB1,PGM1,P,STOW
[2012/04/05][10:26:56.954],MY_LIB1,PGM2,P,STOW
[2012/04/05][10:26:48.930],MY_LIB2,PGM2,P,STOW
[2012/04/05][10:26:58.954],MY_STEP1,CPYCDE1,C,SAVE
[2012/04/05][10:26:50.930],MY_STEP1,LDA1,L,STOW
[2012/04/05][10:27:07.126],MY_STEP1,SUBROUT1,S,STOW
[2012/04/05][10:27:03.345],MY_STEP2,CPYCDE2,C,SAVE
```

The content of the log file can be seen as a compilation of all actions which are needed to deploy updates. Each time an action is performed on an object, either a new entry is created in the log file or the stored entry is updated.

For example, when the log file already contains a `SAVE` entry for an object and a `CAT` entry is later added for the same object, both log entries are automatically combined into a single `STOW` entry (which logically contains the `SAVE` and `CAT` operations).

Or, when a `STOW` operation occurs several times for the same object, only the latest `STOW` operation (with the corresponding timestamp) is shown in the log file.

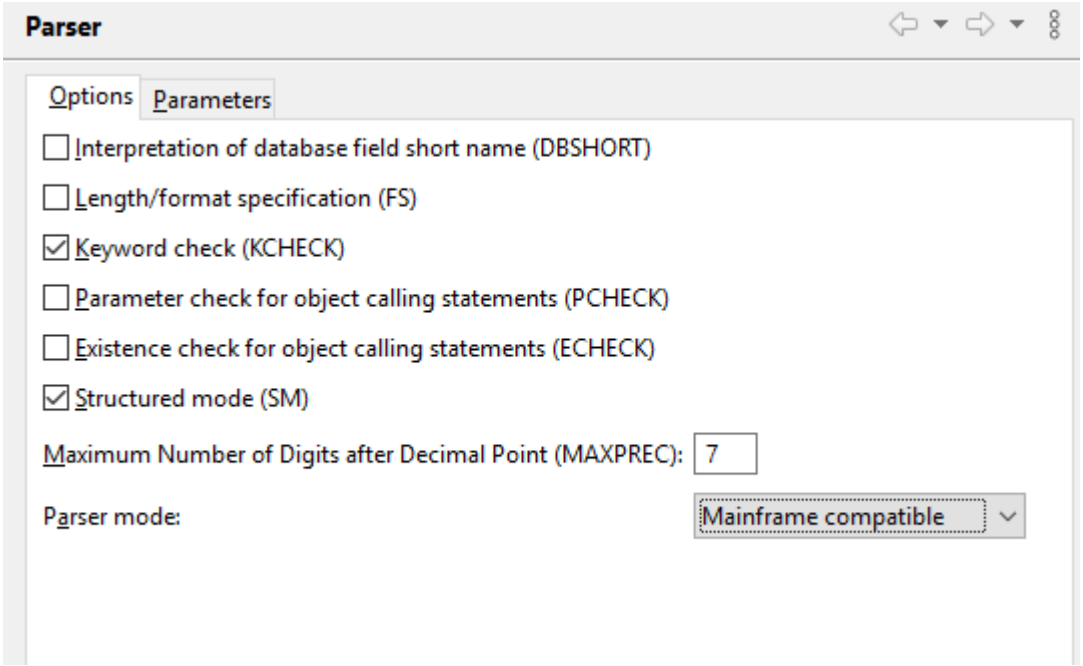
Parser

This property page provides the following tabs:

- Options
- Parameters

Note: If Natural Security is active, the parser settings can also be defined in the properties of a library. In this case, the library properties override the project properties. See *Changing the Library Properties*.

Options



On this tab, you can modify the following Natural profile parameters:

Option	Corresponding Natural Profile Parameter
Interpretation of database field short name	DBSHORT
Length/format specification	FS
Keyword check	KCHECK
Parameter check for object calling statements	PCHECK
Existence check for object calling statements	ECHECK
Structured mode	SM
Maximum Number of Digits after Decimal Point	MAXPREC

For detailed information on these profile parameters, see the Natural documentation for the appropriate platform.

The **Parser mode** drop-down list box, which is also provided on this tab, allows you to define the platform for which the Natural language syntax is to be checked. You can select one of the following options:

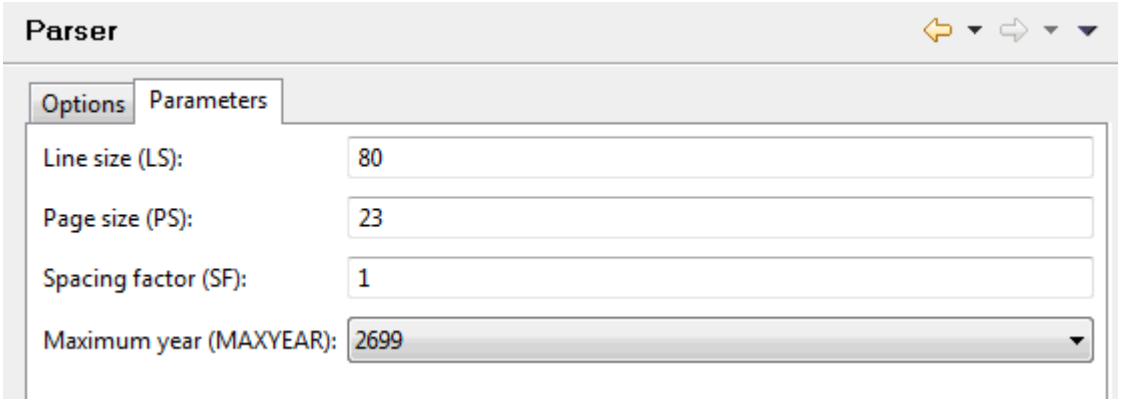
- **Mainframe compatible**
  Natural sources are checked according to the Natural syntax for mainframe platforms.

- **Open systems compatible**
  Natural sources are checked according to the Natural syntax for Windows, UNIX and OpenVMS platforms.
- **Error tolerant**
  This is the default setting. When the Natural sources are checked, all valid Natural syntax is accepted, no matter for which platform you are currently developing. An error will occur only if invalid Natural syntax is found which does not apply to any of the supported platforms.

  For example, if you are developing for UNIX platforms and your code contains special syntax which is only valid for mainframe platforms (such as the SQL extended set for DB2 databases), the parser will *not* consider this as an error.
- **Platform compatible**
  When the Natural sources are checked, not all Natural syntax is accepted. The parser only accepts syntax which can be used on *all* supported platforms. An error will occur if platform-specific syntax is found (even if this is the correct syntax for a specific platform).

  For example, if your code contains special syntax which is only valid for mainframe platforms (such as the SQL extended set for DB2 databases), the parser will consider this as an error.

Parameters



On this tab, you can modify the following Natural profile parameters:

Option	Corresponding Natural Profile Parameter
Line size	LS
Page size	PS
Spacing factor	SF
Maximum year	MAXYEAR

For detailed information on these profile parameters, see the Natural documentation for the appropriate platform.

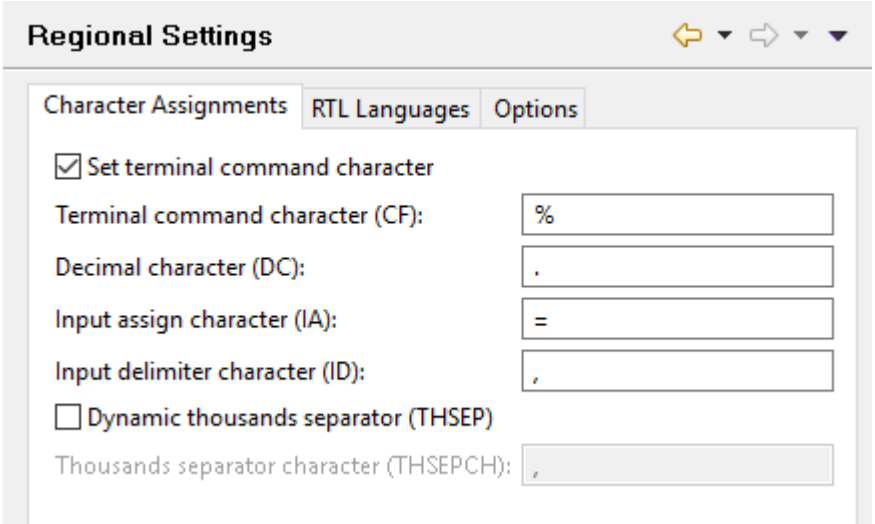
Regional Settings

This property page provides the following tabs:

- Character Assignments
- RTL Languages

- Options

Character Assignments



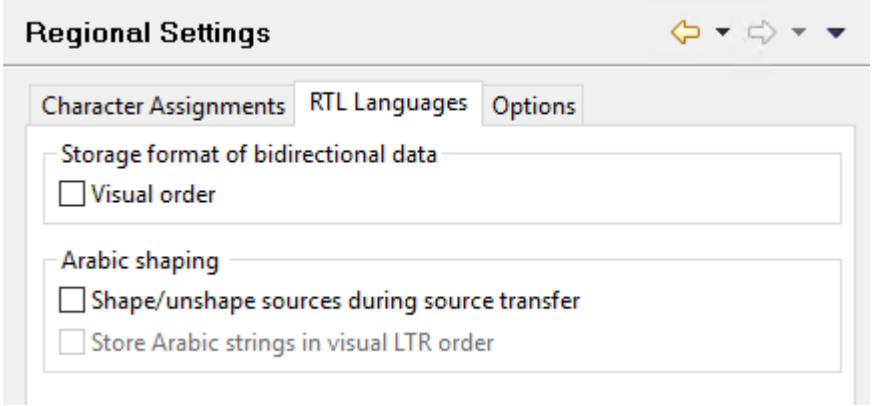
On this tab, you can modify the following Natural profile parameters:

Option	Corresponding Natural Profile Parameter
Set terminal command character	CF
Terminal command character	
Decimal character	DC
Input assign character	IA
Input delimiter character	ID
Dynamic thousands separator	THSEP
Thousands separator character	THSEPCH

For detailed information on these profile parameters, see the Natural documentation for the appropriate platform.

RTL Languages

On this tab, you can specify the default settings for languages that are written from right-to-left (RTL).



Visual order

This option is intended to support data that is stored in visual order (rather than in the usual logical order), in order to appear correctly on terminals that are not aware of bidirectional languages. Because modern GUI environments do the transformation from the logical to the visual character sequence implicitly, NaturalONE needs to know (via this option) whether the data is stored already in reordered form (that is, in visual order) so that it can be converted back into logical order in internal storage. Otherwise, due to the reordering performed by the GUI, the data would effectively be reordered twice.

When selected, the application data (Natural sources and data from databases) is assumed to be in visual order. When deselected (default), the data is assumed to be in logical order.

This option is evaluated for the following data:

- Text constants in the **map editor**.
- Natural error messages displayed in the **error message editor**.
- Variable data in the **debugger**.
- Alphanumeric fields displayed with the **data browser**.
- Header and edit mask columns of the **DDM editor**.

See also *Bidirectional Language Support*.

Shape/unshape sources during source transfer

When selected, all sources are unshaped when they are added to the project and shaped when they are uploaded to the server.

See also *Arabic Shaping*.

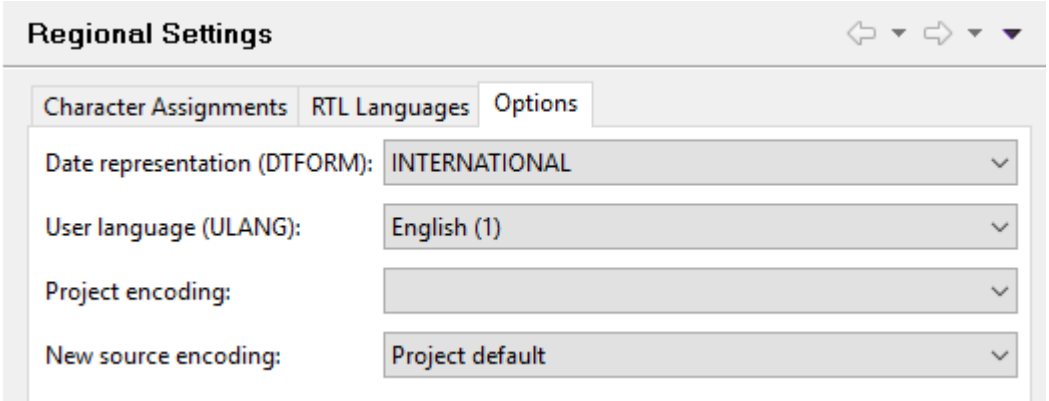
Store Arabic strings in visual LTR order

Only enabled when the **Shape/unshape sources during source transfer** check box is selected.

When selected, it is assumed for the shaping conversion that the Arabic strings are stored in visual left-to-right (LTR) order. This means that the final character of the string is the first character in storage and the initial character of the string is the last character in storage.

When not selected, it is assumed that the Arabic strings are stored in logical order. This means that the initial character of the string is the first character in storage and the final character of the string is the last character in storage.

Options



On this tab, you can modify the following Natural profile parameters:

Option	Corresponding Natural Profile Parameter
Date representation	DTFORM
User language	ULANG

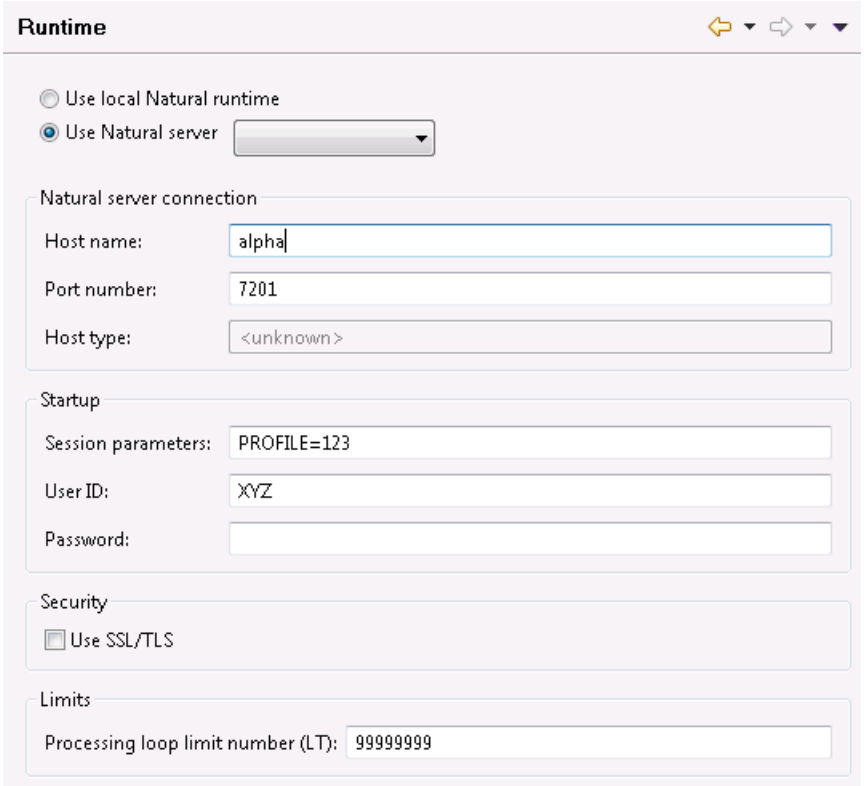
For detailed information on these profile parameters, see the Natural documentation for the appropriate platform.

The **Project encoding** drop-down list box, which is also provided on this tab, provides for selection all code pages that are valid for the Natural environment (Natural server or local Natural runtime) which is currently defined on the **Runtime** page of the project properties. When set to blank, a code page is not defined for the project. When the Natural environment is updated, the default code page defined in this environment will be used. New sources will be created using the code page that is defined for the project. In this case, any features and checks that are code page-dependent will be disabled.

The **New source encoding** drop-down list box determines the code page that is used for newly-created sources. When set to "Blank", no code page is explicitly set for these sources. When the Natural environment is updated, the default code page defined in this environment will be used. When set to "Project default", new sources will be created using the code page that is defined above for the project.

Runtime

This property page shows the mapping information for the Natural environment (Natural server or local Natural runtime) to which this project belongs. This information is used, for example, when you execute an object.



For information on the Natural server connection and startup options on this property page, see *Mapping a Natural Environment*.

The **Use local Natural runtime** option button is only visible when the local Natural runtime has been installed. When the local Natural runtime is selected, host name and port number are automatically provided and the corresponding text boxes appear gray.

If you want, you can assign your project to a different Natural environment. If you do so, it is likely that the security status of the project no longer matches the security status of the new Natural environment. Therefore, a dialog box appears which allows you to change or keep the current security status:

- When you change the Natural environment from a secured environment (protected by Natural Security) to an unsecured environment or vice versa, a dialog box appears, asking whether you want to change the security status of the project (for example, from secured to unsecured). In addition, when you decide to change the security status, you can adapt the properties for all libraries so that they match the properties of the new environment:

  - When you change to an unsecured environment, you can specify that the existing security properties are to be removed from all Natural libraries.
  - When you change to a secured environment, you can specify that properties are to be added to all Natural libraries with the security settings from the secured environment.
- When you change from a secured environment to a different secured environment, a dialog box appears asking whether you want to update the properties of all Natural libraries with the security settings from the new environment.

You can modify the following Natural profile parameter on this page:

Option	Corresponding Natural Profile Parameter
Processing loop limit number	LT

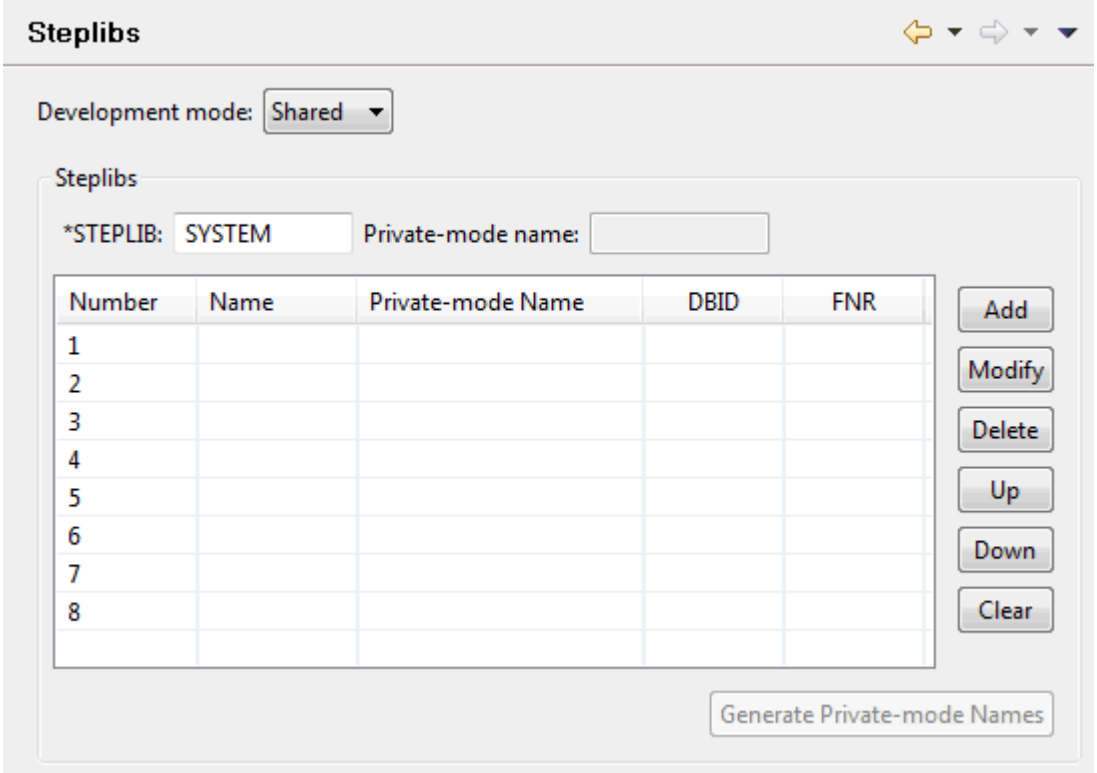
For detailed information on this profile parameter, see the Natural documentation for the appropriate platform.

Note: If Natural Security is active, this profile parameter can also be set in the properties of a library. In this case, the library properties override the project properties. See *Changing the Library Properties*.

Steplibs

This page is only available when the project pertains to a server environment which is *not* protected by Natural Security. If the server environment is protected by Natural Security, you can only define steplibs for a library; see *Changing the Library Properties*.

On this property page, you can define steplibs and the development mode in which the build is to be performed on the Natural server.



Development mode

When the Natural server is updated (see ***Updating the Objects in the Natural Environment***), all new and changed sources of a project are uploaded to the server and are stowed there. The development mode that you define determines the target libraries on the server. Two modes are available:

- **Shared Mode**

  This is the default mode. The sources are uploaded to libraries which have the same names as in the Eclipse workspace and are stowed in these libraries.

  Shared mode has the disadvantage that unpredictable results may occur if two users update the same sources on the same server at about the same time. In the best case, the update initiated by the second user overwrites the updates made by the first user, and the subsequent stow process thus considers only the changes made by the second user.

  When your original sources are stored on the Natural server (and not in the repository of a version control system), shared mode must be active for production.

- **Private Mode**

  When several users work with the same libraries, it is recommended that you use private mode for development and testing. With this mode, private-mode libraries are automatically created on the Natural server. They remain there persistently. For more information, see *Private-mode Libraries* in *NaturalONE in a Nutshell*.

The name of each private-mode library has a prefix which is defined in the Natural preferences (see ***Natural > Options*** in *Setting the Preferences*). In the Natural preferences, it is also possible to define a label decoration for a library in the workspace which displays the name of the associated private-mode library (see ***Label Decorations*** in *Setting the Preferences*).

The sources are uploaded to private-mode user libraries which belong only to one user; they are not shared by other users. The resulting cataloged objects are also stored in these private-mode user libraries. This has the advantage that each user can develop and test an application without affecting the sources of other users.

Exception: DDMs can only be uploaded to private-mode user libraries if they are stored in libraries in the server environment. This is the case in UNIX, OpenVMS and Windows environments when the `FDDM` parameter has not been set. DDMs which are not stored in libraries in the server environment are always uploaded to their original locations. It is not possible to upload them to private-mode user libraries. These are the DDMs which are stored in the `FDIC` system file (mainframe) and in the `FDDM` system file (UNIX, OpenVMS and Windows when the `FDDM` parameter has been set).

In addition to the private-mode user libraries, a private-mode steplib can be created for each steplib which exists in the current project. The search sequence is changed in such a way that each private-mode steplib is searched before the corresponding original steplib is searched.

You can define your own names for the private-mode steplibs or you can create the names automatically. See below.

After a successful test, the changes can either be stored on the server (in shared mode) or they can be committed to the repository of a version control system, depending on the location where your original sources are stored.

- **Private Mode Libraries**
  It is not intended to use private-mode libraries outside of a project. For this reason, it is not possible to change a private-mode library in the **Natural Server** view, and it is not possible to change any object in a private-mode library (for example, it is not possible to edit such an object).

  A dialog will appear in the following cases, asking whether you want to delete private-mode libraries in the runtime environment currently associated with a project:

  - when you switch from private mode to shared mode in the project properties or library properties,
  - when a project makes use of private-mode libraries and you switch to a different runtime environment via the **Runtime** property page,
  - when you delete a library inside a project which has an associated private-mode library or
  - when you delete a project from the workspace which makes use of private-mode libraries.

Steplibs

A steplib is a Natural user library or system library that is concatenated with the current user or system library. This avoids redundant storage of identical objects and helps organize applications. Natural searches in a steplib when an object is not found in the current library. The standard steplibs are the libraries `SYSTEM` in the system files `FUSER` and `FNAT`. The additional steplibs that you define on this property page are searched for an object before the standard steplibs.

- ***STEPLIB**
  This option corresponds to the Natural profile parameter `STEPLIB` which specifies the initial setting for the system Natural variable `*STEPLIB`. You can specify any valid library name.
- **Steplib Table**
  You can define up to eight steplibs in the steplib table. The steplib number is shown in the first column and cannot be changed. The search sequence is determined by the steplib number (number 1 is searched first).

  When you add or modify a steplib, you specify the following information:

Option	Description
Library name	The name of an existing library.
Private-mode name	Only available in private mode. Optional. The name of the private-mode steplib that is used in addition to the existing steplib. Private-mode names can only be generated for libraries which exist in the current project. When you add or modify a steplib and the library name you specify cannot be found in the current project, the text box is dimmed and it is thus not possible to enter a private-mode name. On the other hand, when you modify a steplib and if a private-mode name has already been generated, the generated name is shown in a dimmed text box.
DBID	The database ID of the system file in which the library is located.
FNR	The file number of the system file in which the library is located.

The following command buttons are available:

Command Button	Description
Add	Add a new steplib. Alternative: double-click any empty line. A new steplib is always added in the next empty line. It is not possible to add more than 8 steplibs.
Modify	Modify the library name, private-mode name, DBID and/or FNR for the selected steplib. Alternative: double-click a defined steplib.
Delete	Delete the selected steplib. Any defined steplibs after the deleted steplib are automatically moved up in the table and thus receive a different steplib number.
Up	Move the selected steplib to a higher steplib number (in this case, the steplib is moved down in the table).

Command Button	Description
Down	Move the selected steplib to a lower steplib number (in this case, the steplib is moved up in the table).
Clear	Delete all steplib entries.
Generate Private-mode Names	Only available in private mode. Generates a private-mode name for each steplib for which a private-mode name has not yet been defined. The generated name includes the prefix which is defined in the Natural preferences; see *Natural > Options* in *Setting the Preferences*.

Quickly Viewing the Properties

The **Properties** view shows enhanced, Natural-specific information for a node which is currently selected in the **Natural Navigator** view. Such information is not shown when working with the standard **Navigator** view of Eclipse.

The information that is shown in the **Properties** view depends on the type of node that is currently selected in the **Natural Navigator** view. It is very similar to the information that is shown when using the **Properties** command. But it also provides additional information such as the number of objects in a node.

When accessed via the **Natural Navigator** view, the **Properties** view provides different tabs. You can immediately see the settings for a selected project, library or object. For example, you can see the following information at a glance:

- for a project: whether is it controlled by Natural Security, or whether new objects are to be grouped by type,
- for a library: the development mode, and whether a private-mode name is used,
- for an object: the different names (object name, file name, and, if appropriate, long name), and the programming mode.

In addition, you can quickly compare the settings for two different nodes. For example, when you display the **Parser** tab for one library and then select a different library in the tree, the **Parser** tab is also shown for the newly selected library.

Enabling a Project for NaturalONE

You can use any kind of project (for example, a Java project) with NaturalONE. However, you have to enable it first.

Natural projects which have been created with Natural for Eclipse cannot be used immediately with NaturalONE. If you have imported such a project into NaturalONE or if you have checked it out from your version control system, you first have to enable it for NaturalONE. A Natural project which has not yet been enabled for NaturalONE is indicated by the following icon: .

To enable a project for NaturalONE

1 In the **Navigator** view or in the **Natural Navigator** view, select the project that you want to enable.

2 Invoke the context menu and choose **Enable for NaturalONE**.

Note: This command is only visible when you select a project which has not been created with NaturalONE.

6 Managing Libraries in a Natural Project

- Libraries in a Natural Project 68
- Creating Libraries 75
- Changing the Library Properties 78
- Updating the Library Properties with the Settings from the Server 83
- Copying, Pasting, Moving, Renaming and Deleting Libraries 83

Libraries in a Natural Project

Each Natural project can contain one or more Natural libraries. A Natural library is indicated by a green-colored folder icon: .

Other than on the Natural server, a Natural library in the Eclipse workspace contains only the sources. Generated programs (which are also called "GPs" or "cataloged Natural objects") are not stored in the Eclipse workspace. These are **executed** or **debugged** directly in the appropriate Natural environment.

In the Eclipse workspace, it is possible to store your Natural objects in arbitrary folder structures. This enables you to organize your objects in a more logical way than with the standard library structure that is mandatory on the Natural server. For example, you can group your objects according to the different tasks that are covered by your application.

Notes:

1. Keep in mind that the **Navigator** view shows the objects as they are physically stored in the file system, whereas the **Natural Navigator** view allows you to view the objects in a logical way.
2. For an overview of the icons and file extensions that are used for the different types of objects in the **Navigator** view or in the **Natural Navigator** view, see *Types of Natural Editors*.

On the Natural server, the Natural library name and the objects in a library must meet the Natural naming conventions (see the Natural documentation for the appropriate platform for further information). However, in the Eclipse workspace, it is possible to use alternative folder and file names for the libraries and objects.

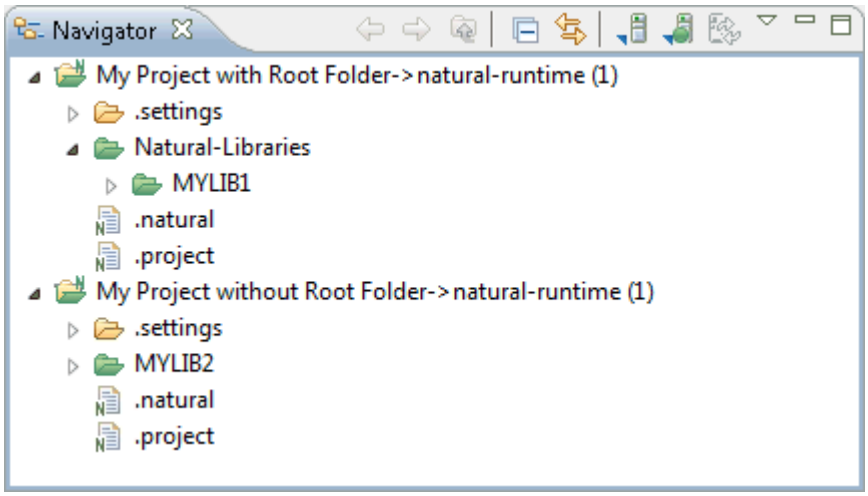
More detailed information is provided in the topics below:

- Library Root Folder
- Special Folders in a Library
- Library Folders
- Group Folders
- File Names

- Label Decorations

Library Root Folder

A Natural project may contain a library root folder which has the fixed name "Natural-Libraries".



Library root folders can be created when the **Root folder support** option is activated in the Natural preferences (see ***Natural > Project*** in *Setting the Preferences*). When you create a new Natural project (either by using the **wizard** or by **downloading** objects into a new project), you can then define whether the library root folder is to be created, or not. To do so, you simply have to make sure that the **Create the library root folder** option is activated. The state of this option is stored and reused the next time you create a new project.

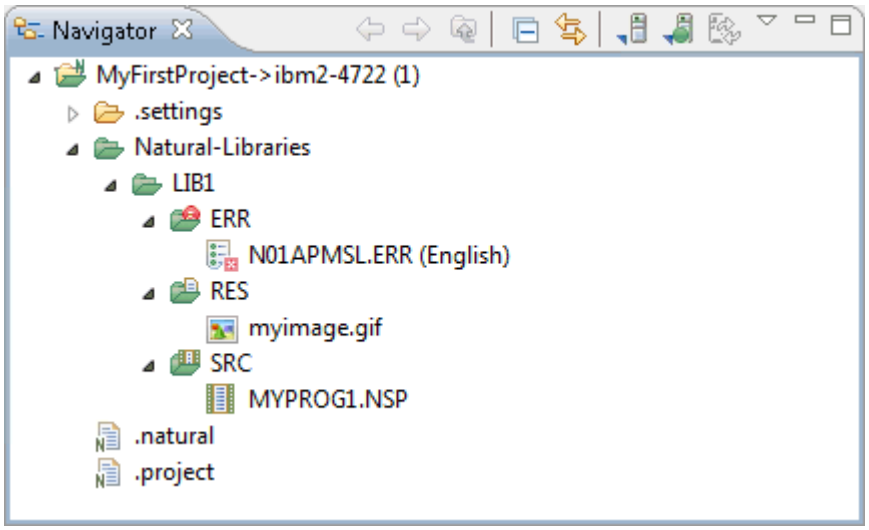
When an existing project does not yet contain a library root folder, you can also create the library root folder by adding a “normal” folder (**File > New > Other > General > Folder**) in a Natural project and naming this folder "Natural-Libraries". With this name, the folder is considered to belong to Natural and its icon is shown with a green color. Next, you have to move all existing libraries which are still located outside this new library root folder into this new library root folder.

When a project contains the library root folder "Natural-Libraries", any libraries that are located outside this root folder are ignored when the project is built. When a library is located outside the library root folder, its icon is shown with the standard color for folders, which is yellow.

When a project does not contain a "Natural-Libraries" folder, all libraries are stored directly below the project node, and they are considered when the project is built.

Special Folders in a Library

A Natural library may contain the special folders *SRC*, *ERR* and *RES*.



- ***SRC* - Source Folder**
  The source folder has the reserved name "SRC". It contains Natural sources (programs, maps, DDMs, etc.).

  Note: DDMs in mainframe environments have only long names. When downloading DDMs, the long names are automatically mapped to short names, if necessary.

- ***ERR* - Error Message Folder**
  The error message folder has the reserved name "ERR". It contains application-specific messages. See also *Creating Application-Specific Messages*.

- ***RES* - Resource Folder**
  The resource folder has the reserved name "RES". It contains resources in the Natural sense of the term, that is, non-Natural files (such as images or HTML files) that are used in a Natural application.

  Resources which belong to a Natural project can be altered using standard Windows or Linux tools, if available.

As long as these folders are stored within a Natural library, they cannot be renamed. However, when you move such a folder outside the library, it can be renamed as desired and can be used as a library of its own.

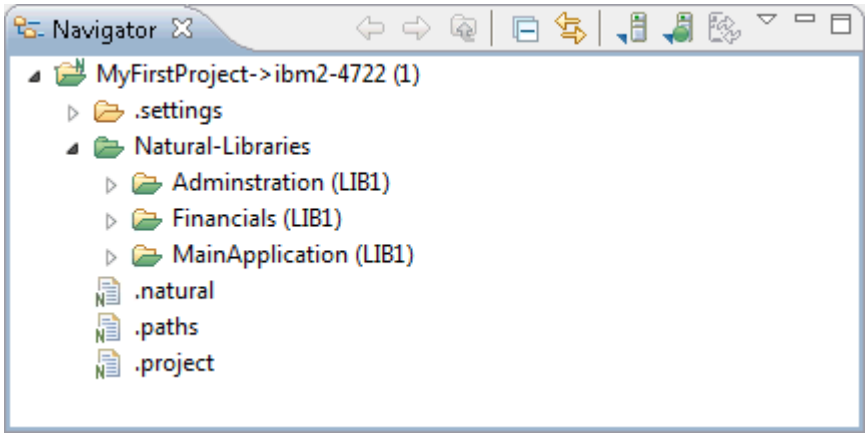
It is not mandatory to use the above mentioned folders. You can omit them altogether. By default, objects with the file extension *.NS** are handled as Natural sources, and objects which adhere to the naming convention for Natural error messages are handled as error messages. All other items are considered to be resources, in the Natural sense of the term.

Exception: If Natural sources or error messages are to be handled as resources, you have to create a subfolder in the library with the reserved name "RES" and move the objects into this subfolder.

Library Folders

A library folder is a folder that is assigned to a specific Natural library. For example, you can have multiple library folders for one and the same Natural library, where each library folder just contains the objects which pertain to a specific part of the application. When the "Natural-Libraries" root folder exists, this works only when the library folders are contained in this root folder.

A library folder is indicated by a green- and yellow-colored folder icon: .



Other than the library name which must adhere to the Natural naming conventions (for example, it may only be up to 8 characters long and must not contain lowercase characters), the name of a library folder may be up to 255 characters long. This can be any combination of uppercase and lowercase, but has to follow the rules of the underlying file system (Windows or Linux).

If you want to create a library and a library folder at the same time, you can do this as described in the section *Creating Libraries*.

You can also create a library folder by simply **renaming** a Natural library so that its name is no longer considered to be a valid Natural library name. When this is the case, the library name is modifiable in the **properties** of the library folder. This allows you to assign the library folder to a different library. If you rename a library folder in such a way that its name adheres again to the Natural naming conventions, the folder is no longer considered as a library folder; it is then considered as a regular Natural library (in this case, the library name is no longer modifiable in the library properties).

You can also create a “normal” folder (**File > New > Other > General > Folder**) in a Natural project and then assign it to a Natural library by specifying the library name in the folder properties. As long as a “normal” folder has not been assigned to a Natural library, the folder icon still has the standard color, which is yellow. When the folder has been assigned to a library, the color of the folder icon changes to green and yellow, indicating that it is now considered as a library folder.

When you create a subfolder within a Natural library or library folder, this subfolder automatically inherits the library from the parent folder. However, it is also possible to assign a different library to a subfolder.

Important: Even if the objects of one and the same library are stored in different library folders, all object names must be unique within a library. Keep this in mind, for example, when copying objects to a different library folder.

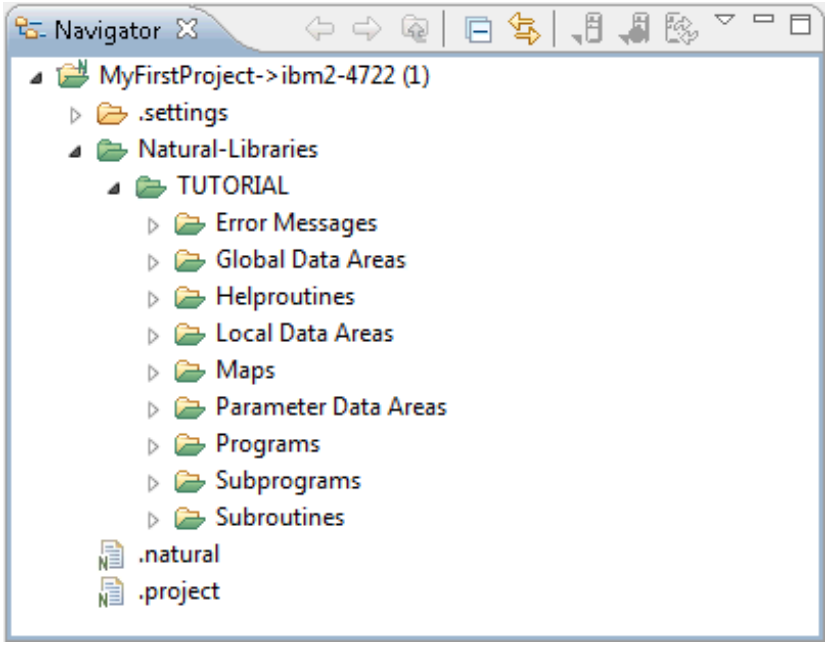
The library that you assign to a library folder need not necessarily be contained in the workspace. When it does not yet exist on the Natural server, it will be created the next time you upload your changes.

The library folders are only used in the Eclipse workspace. They are not uploaded to the server (for example, when you deploy your application). On the server, only the conventional Natural libraries are used. When the objects in a library folder are uploaded to the server, they are always placed into the libraries that have been assigned to the library folder.

Caution: As soon as a library folder is assigned to a library, a *.paths* file is created in the project. This file contains the mappings of the folder names to the Natural library names. You must not change the *.paths* file manually since this may result in damaging your project.

Group Folders

The Natural objects in a library can be grouped physically into separate folders, according to their object types. For example, all programs can be grouped into a folder named "Programs" and all adapters can be grouped into a folder named "Adapters".



This is similar to the **Natural Server** view where the objects in a library are automatically grouped into different nodes, according to their object types. However, other than in the **Natural Server** view where the objects are just grouped in a logical way, the objects in the workspace are grouped physically. This means, when an object in the **Navigator** view is shown within a folder named "Programs", this folder also exists in the file system (for example, *\MyFirstProject\Natural-Libraries\TUTORIAL\Programs*).

Note: If you do not want to group the objects physically, the **Natural Navigator** view offers the possibility to group them in a logical way. See ***Using the Natural Navigator View***.

If you want to make use of physical group folders, you have to select the **Group new objects by object type** option when you create a new project (either by using the **wizard** or by **downloading** an existing library or object from a Natural server into a new project). You can also select this option in the **project properties** of an existing project. When you create a **Natural object** or **error message**, the group folders are then automatically created.

Note: The default setting of the **Group new objects by object type** option when creating new projects can be determined in the **Natural preferences**.

The group folders are only created for existing objects. For example, when a downloaded library does not contain any subprograms, a folder with the name "Subprograms" is not created.

In addition to the group folders for the different Natural object types, it is also possible to have a group folder with the name "Resources". This folder is automatically created when you download resources (that is, non-Natural files such as images or HTML files) from a Natural server.

When you have chosen to use group folders, the special folders with the reserved names "SRC", "ERR" and "RES" are not created.

A group folder is the same as a library folder and is also indicated by a green- and yellow-colored folder icon: .

File Names

In the workspace, different types of names can be used for the Natural objects. In addition to the Natural object names which must adhere to the Natural naming conventions, you can define alternative (long) file names which may be up to 255 characters long (without the file extension). This can be any combination of uppercase and lowercase, but has to follow the rules of the underlying file system (Windows or Linux).

The file name must always be followed by the Natural file extension for the corresponding object type (such as *.NSP* or *.NSS*). All alphabetical characters in this file extension must be in uppercase.

Take care: It is also possible to create “long” file names which have less than 8 characters. This is the case, for example, when the name contains lowercase characters which are not allowed for Natural object names.

Important: Do not confuse the "long" file names with the long names of Natural objects (for example, of subroutines or DDMs). These are two different things.

The file name can either be specified when **creating** a Natural object or by **renaming** an existing Natural object. If an object has a file name, the original Natural object name can be modified in the **object properties**.

The file names are only used in the Eclipse workspace. They are not uploaded to the server (for example, when you deploy your application). On the server, only the conventional Natural object names are used.

Note: File names are not supported for error messages.

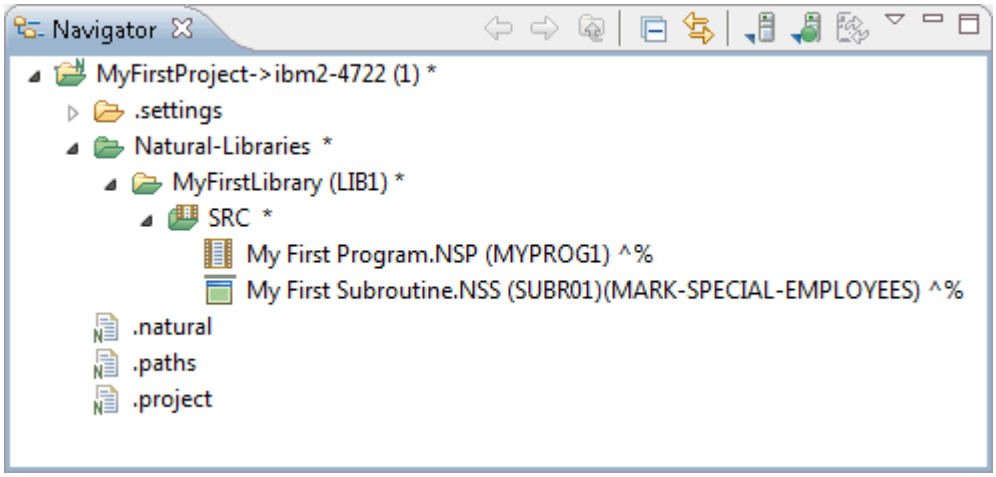
Label Decorations

You can define your individual label decorations for the different nodes in the **Navigator** view and **Natural Navigator** view. For example, you can define to show the object size, the long name and the code page with the name of an object. For information on the label decorations that can be used with a Natural project, see ***Label Decorations*** in *Setting the Preferences*.

By default, several defined names are shown in the label decoration of a library folder or object:

- For a library folder, the folder name is shown first, followed by the name of the Natural library, in parentheses, to which this folder has been assigned.
- For an object, the file name is shown first, followed by the Natural object name in parentheses. When a long name is available (for example, of a subroutine), it is additionally shown, also in parentheses.

Example:



Notes:

1. In the **Natural Navigator** view, the visibility of the folder names depends on your settings. See *Using the Natural Navigator View*.
2. NaturalONE also uses other label decorations. The visibility of the label decorations is controlled by the Eclipse preferences under **General > Appearance > Label Decorations**. If you do not want to see a specific type of label decoration, you can deselect the corresponding option in the Eclipse preferences.

Creating Libraries

In the Eclipse workspace, you can create a Natural library in different ways:

- by downloading a library or object from a Natural server, see *Downloading an Existing Library or Object from a Natural Server*,
- by creating a new library using a wizard, see below,
- by checking out a library from the repository of your version control system.

To create a new library using a wizard

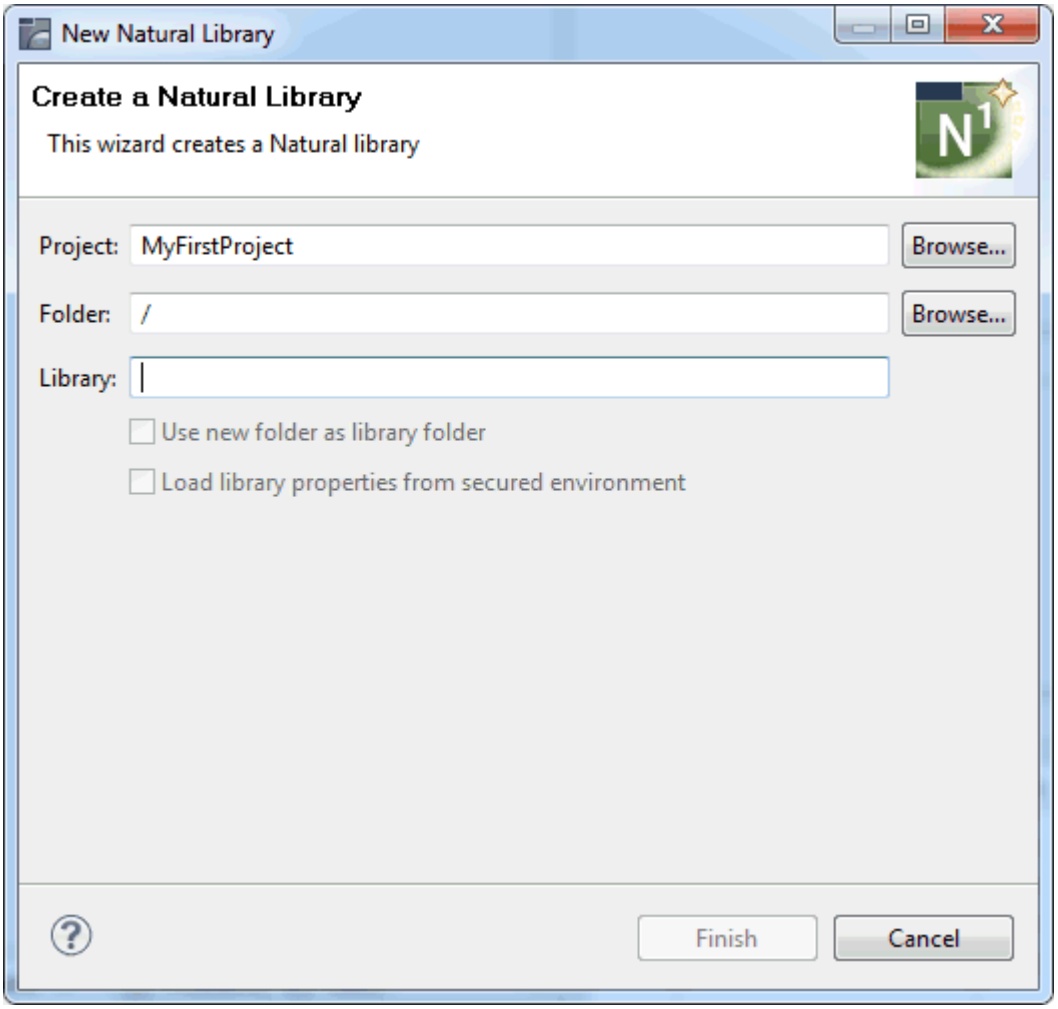
1 In the **Navigator** view or in the **Natural Navigator** view, select the Natural project in which you want to create the library.

2 From the **File** menu, choose **New > Natural Library**.

Or:

Invoke the context menu and choose **New > Natural Library**.

The following dialog box appears.



3 Specify the following information:

Option	Description
Project	The project in which the library is to be created. If you have selected a project before invoking this dialog box, the project name is automatically shown in this text box. However, if you have not selected a project (or if you have invoked the dialog box, for example, in a new empty workspace), you have to enter the name of a project to be created.
Folder	If you want to store your new library at a different location within the selected project, choose the **Browse** button. You can then select a library, library folder or even a subfolder such as *SRC*. See also *Library Folders*.
Library	A unique name for the new library. This name must adhere to the Natural naming conventions.
Use new folder as library folder	Only enabled when you have specified a folder name. If you want to create a library and a library folder at the same time, enable this check box. In this case, the new library is created as a library folder. The folder name you have specified is used as the name for the library folder.

Option	Description
	It is important that you specify the new folder name as shown in the following example: `/mylibfolder` Do not omit the slash.
Load library properties from secured environment	Only enabled when Natural Security is active in the associated Natural environment. When enabled, this check box is activated by default. In this case, the library properties for the newly created library are loaded from the server. If a corresponding library does not exist on the server, the properties from the logon library of that server are used. If the associated Natural environment is not accessible, a dialog box appears. In this case, you can either cancel or continue the operation. If you continue, the properties are set to the defaults of the selected project. When Natural Security is active and this check box is deactivated, the properties for the newly created library are set to the defaults of the selected project.

4 Choose the **Finish** button.

The new library or library folder is created in the specified project. When you have just created a library (and no library folder), the library contains empty folders for Natural sources, resources (that is, non-Natural objects) and application-specific messages (see also *Special Folders in a Library*).

You can now add Natural objects to the new library as described in *Creating Natural Objects*.

Notes:

1. If you do not need one of the empty folders (for example, the resource folder), you may delete it. If you need such a folder later, you can recreate it by selecting the library and then choosing **New > Other > General > Folder** from the **File** menu.
2. After having created the library contents, the **Upload** or **Build Natural Project** command generates this library on the appropriate Natural server during the upload process. See *Updating the Objects in the Natural Environment* for further information.

Changing the Library Properties

The information below applies to both libraries and library folders.

To change the library properties

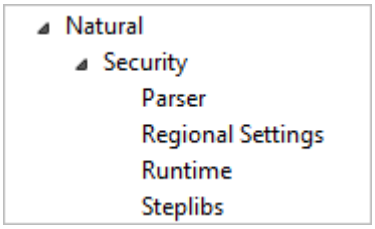
1 Select the library or library folder in the **Navigator** view or in the **Natural Navigator** view.

2 Invoke the context menu and choose **Properties**.

The **Properties** dialog box appears.

3 In the tree on the left side of the dialog box, expand the **Natural** node.

When Natural Security is active in the associated Natural environment, a **Security** subnode is shown. Information on the pages that are available for this subnode is provided in the topics below.



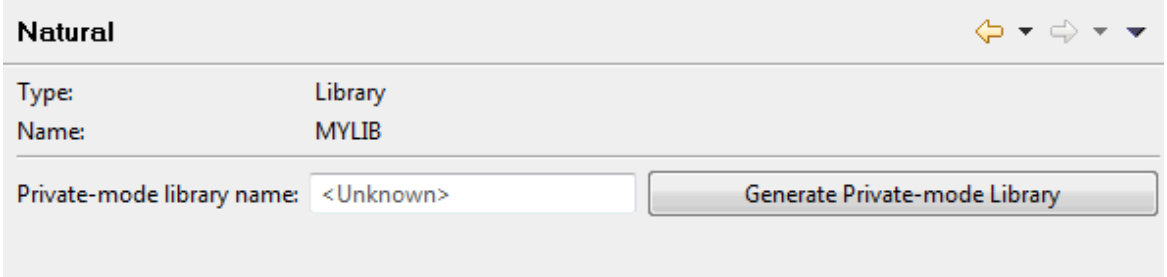
When Natural Security is not active, only the **Natural** node is shown, without any further subnodes.

Notes:

1. With Natural Security, steplibs and some profile parameters can be defined for each library. The library properties override the project properties. In the library properties, however, it is only possible to change the parameters which can also be changed with Natural Security in the Natural environment. All other parameters, which are shown in gray on the different property pages of the **Security** subnode, are inherited from the project properties and cannot be changed here.
2. If Natural Security is not active, it is not possible to specify any steplib information or profile parameters in the library properties. In this case, you can only define these properties for a **project**.

When you select **Natural** in the tree, the information that is shown depends on whether you have selected a library or a library folder in the **Navigator** view or in the **Natural Navigator** view.

- The following information is shown for a library:



The information on the private-mode library is only shown when private mode has been enabled. For each library in a project, you can create a private-mode library in the associated server environment. You can do this in one of the following ways:

- Leave the **Private-mode library name** text box empty and choose the **Generate Private-mode Library** button. The generated name for this library includes the prefix which is defined in the Natural preferences; see *Natural > Options* in *Setting the Preferences*.
- Enter the name for the library in the **Private-mode library name** text box. The name of the button then changes to **Create Private-mode Library**. Choose this button to create the private-mode library with the name you have just specified.

Once the private-mode library exists in the server environment, its name is shown in the **Private-mode library name** text box (which is read-only in this case) and the name of the command button changes to **Delete Private-mode Library**. When you choose this button, a dialog box appears, asking whether you want to delete the private-mode library in the server environment. When you delete the private-mode library and the server environment is protected by Natural Security, the corresponding entries in Natural Security's system file are also deleted.

- The following information is shown for a library folder. In this case, you have the possibility to assign the library folder to a different library.

See also *Renaming Objects and Libraries*.

4 Make all required changes.

5 Choose the **OK** button to save your changes and to close the dialog box.

Note: When a library is downloaded from Natural environment which is protected by Natural Security, the steplib information and a number of profile parameters are downloaded from the security profile which is located in the Natural environment and

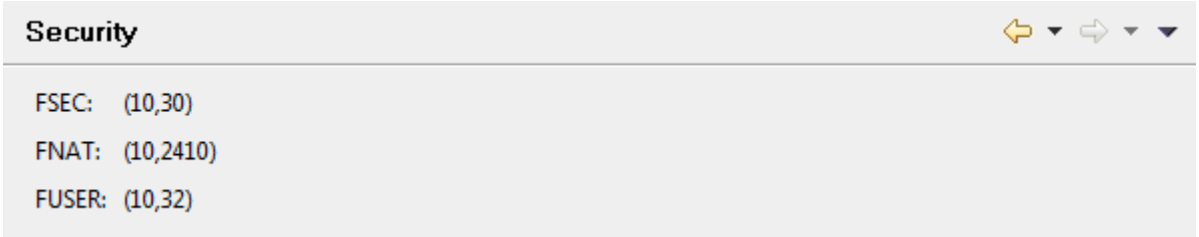
are saved in the Eclipse workspace. Any changes to the library properties are written to the *.natural* file. They are not written back to the security profile.

When you expand the **Security** node (which is only shown when Natural Security is active in the associated Natural environment), the following property pages are available:

- Security
- Parser
- Regional Settings
- Runtime
- Steplibs

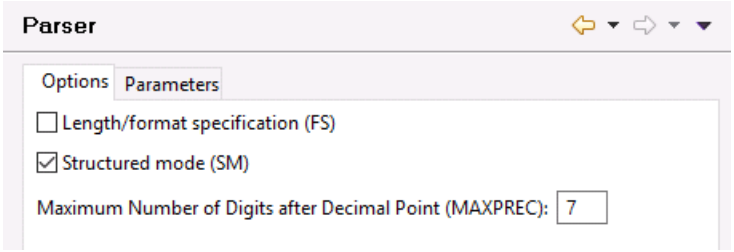
Security

When you select **Security** in the tree, the following property page is shown. It shows the settings for the `FNAT`, `FUSER` and `FSEC` system files. If information on a system file cannot be found, "unknown" is shown.



Parser

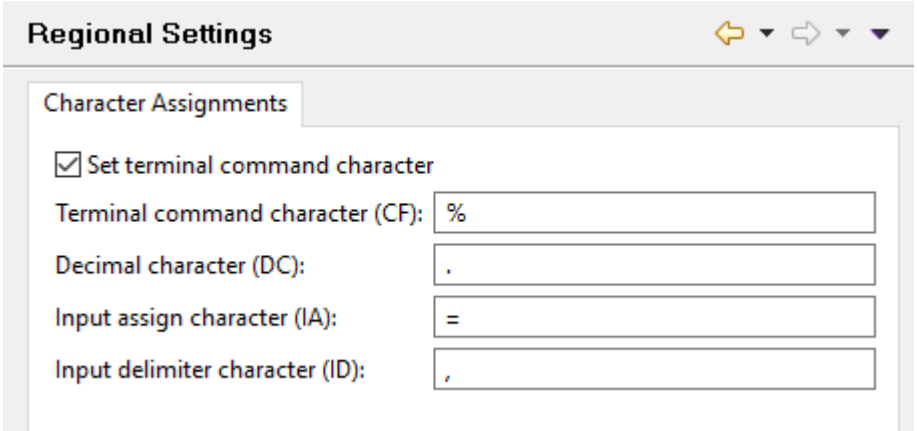
When you select **Parser** in the tree, the following property page is shown.



This property page provides different tabs. On these tabs, you can define the related profile parameters that are relevant for a library when Natural Security is active. More information on the individual options contained on these tabs is provided under ***Parser*** in the section *Changing the Project Properties*.

Regional Settings

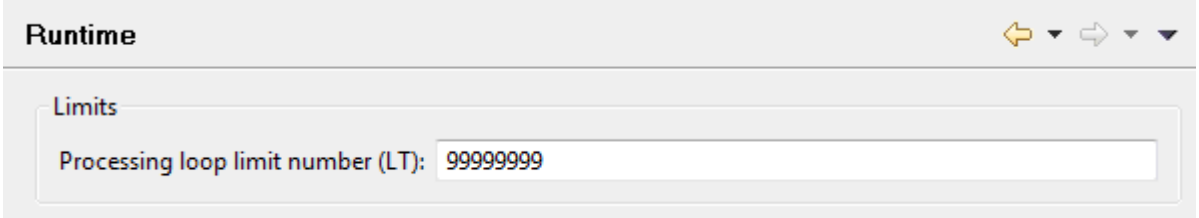
When you select **Regional Settings** in the tree, the following property page is shown.



This property page provides a **Character Assignments** tab. On this tab, you can define the related profile parameters that are relevant for a library when Natural Security is active. More information on the individual options contained on this tab is provided under ***Regional Settings*** in the section *Changing the Project Properties*.

Runtime

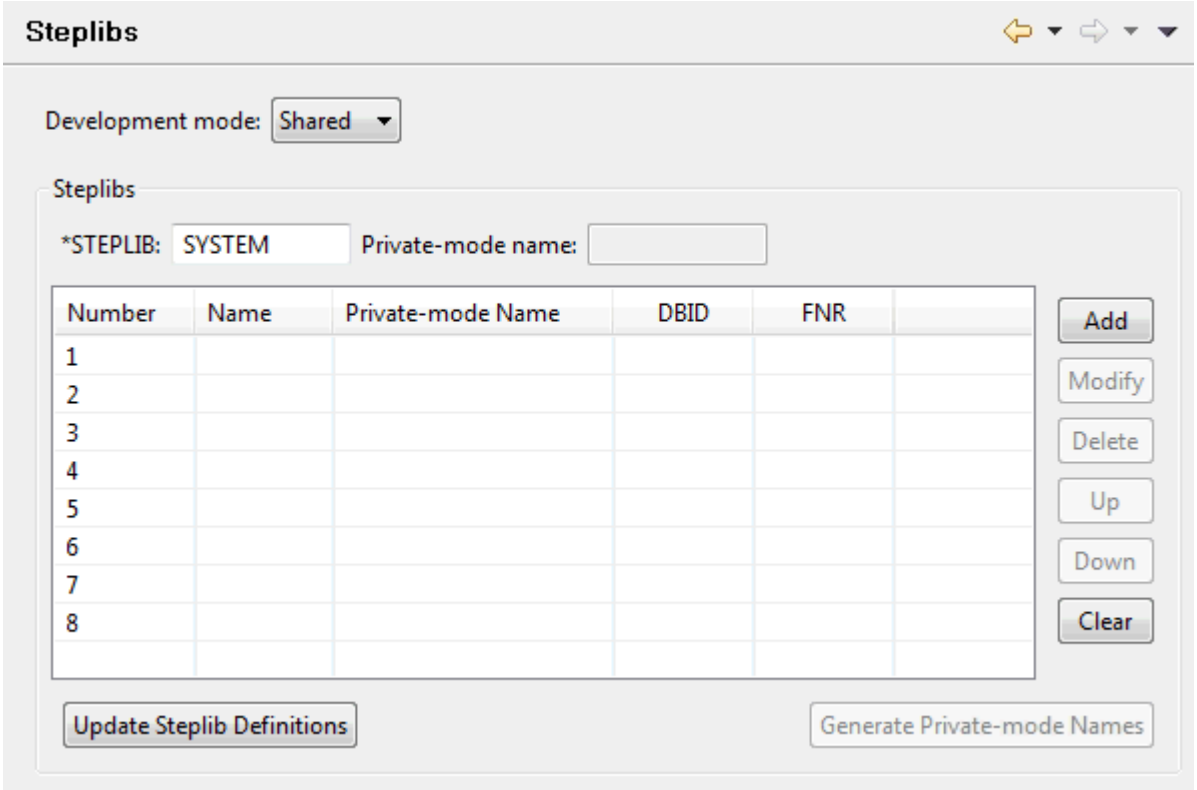
When you select **Runtime** in the tree, the following property page is shown.



On this property page, you can define the `LT` profile parameter which is relevant for a library when Natural Security is active. See also ***Runtime*** in the section *Changing the Project Properties*.

Steplibs

When you select **Steplibs** in the tree, the following property page is shown.



When Natural Security is active, steplibs can only be defined in the library properties. In this case, it is not possible to define them in the project properties.

It is possible to define different development modes for the individual libraries that are contained in a project.

When private mode is defined for the library, it is possible to define private-mode names for the steplibs which exist in the project.

For detailed information on the information that you can specify on this property page, see ***Steplibs*** in *Changing the Project Properties*.

The property page in the library properties provides the following command button which is not available in the project properties:

Update Steplib Definitions

When you choose this button, the steplib definitions in your local workspace are updated with the steplib definitions from the associated Natural environment.

When Natural Security is active, it is important that your local steplib definitions match those in the Natural environment. For example, when you update a project in a Natural environment which is protected by Natural Security, the local steplibs are first compared with those in the Natural environment. When they match, the project is updated. When they do not match (for example, when different names are used or additional steplibs are defined in the local workspace), a dialog box appears listing the local steplib names and the steplib names in the Natural environment. Using the buttons in this dialog box, you can either update the local steplib definitions automatically and continue with the update, or you can cancel the update.

Updating the Library Properties with the Settings from the Server

When the project containing a library pertains to a Natural environment which is protected by Natural Security, you can update the Natural profile parameters settings of a library in your workspace (such as the programming mode or the input delimiter character) with those which are currently active on the server. This is helpful, for example, when the Natural builder finds an error due to conflicting settings in the Eclipse and Natural server environments.

To update the library properties with the settings from the server

1 In the **Navigator** view or in the **Natural Navigator** view, select the library that you want to update

2 Invoke the context menu and choose **NaturalONE > Update Library Properties from Server**.

Note: This command is only visible when the library belongs to a project where the associated server environment is protected by Natural Security.

The settings from the server are now available in your **library properties**.

Copying, Pasting, Moving, Renaming and Deleting Libraries

You copy, paste, move, rename and delete Natural libraries in the same way as any Natural object. For detailed information, see the following topics:

- *Copying and Pasting Objects and Libraries*
- *Moving Objects and Libraries*
- *Renaming Objects and Libraries*
- *Deleting Objects and Libraries*

7 Managing Objects in a Natural Project

- Creating Natural Objects 86
- Changing the Object Properties 89
- Editing Objects 91
- Saving Objects 91
- Printing Objects 92
- Executing Objects 93
- Copying and Pasting Objects and Libraries 94
- Moving Objects and Libraries 97
- Renaming Objects and Libraries 98
- Deleting Objects and Libraries 102

Creating Natural Objects

In the Eclipse workspace, you can create a Natural object in different ways:

- by downloading a library or object from a Natural server, see *Downloading an Existing Library or Object from a Natural Server*,
- by creating a new object using a wizard, see below,
- by checking out an object from the repository of your version control system.

To create a new Natural object using a wizard

1 In the **Navigator** view or in the **Natural Navigator** view, select the Natural library in which you want to store the new object.

2 From the **File** menu, choose **New >** *`object-type`*, where *`object-type`* can be one of the following:

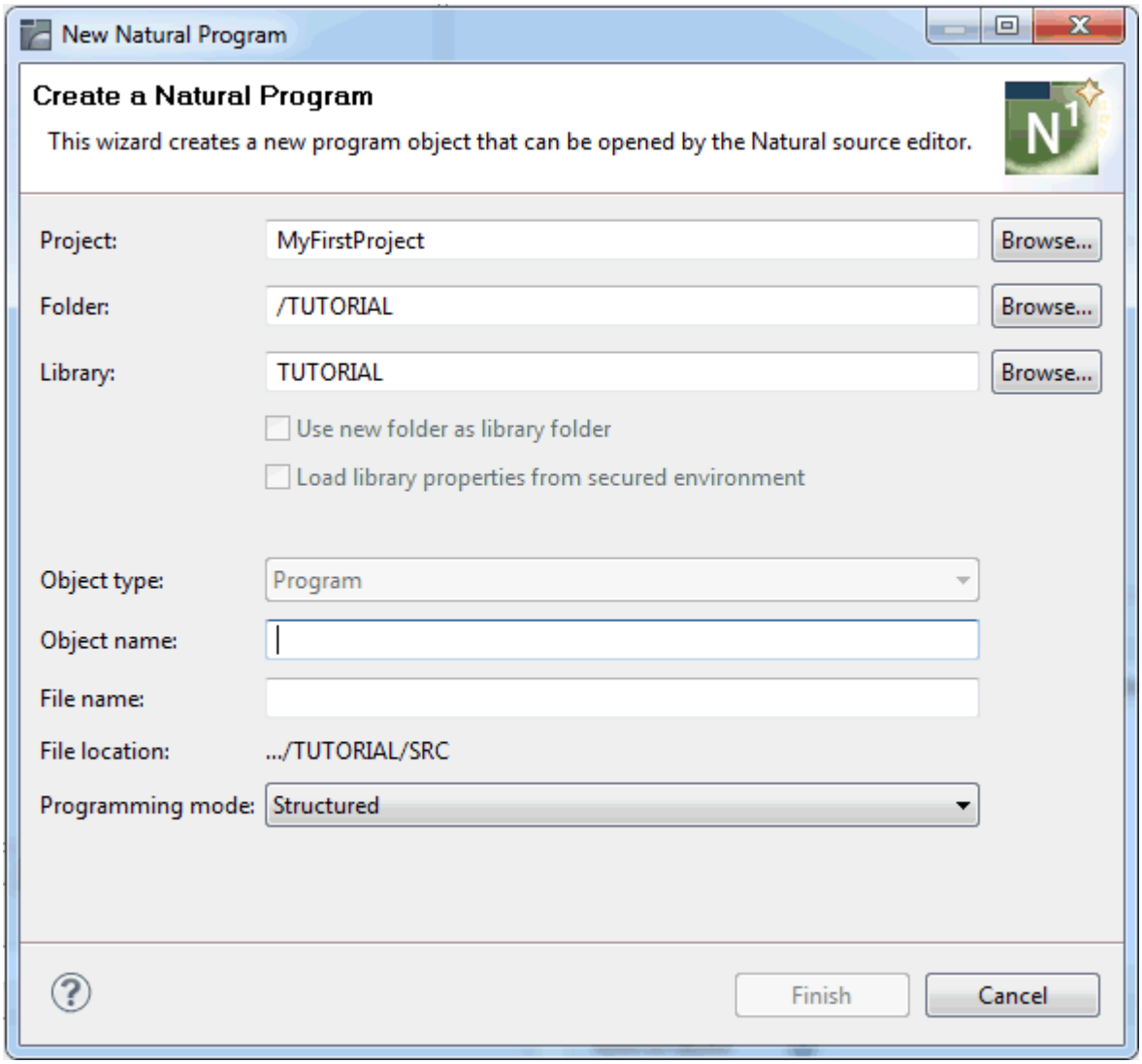
Program
Subprogram
Subroutine
Function
Copycode
Helproutine
Text
Global Data Area
Local Data Area
Parameter Data Area
Map
DDM

Or:

Invoke the context menu and choose **New >** *`object-type`*.

Tip: When you invoke the context menu in the **Natural Navigator** view, the items under **New** are grouped according to the plug-ins by which they are provided.

A dialog box appears for the selected object type. For example:



3 Specify the following information:

Option	Description
Project	The project in which the object is to be created. If you have selected a library before invoking this dialog box, the project name is automatically shown in this text box. However, if you have not selected a library (or if you have invoked the dialog box, for example, in a new empty workspace), you have to enter the name of a project to be created.
Folder	The folder in the file system in which the object will be created. Using the **Browse** button, you can select a different folder.
Library	The library in which the object is to be created. If you have selected a library before invoking this dialog box, the library name is automatically shown in this text box. However, if you have not selected a library, you have to specify the library yourself. The dialog box that is invoked by choosing the **Browse** button only offers Natural

Option	Description
	libraries for selection. Other folders on the same level which are not recognized as Natural libraries are not offered for selection.
Use new folder as library folder	Only enabled when you have specified a new folder name. If you want to store the new object in a new library folder, enable this check box. In this case, a new library is created as a library folder. It is important that you specify the new folder name as shown in the following example: `/TUTORIAL/mylibfolder` Do not omit the slashes and the library name.
Load library properties from secured environment	Only enabled when Natural Security is active in the associated Natural environment. When enabled, this check box is automatically activated when you add the object to a library which does not yet exist in your workspace. In this case, the library properties for the newly created library are loaded from the server. If a corresponding library does not exist on the server, the properties from the logon library of that server are used. If the associated Natural environment is not accessible, a dialog box appears. In this case, you can either cancel or continue the operation. If you continue, the properties are set to the defaults of the selected project. When Natural Security is active and this check box is deactivated, the properties for the newly created library are set to the defaults of the selected project.
Object type	Read-only. Indicates the object type that will be created.
DDM name	Only shown when you create a DDM. This is the long name of the DDM as it is used in a program.
Object name	A name for the new object. This name must correspond to the Natural naming conventions. The object name is not relevant for DDMs and therefore is not shown for this object type.
File name	The name from the **DDM name** text box (for DDMs) or **Object name** text box (for all other object types) is automatically provided as the file name. If you want, you can enter a different file name for this object. As long as both the object name and file name are identical, a file name is not used. The file name is only shown in the **Navigator** view or in the **Natural Navigator** view when it is different from the object name. This enables you to use a file name which does not adhere to the Natural naming conventions (for example, when it contains lowercase characters).
File location	Read-only. Shows the path where the new object will be created in the file system. The path that is shown here depends on the setting of the **Group new objects by object type** option in the **project properties**. When this option is not selected, this path may include the special folder *SRC* or a library folder. When this option is selected, the path includes the appropriate group folder for the current object type. For further information, see ***Group Folders***.

Option	Description
	The notation ".../" which is shown in front of the path is used to indicate that the project name and, if used, the name of the library root folder are also part of the path. For example, ".../" can stand for "/MyFirstProject/Natural-Libraries/".
Programming mode	Select either **Structured** or **Reporting**. The programming mode as specified in the project properties is provided as the default value (see the description of the **parser options**). For further information on the programming modes, see the *Programming Guide* in the Natural documentation for the appropriate platform. **Note:** A programming mode cannot be specified when you create a new DDM or text. The corresponding drop-down list box is disabled in these cases.

Note: When you create a new DDM, additional pages are available in the wizard. See *Creating a DDM* for further information.

4 Choose the **Finish** button.

The new object is created in the specified library and the associated editor is automatically invoked. See *Using the Natural Editors* for further information.

Note: When you create a new Natural object which uses the source editor, a skeleton which is typical for this type of object is automatically provided in the source editor. If you want to change a skeleton, see *Object Templates* in *Setting the Preferences*.

Changing the Object Properties

Some of the object properties (such as programming mode and code page) are stored in the **source header**.

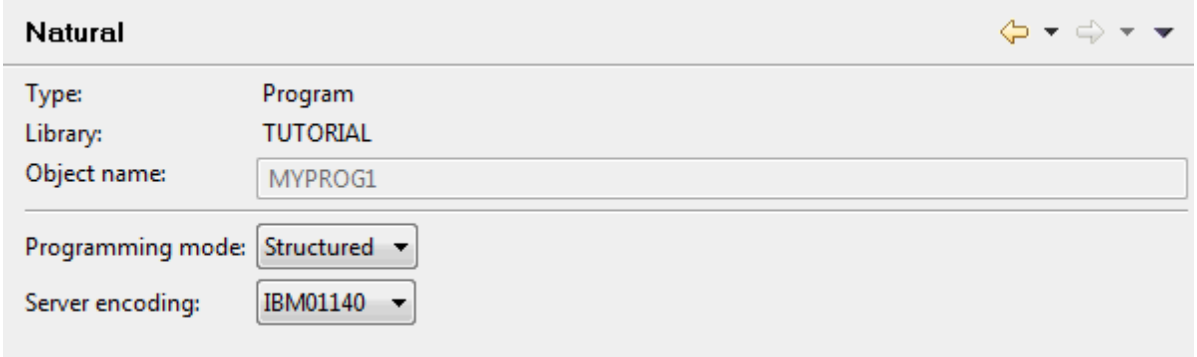
To change the object properties

1 Select the object in the **Navigator** view or in the **Natural Navigator** view.

2 Invoke the context menu and choose **Properties**.

3 In the tree of the resulting dialog box, select the **Natural** node.

The following property page is shown (example for a program).



If available, the long name of an object is shown. For a DDM, the database ID and file number are also shown.

Note: If the object is currently open in an editor and if the editor content has been changed but not yet saved (indicated by an asterisk in the editor tab), the options on this property page are disabled. It is only possible to change the options when the editor content does not contain unsaved changes (that is when the editor tab does not contain an asterisk).

4 Make all required changes.

Option	Description
Object name	The object name must adhere to the Natural naming conventions. When a file name has been defined, you can only change the object name in the object properties. This property is not relevant for DDMs and therefore is not shown for this object type. **Note:** The file name is defined by **renaming** the object directly in the workspace. With the default **label decorations**, the new object name is shown in parentheses in the **Navigator** view or in the **Natural Navigator** view, directly behind the file name. **Important:** When you change the object name in the properties, make sure to adapt this name in the code of all other Natural objects which reference this object.
Programming mode	Structured mode is intended for the implementation of complex applications with a clear and well-defined program structure. It is recommended to use structured mode exclusively. Reporting mode is only useful for the creation of ad hoc reports and small programs which do not involve complex data and/or programming constructs. For further information on the programming modes, see the *Programming Guide* in the Natural documentation for the appropriate platform.
Server encoding	This drop-down list box provides for selection the same code pages that are available in the properties of the corresponding project. Select the code page that is to be used when the object is stored on the server. For further information, see ***Unicode and Code Page Support***.

5 Choose the **OK** button to save your changes and to close the dialog box.

Editing Objects

When you edit a Natural object, the appropriate editor is invoked.

To edit Natural objects

1 In the **Navigator** view or in the **Natural Navigator** view, select the object(s) that you want to edit.

2 Invoke the context menu and choose **Open**.

Or:

Double-click each object that you want to open.

An editor window appears for each selected object. See ***Using the Natural Editors*** for further information.

Saving Objects

Object sources are saved using the standard Eclipse functionality.

Note: In the Natural preferences, you can specify that all changes to an object source are automatically updated on the server. See also ***Updating the Objects in the Natural Environment***.

To save an object

1 Activate the editor window for the source that you want to save.

2 From the **File** menu, choose **Save**.

Or:

Press CTRL+S.

Printing Objects

You can print object sources in two different ways, either directly from the source editor or from the **Natural Navigator** view.

When you print an object source from the source editor, the Eclipse functionality is used. The printout always contains the same information which is currently shown in the editor. This includes the source header, expanded and collapsed nodes, and, if enabled, line numbers.

When you print an object source from the **Natural Navigator** view, the printout contains Natural-specific information. It has a header containing the object name and object type, the name of the library in which is its stored, and the date and time of printing. The printout always shows the source header and the complete Natural code. Whether line numbers are printed depends on the setting of the corresponding option in the Natural preferences (see ***Natural Navigator*** in *Setting the Preferences*).

To print an object from the Natural Navigator view

1 In the **Natural Navigator** view, select the object(s) that you want to print.

2 From the **File** menu, choose **Print**.

   Or:

   Invoke the context menu and choose **Print**.

   Or:

   Press CTRL+P.

3 In the resulting **Print** dialog box, specify all required information and choose the **Print** button.

   Note: The settings given under **Page Range** in the **Print** dialog box are ignored if you print from the **Natural Navigator** view. In this view, always the complete Natural code of all selected members is printed.

To print an object from the source editor

1 Edit the Natural source and set the focus to the editor area.

2 Eventually select individual lines of source code that you want to print.

3 From the **File** menu, choose **Print**.

   Or:

Press CTRL+P.

4 In the resulting **Print** dialog box, specify all required information and choose the **Print** button. If you want to print only the selected lines or specific pages, provide the appropriate settings under **Page Range** in the **Print** dialog box.

Executing Objects

To execute a Natural program with NaturalONE, you must first compile it in your Natural environment in order to create a generated program in this environment. See also ***Updating the Objects in the Natural Environment***.

The library in which a generated program is executed is determined by the mode you have defined for the project (either shared mode or private mode). See ***Steplibs*** in *Changing the Project Properties* for further information.

Important: On a Natural server, Web I/O must be enabled. Otherwise, the output of an executed program cannot be displayed.

This section covers the following topics:

- Executing an Object

Executing an Object

For details concerning the **Default Launch** settings, see ***Launching Natural Applications***.

To execute an object

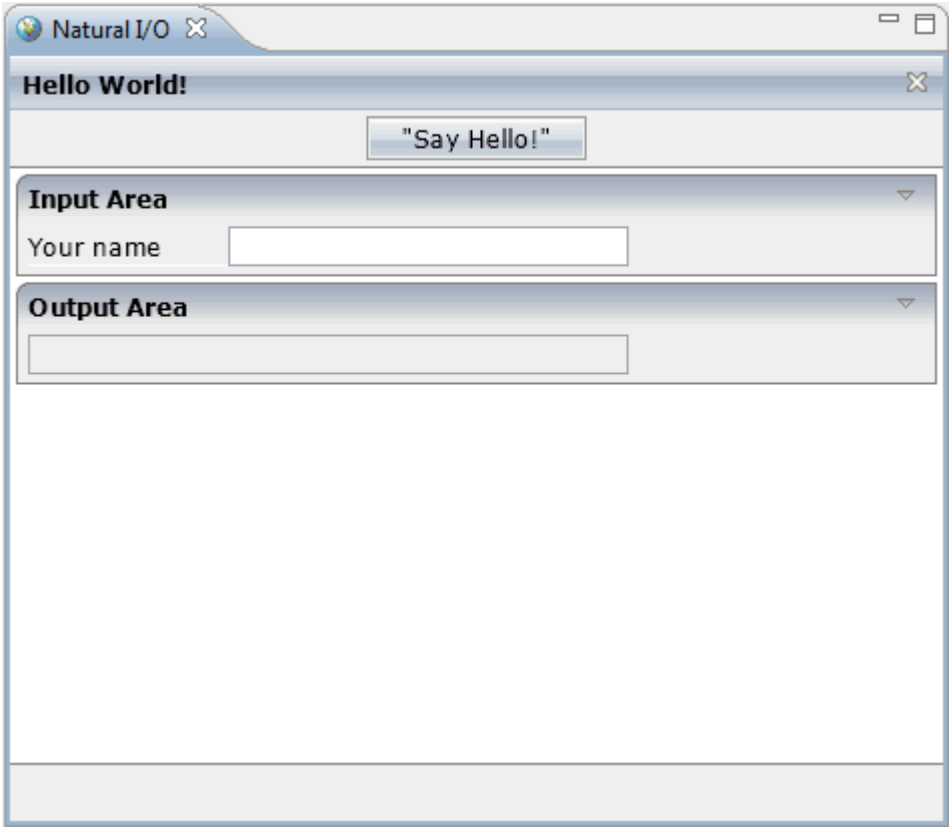
1 Select the program that you want to execute.

2 Invoke the context menu and choose **Run As > Natural Application**.

Or:

Press ALT+SHIFT+X, N.

The object is executed in your Natural environment (either the local Natural runtime or a Natural server). The output is either shown in the internal browser or in an external browser, depending on the settings in the Natural **preferences** or in the launch configuration that you have created (see ***Launching Natural Applications***). Example for the internal browser:



Copying and Pasting Objects and Libraries

You can use the commands **Copy** and **Paste** in the **Navigator** view and in the **Natural Navigator** view. Whereas the **Navigator** view follows the Eclipse-specific rules, the **Natural Navigator** view provides enhanced, Natural-specific support for copying and pasting Natural objects and libraries.

Caution: While the creation of new Natural sources via the wizards is under the control of NaturalONE, the creation of Natural sources via copy-and-paste is under the control of Eclipse. This does not cause problems with Natural projects since they use UTF-8 by default. However, it may cause problems with the copy when you are pasting into a non-Natural project - wrong characters may be shown because Eclipse uses a different code page. To solve the problem with the code page, you have to set the default encoding for text files to **UTF-8** in the Eclipse preferences (under **Preferences > General > Workspace > Text file encoding**).

This section covers the following topics:

- Navigator View
- Natural Navigator View

- Copying Objects from the File System

Navigator View

When you copy and paste objects in the **Navigator** view, there are no special restrictions as to the folder structure. However, you have to make sure that the pasted objects are always assigned to a library. You also have to make sure that valid Natural object names are used, especially when you copy an object and paste it in the same location as the original object. The name proposal that is offered by Eclipse does not meet the Natural naming conventions.

Natural Navigator View

How objects and libraries are copied and pasted in the **Natural Navigator** view depends on the settings of the toggle buttons in the local toolbar of that view and also on your project settings. For a description of the toggle buttons and their corresponding commands in the view menu of the **Natural Navigator** view, see ***Using the Natural Navigator View***.

- **Group new objects by object type**
  When this option is active in the **project properties**, the Natural objects you have copied are automatically pasted into the appropriate **group folders**, according to their object types. If a group folder does not yet exist for a Natural object that you paste, the corresponding folder is automatically created in the file system.

  This behavior is independant of the setting of the **Group Objects by Type** command (G toggle button).

- **Copying group folders**
  When the **Group Objects by Type** command (G toggle button) is currently selected and you copy a virtual group folder such as "Programs", only the content of that group is copied. The virtual group folder itself is not pasted at the new position.

  However, when you copy a physical group folder, that group folder including all of its contents is physically pasted at the new position.

  Tip: If you want to copy group folders, make sure that the **Group Objects by Type** command (G toggle button) is not selected. Thus, only the physical group folders are shown and you can see immediately how the pasted objects are stored in the file system.

- **Copying libraries**
  If you want to copy the contents of one or more libraries, it is recommended that you select the **Show Library View** command (L toggle button). Thus, you can copy all objects of these libraries, no matter in which **library folders** they are currently stored.

- **Special folders *SRC*,*RES* and*ERR***
  When the special folders *SRC*, *RES* and *ERR* exist in a library and you want to copy their contents to a virtual folder, make sure to copy only the contents of these folders (and not the special folders themselves). It is not possible, for example, to copy an *SRC* folder and paste it on a virtual folder.

  Make sure not to use both special folders and physical group folders within the same library.

  When you copy, for example, a virtual "Error Messages" group folder and a virtual "Programs" group folder in one library and you want to paste the contents of these folders in another library which contains the special folders, you can simply select the library before pasting. When you paste the objects, they are automatically sorted into the appropriate special folders. When a special folder (for example, for the error messages) does not yet exist, it is automatically created.

  When the special folders are visible, it is not possible to paste an object in a special folder which is not of the appropriate type (for example, it is not possible to paste an error message into an *SRC* folder).
- **Root folders**
  When the root folders are currently not shown and you paste objects onto the project node, a dialog appears in which you have to select the destination folder.

  When the root folder is shown, however, NaturalONE assumes that you want to paste your items directly below the project folder and the dialog does not appear.
- **Duplicating an object or library**
  When you copy an object (except for error messages, see below) or a library and paste it in the same location, a dialog box appears which already contains a proposal for a new name. In this case, you can either use the proposed name or you can enter a different name which does not yet exist. When you paste an object, the dialog box also allows you to change the object type.

  When a file name exists for an object, a proposal for a new file name is also given. You must not enter a file extension. This is automatically provided, depending on the object type that is selected in the dialog box.

  When you copy an error message and paste it in the same location, a dialog box appears which already contains a proposal for a language which does not yet exist in the library. In this case, you can either use the proposed language or you can select a different language from the drop-down list box. The selected language is automatically reflected in the object name.

  When you paste an object, the **Natural Navigator** view checks whether another folder of the target library already contains an object with the same name. If such an object already exists, a dialog box appears asking, for example, if you want to create the object in the selected folder and at the same time remove the duplicate object from the other folder.

- **Undo commands in the Edit menu**
  When a paste operation in the **Natural Navigator** view causes the deletion of an object (for example, when you confirm to remove a duplicate object from another folder of the same target library), the **Edit** command contains separate commands for undoing the delete operation and for undoing the copy operation. If you want to undo both operations, you first undo the delete operation, and then the copy operation.
- **Different parents**
  In the **Natural Navigator** view, it is possible to copy and paste objects from different parents. This is not possible in the standard Eclipse **Navigator** view. If you copy objects from different parents in the **Natural Navigator** view and then paste them in the **Navigator** view, an error occurs, indicating that the resources must have the same parent.

Copying Objects from the File System

All Natural objects in the workspace follow specific format conventions. Each object is automatically converted into the appropriate code page during the download from the Natural server.

If the contents of your files follow the Natural format conventions and if the file names adhere to the naming conventions for Natural objects, you can drag (or copy and paste) your files from the file system (for example, from the Windows Explorer) to the **Navigator** view or to the **Natural Navigator** view, or you can drag them within the file system to the folder which is defined as your Eclipse workspace. NaturalONE will recognize these files as Natural objects.

In addition to the Natural objects which adhere to the Natural naming conventions, your workspace may also contain Natural objects for which **alternative file names** have been defined (with any combination of uppercase and lowercase in the file name, but with uppercase in the file extension). In this case, it is important that the Natural object name which adheres to the Natural naming conventions can be found in the **source header** of the file. Otherwise, the file will not be considered as a valid Natural object when you copy (or drag) it to your Eclipse workspace or to the **Navigator** view or **Natural Navigator** view.

If the new files are not immediately shown in the **Navigator** view or **Natural Navigator** view, you have to refresh the display.

Moving Objects and Libraries

In the **Navigator** view, which follows the Eclipse-specific rules, you can use the **Move** command, or you can use drag-and-drop. When you use the **Move** command with a selected library or object, a dialog box appears in which you can choose the destination for the selected objects. Such a dialog box does not appear when using drag-and-drop.

The **Natural Navigator** view provides enhanced, Natural-specific support for cutting and pasting Natural objects and libraries. How objects and libraries are cut and pasted in the **Natural Navig-**

ator view depends on the settings of the toggle buttons in the local toolbar of that view and also on your project settings. See also the information under *Copying and Pasting Objects and Libraries*.

When you cut an object in the **Natural Navigator** view and paste it on a node in the **Natural Navigator** view, the source object is deleted after it has been copied to the target node. The deletion, however, will only work if the paste operation is also performed in the **Natural Navigator** view. If you cut an object in the **Natural Navigator** view and paste it in the standard Eclipse **Navigator** view, this will be handled as a copy operation.

Renaming Objects and Libraries

You can use the **Rename** command in the **Navigator** view and in the **Natural Navigator** view. Whereas the **Navigator** view follows the Eclipse-specific rules, the **Natural Navigator** view provides enhanced, Natural-specific support for renaming Natural objects and libraries.

Important: On the Natural server, only object names and library names are used. They must adhere to the Natural naming convention for this platform. When you update the Natural server, the handling for renamed objects on the Natural server is determined by the option **Scratch server objects** in the Natural preferences (see *Natural > Builder* in *Setting the Preferences*).

This section covers the following topics:

- Navigator View
- Natural Navigator View

Navigator View

When you rename an object or library in the **Navigator** view, keep the following in mind: As long as the new name adheres to the Natural naming conventions, the name is considered as an object name or library name. When the new name no longer adheres to the Natural naming conventions (for example, when it is longer than 8 characters and/or contains lowercase characters), it is considered as a file name or library folder name. However, when a library folder name is renamed in such a way that it again adheres to the Natural naming conventions, the new name is then considered as a library name. With file names, this is slightly different: The new name is only considered as an object name when the original object name (which is shown in parentheses when a file name exists) and the new name are identical.

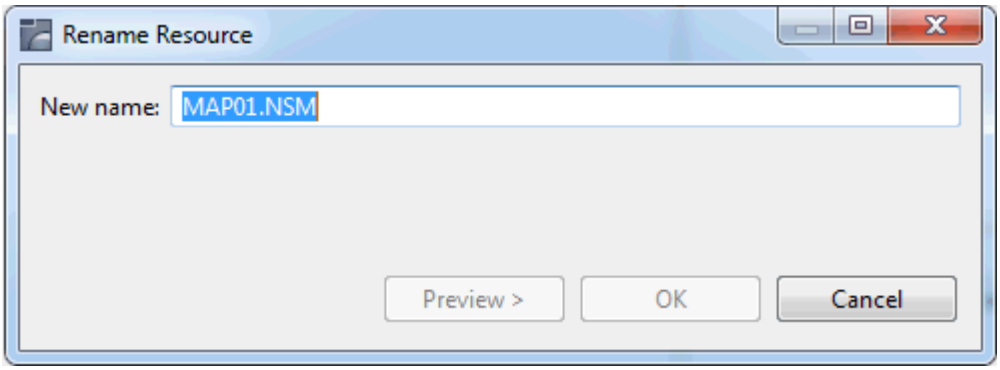
When a file name is defined, the object name can be changed in the **object properties**. When a library folder name is defined, a different library can be assigned in the **library properties**.

To rename an object or library

1 In the **Navigator** view, select the node that is to be renamed.

2 Invoke the context menu and choose **Rename**.

The following dialog box appears.

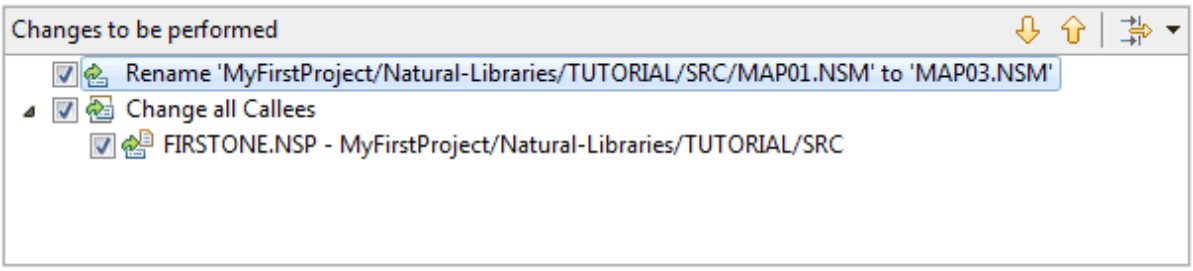


3 Enter a new name.

Caution: Do not change the file extension of an object. This will corrupt your application, and the object can no longer be edited with the appropriate editor.

4 Optional. If you want to see which changes are performed and which objects are affected by this operation, choose the **Preview** button.

Information such as the following can be shown. The object reference will automatically be changed in all objects (callees) for which the corresponding check box is selected.



Note: A list of callees is only shown when the Natural object name is changed, that is, the name which is referenced in the code of other Natural objects. When the name change results in a file name, a list of callees is not shown since such a change does not affect the code in other Natural objects.

5 Choose the **OK** button to rename the node.

Natural Navigator View

When you rename an object in the **Natural Navigator** view, you can be sure that the names of the libraries and objects follow the Natural naming conventions and that they meet the requirements for Natural objects (for example, when you change the object type).

To rename an object or library

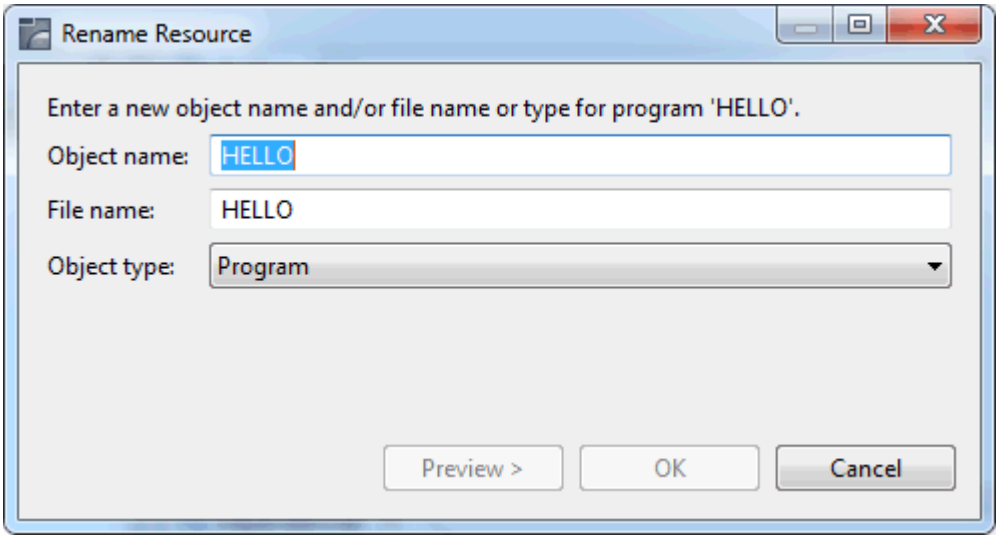
1 In the **Natural Navigator** view, select the node that is to be renamed.

2 Invoke the context menu and choose **Rename**.

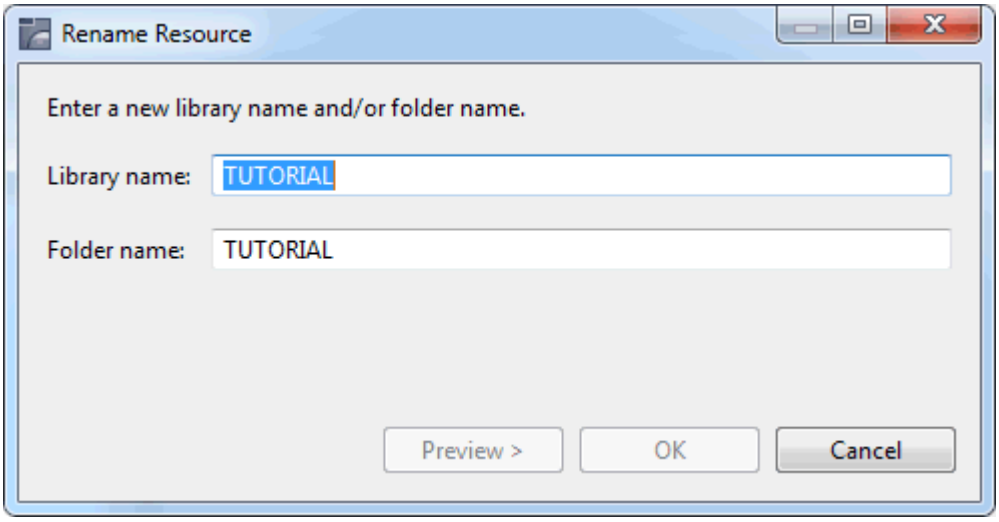
Or:

Press F2.

The following dialog box appears when you rename an object.



The following dialog box appears when you rename a library.



3 Enter a new object name or library name.

Or:

Enter a new file name for an object, or a new folder name for a library.

Or:

Enter both a new object name and a new file name for an object, or a new library name and a new folder name for a library.

4 Optional. If you want to change the object type (for example, from program to subprogram), select the corresponding entry from the **Object type** drop-down list box.

Only the allowed object types are available from this drop-down list box. A local data area, for example, can only be changed to a global data area or to a parameter data area.

It is not possible to change the object type for a map. Therefore, this drop-down list box is dimmed when renaming maps.

5 Optional. If you want to see which changes are performed and which objects are affected by this operation, choose the **Preview** button.

The same information is shown as when renaming an object in the **Navigator** view. The object reference will automatically be changed in all objects (callees) for which the corresponding check box is selected.

Note: A list of callees is only shown when the Natural object name is changed, that is, the name which is referenced in the code of other Natural objects. When the name change results in a file name, a list of callees is not shown since such a change does not affect the code in other Natural objects.

6 Choose the **OK** button to complete the rename operation.

Deleting Objects and Libraries

You can delete any Natural objects and libraries in the **Navigator** view or **Natural Navigator** view.

When you update the Natural server, the handling for deleted objects on the Natural server is determined by the option **Scratch server objects** in the Natural preferences (see ***Natural > Builder*** in *Setting the Preferences*).

To delete an object or library

1 In the **Navigator** view or in the **Natural Navigator** view, select the node that is to be deleted.

2 Invoke the context menu and choose **Delete**.

A dialog box appears, asking whether you really want to delete the node from the file system.

In the **Natural Navigator** view, it is possible to delete nodes which do not physically exist in the file system (for example, a **Programs** group node). If you are about to delete such a node, a special dialog box appears, asking, for example, if you want to delete all programs located in the current library or library folder from the file system.

3 Choose the **OK** button (**Navigator** view) or the **Yes** button (**Natural Navigator** view) to delete the node.

8 Launching Natural Applications

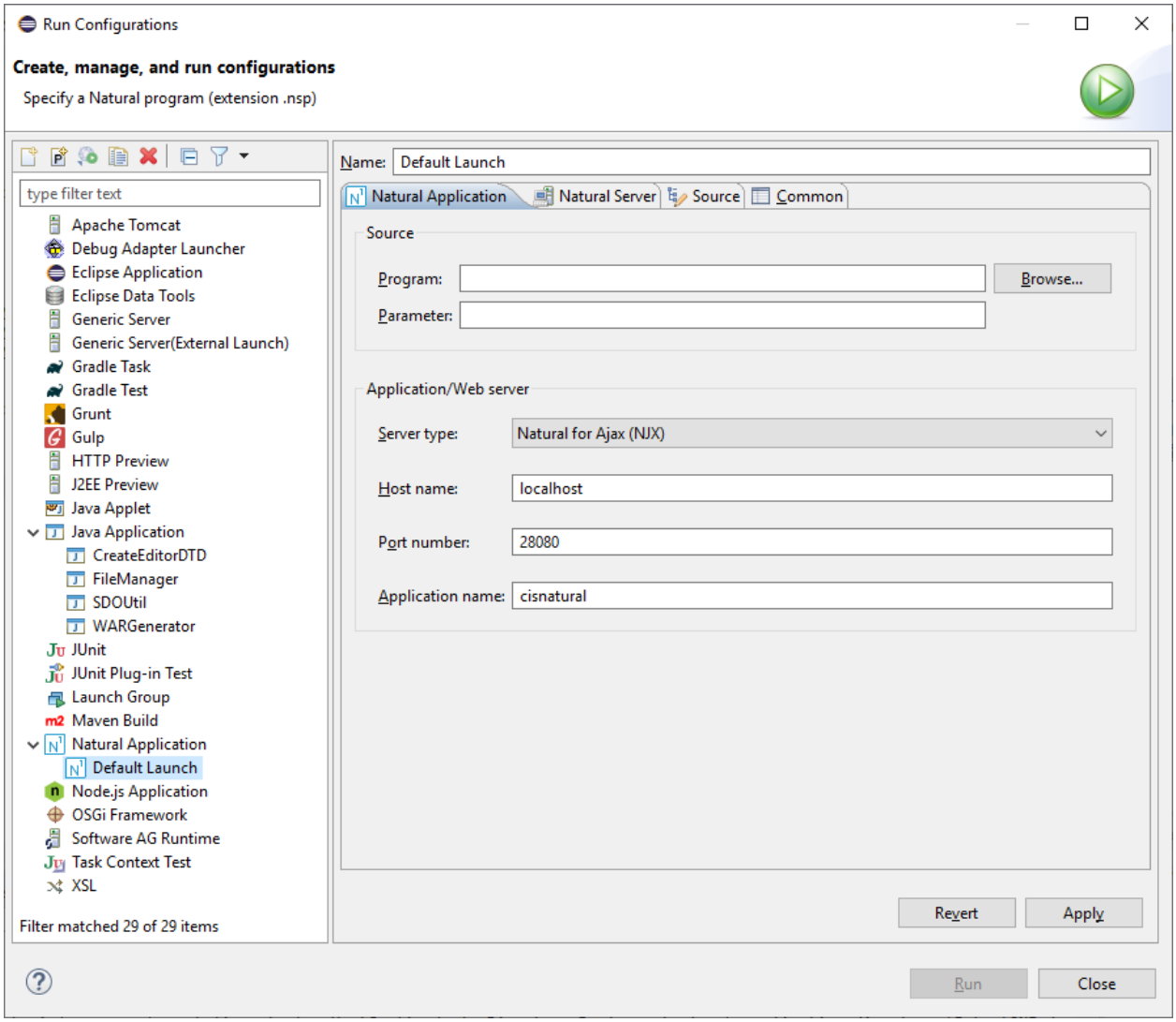
- Defining a Different Start Library 106
- Specifying Application Parameters 107
- Launching Applications Using Shortcut Keys 107

In order to launch a Natural Application (for **executing**, **debugging** or **profiling**), the Eclipse Launch configuration framework is used with the standard Eclipse commands **Run As**, **Debug As** and **Profile As**.

Within these commands NaturalONE is exposing a launch shortcut named **Natural Application** which maintains a configuration named **Default Launch**. Depending on the context where the shortcut is used, this launch will be updated accordingly.

Note: The **Default Launch** configuration is created the first time the **Natural Application** shortcut is used.

The following picture shows the information filled into the **Natural Application** tab of the **Default Launch** configuration for launching a Natural application:

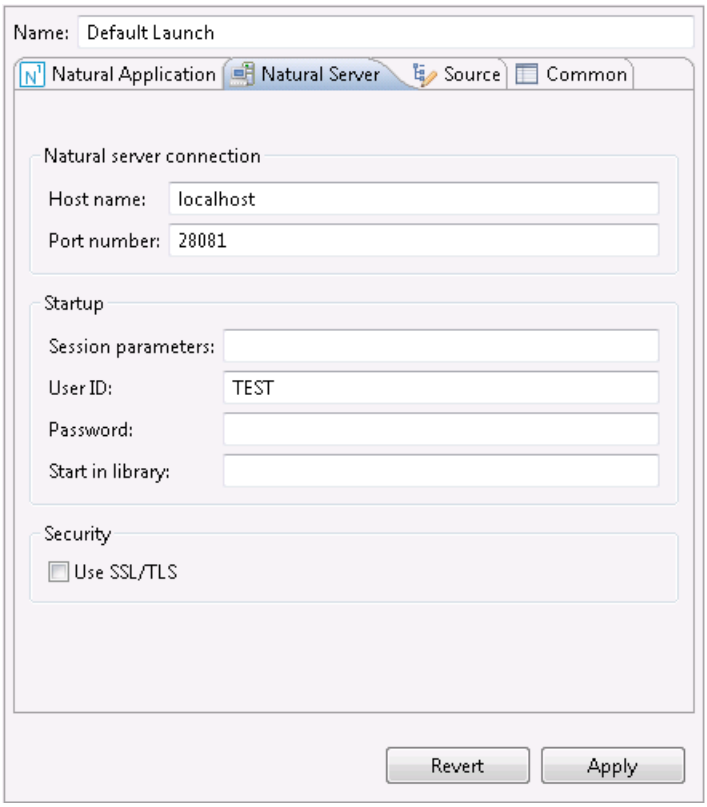


When one of the following commands

- **Run As - Natural Application**
- **Debug As - Natural Application**
- **Profile As - Natural Application**

is applied inside the Project `MyFirstProject` on the program `HELLO.NSP` in library *TUTORIAL*, then the **Program** field is populated with the corresponding path. In addition, the I/O relevant information is taken from the **Natural I/O Runtime Preferences**.

The following picture shows the information how the **Natural Server** tab of the **Default Launch** configuration is populated when launching a Natural application:



The **Natural Server** tab is addressing the runtime environment of the application. This information is retrieved from the Natural project properties (except the **Start in library** field), in this example from `MyFirstProject` which is using the local runtime.

Note:

And please keep in mind: This **Default Launch** configuration is being modified whenever the Natural Application shortcut is applied on different objects, for example when an object of `MySecondProject` running on a Linux runtime is selected then the **Default Launch** is populated with the corresponding path and server information.

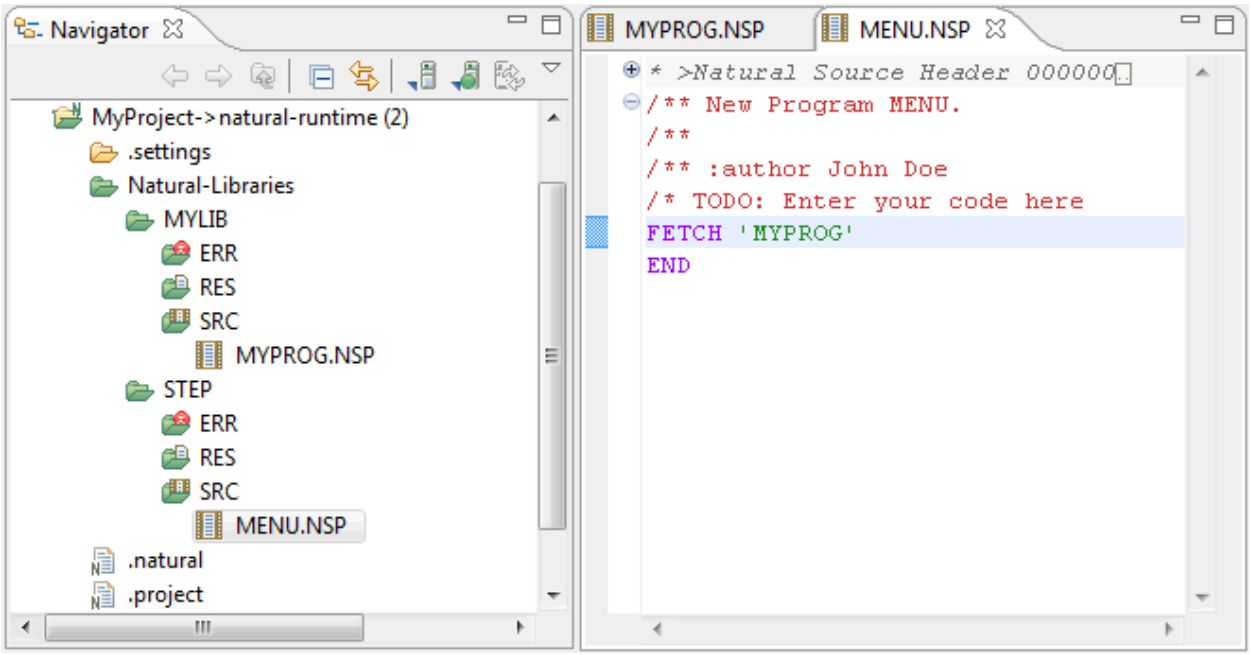
Defining a Different Start Library

When designing an application, the startup program is sometimes located in a steplib, and the main program which is invoked by the startup program is located in a different library. In order to run the startup program directly from the steplib, you have to define a different start library in the launch configuration of the startup program, using the option **Start in library**. This option corresponds to the Natural system command `LOGON` which is available with Natural for Mainframes, UNIX, OpenVMS and Windows.

Note: As already mentioned above, the **Start in library** field is not filled automatically (similar to the other Natural server parameters), but must be added by the user when required. Therefore, a custom Natural launch should be created instead of adding this field to the **Default Launch**. This can be achieved by copying the **Default Launch** to **Custom Launch** using the **Run Configurations** dialog and then manually modifying the appropriate field(s).

Example

Suppose your application has the following structure: The project properties define the library `STEP` as the steplib for all libraries within your project. The library `MYLIB` contains the main program named `MYPROG`, and the library `STEP` contains the startup program named `MENU`. The program `MENU` calls the program `MYPROG` which is located in a different library.



When you try to run the application by executing the program `MENU`, an error will occur because the program `MYPROG` cannot be found in the current library `STEP`. To avoid this error, you have to define a different start library as described below.

Note: When working with Natural itself (for example, with Natural for Mainframes), this would not be a problem: You would define `STEP` as a steplib of `MYLIB` so that the program `MENU` can be found in the steplib. Then you would simply `LOGON` to the library `MYLIB` and `EXECUTE` the program `MENU`.

Specifying Application Parameters

Depending on the application to be executed or debugged, it is also possible to specify the corresponding application parameter values in the **Parameter** field of the **Natural Application** tab. The individual parameters must be separated by one or more blank characters.

Note: As with the **Start in library** field, these parameters are application-specific. Therefore, a custom Natural launch should be created instead of adding the parameters to the **Default Launch** to avoid that selecting a different application in the workspace would then get the parameters of the previously used application.

Launching Applications Using Shortcut Keys

It is also possible to launch an application with the F11 (Debug) or CTRL+F11 (Execute) key binding. This is not specific to NaturalONE but standard Eclipse functionality. These key bindings always process the launch which was last executed.

Instead of using the Natural Application shortcut (which always uses the Default Launch), you can also create multiple launch configurations for different application launches (e.g. "Appl_UnixLaunch_with_Parameters", "Appl_WinLaunch_with_StartLibray" etc.) and execute these launches using the appropriate commands in the Eclipse **Run** menu.

9 Modifying Objects in the Natural Environment or in the Repository

- General Information 110
- Updating the Objects in the Natural Environment 110
- Build Sequence 114
- Canceling a Build 115
- Flags in a Label Decoration 115
- Resetting Flags 116
- Consolidating Objects in Private-Mode Libraries 117
- Rebuilding all Objects in the Natural Environment 118
- Using the Compare Editor 119
- Checking the Time Stamps in the Natural Environment 122
- Excluding Objects from Processing in the Natural Environment 128
- Committing the Objects to the Repository of the Version Control System 133

General Information

For catalog purposes, you can upload your new and changed sources to the appropriate Natural environment (Natural server or local Natural runtime). The appropriate Natural environment (that is, the environment to which your sources will be uploaded) is defined on the **Runtime** page of the project properties.

Objects are always uploaded to the active system file.

When Natural Security is active on the Natural server, the `SAVE` command must be allowed in the "Command Restrictions" of the library security profile of Natural Security in order to upload objects.

Caution: Different parsers are used with NaturalONE and in the Natural environment. The parser in the Natural environment is responsible for the syntactical correctness of a source. Since the NaturalONE parser may indicate errors which are not problematic in a Natural environment, the sources are always uploaded to the Natural environment, even if they contain errors. However, when the parser in the Natural environment then detects an error in a source, this source is not saved, not compiled and a cataloged object is therefore not created - the existing source and cataloged object are not replaced in this case.

You can also commit your new and changed sources to the repository of your version control system. When you work in local mode, it is most likely that your sources are kept in a version control system.

NaturalONE uses two different types of build commands. The Eclipse-specific build commands in the **Project** menu apply to the contents of the workspace (the **Navigator** view or **Natural Navigator** view). The NaturalONE-specific build command which is described under ***Updating the Objects in the Natural Environment*** applies to the contents of your Natural environment (Natural server or local Natural runtime).

See also ***Understanding the Behavior of the Natural Builder***.

Updating the Objects in the Natural Environment

The following topics are covered below:

- Automatic Update

- Manual Update

Automatic Update

When the option **Build Natural projects automatically** is enabled in the Natural preferences (see ***Natural > Builder*** in *Setting the Preferences*), the appropriate Natural environment (Natural server or local Natural runtime) is automatically updated each time you save a source in the Eclipse workspace or when you add a new source. The source is uploaded to the Natural environment and is stowed there.

Manual Update

When the **Build Natural projects automatically** option is disabled in the Natural preferences, the following commands can be used to update the Natural environment manually:

- **Build Natural Project**
  This command updates the Natural environment by uploading and stowing *all* new and changed sources of the current project. In addition to the new and changed sources, all other sources that reference changed sources are cataloged.

  You can prevent the cataloging of the referencing sources by disabling the option **Rebuild dependent objects** in the Natural preferences (see ***Natural > Builder*** in *Setting the Preferences*).
- **Upload**
  This command just uploads the selected sources to the Natural environment. They are neither cataloged nor stowed there.

  Caution: Since objects are not cataloged, the parser in the Natural environment is not invoked. In this case, it is possible to store erroneous sources in the Natural environment.

- **Update**
  This command uploads the selected sources to the Natural environment and stows them there. Sources that reference the selected sources are not stowed.

Notes:

1. In Natural, the term “stow” is used for the following: the source code is compiled and, when no errors are found, the resulting generated object code is stored as a cataloged object in a Natural system file. In addition, the source code is also stored in a Natural system file; it receives the same timestamp as the cataloged object.
2. In Natural, the term “catalog” is used for the following: the source code is compiled and, when no errors are found, the resulting generated object code is stored as a cataloged object in a Natural system file. The source code itself is not stored in a Natural system file.

When the option **Prompt on compile errors** is enabled in the Natural preferences, a message box will appear in the case of a compile error. For further information on this message box, see ***Natural > Builder*** in *Setting the Preferences*.

Actions which have been disallowed in Natural Security are not disabled in the context menus of the **Navigator** view or **Natural Navigator** view. If data is transferred to a Natural server (for example, with the **Build Natural Project** command) and an action is not allowed on this server, the server responds with an error message.

To upload and stow all new and changed sources

1 In the **Navigator** view or in the **Natural Navigator** view, select the project which contains the sources that you want to upload and stow.

   Or:

   Select any source within the project.

2 Invoke the context menu and choose **Build Natural Project**.

   Or:

   Choose the following icon in the local toolbar:

   Or:

   Press ALT+B.

   Note: For the **Build Natural Project** command, the `STOW` command must be allowed in the "Command Restrictions" of the library security profile. See also *Protecting the Natural Development Environment in Eclipse* in the *Natural Security* documentation, which is part of the Natural documentation.

   The **Build Natural Project** command always updates *all* sources in the Natural environment which have been changed in the project, or which are new. Even if you have selected a single file in the project (and this file has not even been changed), all changes in the project are updated in the Natural environment.

   Important: The library into which the sources are written and stowed is determined by the mode you have defined for the project (either shared mode or private mode). See ***Steplibs*** in *Changing the Project Properties* for further information.

To upload selected sources without cataloging

1 In the **Navigator** view or in the **Natural Navigator** view, select the source(s) that you want to upload.

   Or:

   Select a library, if you want to upload all sources in this library.

2 Invoke the context menu and choose **Upload**.

   Or:

   Choose the following icon in the local toolbar:

   If more than one Natural object is about to be uploaded, a dialog box appears, asking whether you really want to upload the selected objects and replace the corresponding objects on the server.

3 Optional. Select the check box **Always process selected objects without prompt**.

   When you select this check box, **Confirm server processing of selected objects** is automatically deselected in the preferences. See ***Natural > Builder*** in *Setting the Preferences*.

4 Choose the **Yes** button.

   Other than the **Build Natural Project** command, the **Upload** command only updates the sources in the Natural environment which have been selected in the project (that is, either single sources or, when a library has been selected, all sources in this library).

To upload and stow selected sources

1 In the **Navigator** view or in the **Natural Navigator** view, select the source(s) that you want to upload and stow.

   Or:

   Select a library or even a project, if you want to upload and stow all sources in this library or project.

2 Invoke the context menu and choose **NaturalONE > Update**.

   Or:

   Choose the following icon in the local toolbar:

Or:

Press ALT+T.

If more than one Natural object is about to be updated, a dialog box appears, asking whether you really want to update the selected objects and replace the corresponding objects on the server.

3 Optional. Select the check box **Always process selected objects without prompt**.

When you select this check box, **Confirm server processing of selected objects** is automatically deselected in the preferences. See *Natural > Builder* in *Setting the Preferences*.

4 Choose the **Yes** button.

Note: For object types that only have sources (such as copycode and text), the **Update** command behaves just like the **Upload** command.

Build Sequence

When the **Build Natural Project** command is used to upload and stow objects in the Natural environment, the following sequence is used by default:

Data definition modules
Global data areas
Local data areas
Parameter data areas
Classes
Maps
Adapters
Dialogs
Subroutines
Subprograms
Functions
Helproutines
Programs

This default sequence can be changed for all Natural projects in the Natural preferences (see *Natural > Build Sequence* in *Setting the Preferences*) or for a specific Natural project in the project properties (see *Builder* in *Changing the Project Properties*).

Canceling a Build

When you build a project with a large number of objects, this may take a while. You can watch the build progress in the **Progress** view. This is a standard Eclipse view. It is not part of the NaturalONE perspective. You can display it with **Window > Show View > Other > General > Progress** or by double-clicking the progress indicator at the bottom right of the Eclipse workbench.

Using the **Progress** view, it is possible to cancel a build.

Caution: It is recommended that you do not cancel a build since this will result in inconsistent builder data. This in turn may cause NaturalONE not to work properly anymore until a full build of the project is performed.

When you still decide to cancel a build, a dialog box appears, asking whether you really want to cancel the current build of the project. This dialog box contains the **Always cancel the build without prompt** check box. When you select this check box, **Confirm cancelation of builds** is automatically deselected in the preferences. See ***Natural > Builder*** in *Setting the Preferences*.

Flags in a Label Decoration

When the **Build Natural projects automatically** option is disabled in the Natural preferences, different types of flags can be shown in the **Navigator** view or in the **Natural Navigator** view, indicating the status of the corresponding object. The following flags are used by default (see also ***Label Decorations*** in *Setting the Preferences*). They are shown with the name of a source.

Flag	Type of Flag	Description
^	Upload flag	Indicates that the source for which this flag is shown needs to be uploaded to the appropriate Natural environment.
%	Stow flag	Indicates that the source for which this flag is shown needs to be stowed in the appropriate Natural environment. For example, when a program has been changed, the program shows the upload flag and the stow flag.
°	Catalog flag	Indicates that the source for which this flag is shown uses another source which has been changed. The source for which this flag is shown needs to be cataloged in the appropriate Natural environment. For example, when a program uses copycode and this copycode is changed, the copycode then shows the upload flag and the program shows the catalog flag. When a program uses, for example, a subroutine and this subroutine is changed, the subroutine then shows the upload flag and the stow flag. The program itself, which has not been changed, shows the catalog flag.

Flag	Type of Flag	Description
		As a rule, the catalog flag is only shown when the stow flag does not apply.
~	Scratch flag	Only shown when the **Clean** command from the **Project** menu is used while the **Build Natural projects automatically** option is disabled in the Natural preferences. Indicates that the source and the generated object for which this flag is shown need to be deleted in the appropriate Natural environment.

Flags are not only shown for sources but also for other nodes. By default, an update flag is shown for all upper nodes (project nodes, library nodes, and the subnodes of a library). When a project node is collapsed, you can thus see immediately that this project contains sources that need to be updated in the appropriate Natural environment.

Flag	Type of Flag	Description
*	Update flag	Indicates that one or more sources need to be updated in the appropriate Natural environment.

All of the above flags disappear when the command **Build Natural Project** is issued.

The following flags are displayed independently of the **Build Natural projects automatically** option and they do not disappear when the command **Build Natural Project** is issued. These flags are not shown by default. You have to define them in the Natural preferences by adding the corresponding variables (see ***Label Decorations*** in *Setting the Preferences*).

Flag	Type of Flag	Description
+	Private-mode flag	Shown for all objects that are available in a private-mode library.
-	Consolidate flag	Only shown when the **Find Objects to Consolidate** command was used in order to find all objects which are obsolete in the private-mode library. See also ***Consolidating Objects in Private-Mode Libraries***.

Note: Flags are not shown for excluded objects. See also ***Excluding Objects from Processing in the Natural Environment***.

Resetting Flags

You can reset the flags in the label decorations (which are described **above**) so that they are no longer shown. Any information on the objects that need to be updated in the Natural environment is then lost.

Resetting flags is helpful when you are sure that the objects in your workspace are the same as those in the Natural environment. This may be the case, for example, when you are working with a version control system and the objects in the Natural environment are automatically updated

each night. When you update the objects in your workspace with the modifications of another user, you can then safely reset the flags.

To reset the flags

1 In the **Navigator** view or in the **Natural Navigator** view, select the node(s) for which you want to reset the flags. This may be a Natural project, library, library folder or Natural object.

2 Invoke the context menu and choose **NaturalONE > Reset Flags**.

> **Note:** When all objects in the selected node are currently **excluded from processing** in the Natural environment, the **Reset Flags** command is not available.

A dialog box appears, asking whether you really want to reset the flags.

3 Optional. Select the check box **Always reset without prompt**.

When you select this check box, **Confirm resetting flags** is automatically deselected in the preferences. See *Natural > Builder* in *Setting the Preferences*.

4 Choose the **Yes** button.

Consolidating Objects in Private-Mode Libraries

Only available in private mode.

You can show which objects are stored in private-mode libraries and which objects are obsolete in the private-mode libraries because they are identical to those stored in the base libraries. Obsolete objects can be consolidated. The following commands are available for this purpose:

- **Find Objects to Consolidate**
  This command checks for all selected objects for which the private-mode flag is shown whether the object in the private mode library is obsolete or not. All objects which can be consolidated are then shown with the consolidate flag.

  A Natural source is obsolete in the private-mode library if its source is identical to the corresponding source in the base library.

  The cataloged object of a Natural source (also called "generated program" or "GP") is obsolete in the private-mode library if the sources of all objects which are referenced from the current object, are obsolete or not available in the private-mode library.

- **Consolidate**
  This command acts on all selected objects which can be consolidated in the private-mode library. These are the objects which have been found with the **Find Objects to Consolidate** command and for which the consolidate flag is shown. The **Consolidate** command purges, uncatalogs or

scratches the objects from the private-mode library, depending on whether the source, cataloged object or both can be deleted.

The above-mentioned flags are part of the label decoration. However, they are not shown by default. You have to define them in the Natural preferences by adding the corresponding variables. For more information, see ***Label Decorations*** in *Setting the Preferences*.

To find the objects that can be consolidated

1. In the **Navigator** view or in the **Natural Navigator** view, select the node that you want to check. If you want to find all objects in a project which can be consolidated, it is recommended that you choose the project node.
2. Invoke the context menu and choose **NaturalONE > Find Objects to Consolidate**.

   All objects which can be consolidated are marked with the consolidate flag.

To consolidate the found objects

1. In the **Navigator** view or in the **Natural Navigator** view, select the node that you want to consolidate. If you want to consolidate all objects in a project, it is recommended that you choose the project node.
2. Invoke the context menu and choose **NaturalONE > Consolidate**.

   Note: This command is only available if objects to be consolidated have been found.

Rebuilding all Objects in the Natural Environment

Only available in shared mode.

Instead of updating only the new and changed objects in a Natural environment, you can also update *all* objects in the Natural environment which belong to your project.

Caution: When you rebuild all objects in the Natural environment, this will delete any changes of other users in the affected libraries.

Rebuilding the objects in the Natural environment is also helpful, for example, in the following cases:

- You have checked out a project from the repository of your version control system, and the cataloged objects for the sources in this project are not yet available on the Natural server.
- You want to move your project to another Natural server. In this case, you first have to specify the appropriate mapping information for the new server in the project properties.

To rebuild all objects in the Natural environment

1 In the **Navigator** view or in the **Natural Navigator** view, select the project that you want to rebuild.

2 Invoke the context menu and choose **Rebuild Natural Project**.

Or:

Press ALT+R.

A dialog box appears, asking whether you really want to rebuild the project.

3 Optional. Activate the check box **Delete the contents of the affected libraries on the server first**.

When you activate this check box, the entire contents of all libraries within your project are deleted from the Natural environment before the objects from the Eclipse workspace are uploaded and stowed in the Natural environment. This avoids an inconsistent state in the Natural environment. Libraries of the `FNAT` system file and the `SYSTEM` library of the `FUSER` system file will not be deleted.

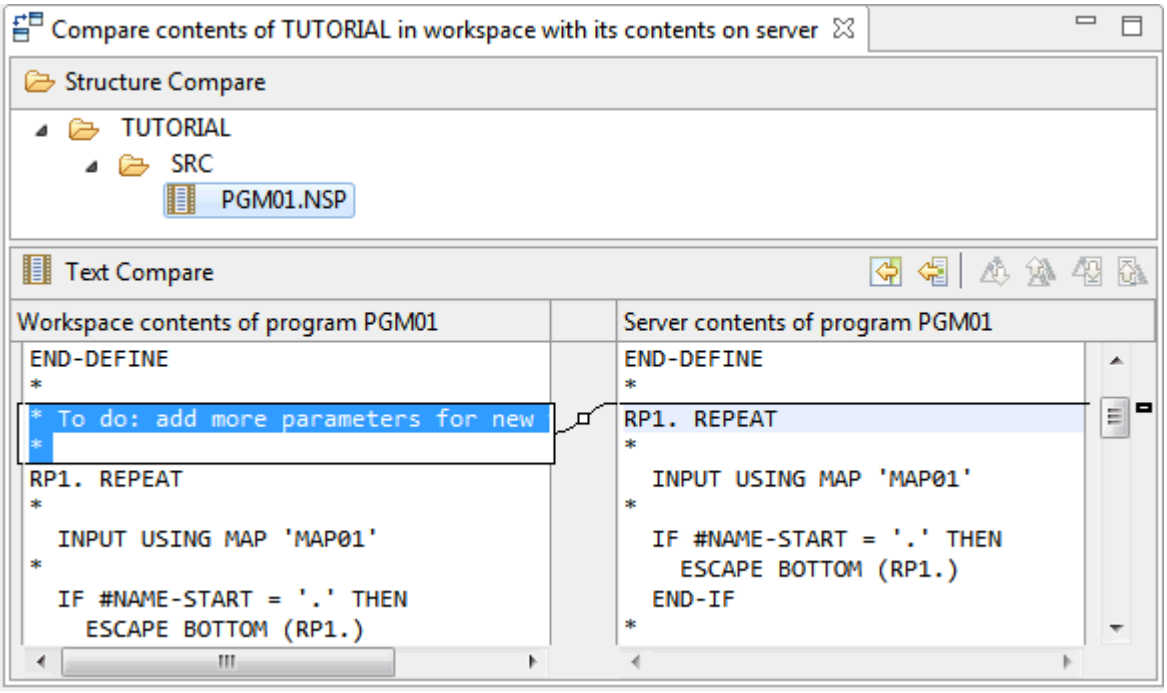
Caution: The deletion may also include objects in the Natural environment that do not exist in the project of your Eclipse workspace.

When this check box is not selected, any objects that are not part of the project remain in the Natural environment. This may cause problems.

Using the Compare Editor

You can use the compare editor of Eclipse to compare the sources from your Natural projects in the workspace with the corresponding objects in the Natural environment (for example, with the objects on a mainframe server). The sources can be compared with both, the objects on the server which are stored in the base library (that is, the library which has the same name as in the workspace) and, if private mode has been enabled, with the objects on the server which are stored in a private-mode library.

You can invoke the compare editor either for single sources or for several sources. In the latter case, you can select, for example, the node for a project or library. If the compare editor is invoked for more than one source, it shows a tree in the upper pane listing all sources which are different. You can then double-click a source to view the differences. For example:



Note: If you invoke the compare editor for more than a single source this may take some time. For example, if you invoke the compare editor for a library with many sources, all sources in that library need to be compared first before the structure of the compare tree can be filled.

For more information on the compare editor, see the Eclipse online help.

To compare sources with the objects in the server's base library

1 In the **Navigator** view or in the **Natural Navigator** view, select the source that you want to compare.

Or:

Select a node (such as a project or library node) which contains the sources that you want to compare.

Or:

In the **Time Stamp Conflicts** view, select the source that you want to compare. See *Checking the Time Stamps in the Natural Environment* for more information on this view.

2 Invoke the context menu and choose **Compare With > Base Library on Server**.

If there are no differences, a corresponding message is shown. If differences are found, the compare editor appears.

Using the compare editor, it is possible to copy the changes from the object on the server (right side) to the source in the workspace (left side). The following buttons are available for this purpose: (Copy All Non-Conflicting Changes from Right to Left) and (Copy Current Change from Right to Left). You can then use the standard Eclipse functionality to save the changes in the source. If you close the compare editor and there are still unsaved changes, you are asked if you want to save them.

It is not possible, however, to copy changes from the source in the workspace to the object on the server using the compare editor. To do so, you have to use NaturalONE's **update** functionality.

To compare sources with the objects in the server's private-mode library

1 In the **Navigator** view or in the **Natural Navigator** view, select the source that you want to compare.

Or:

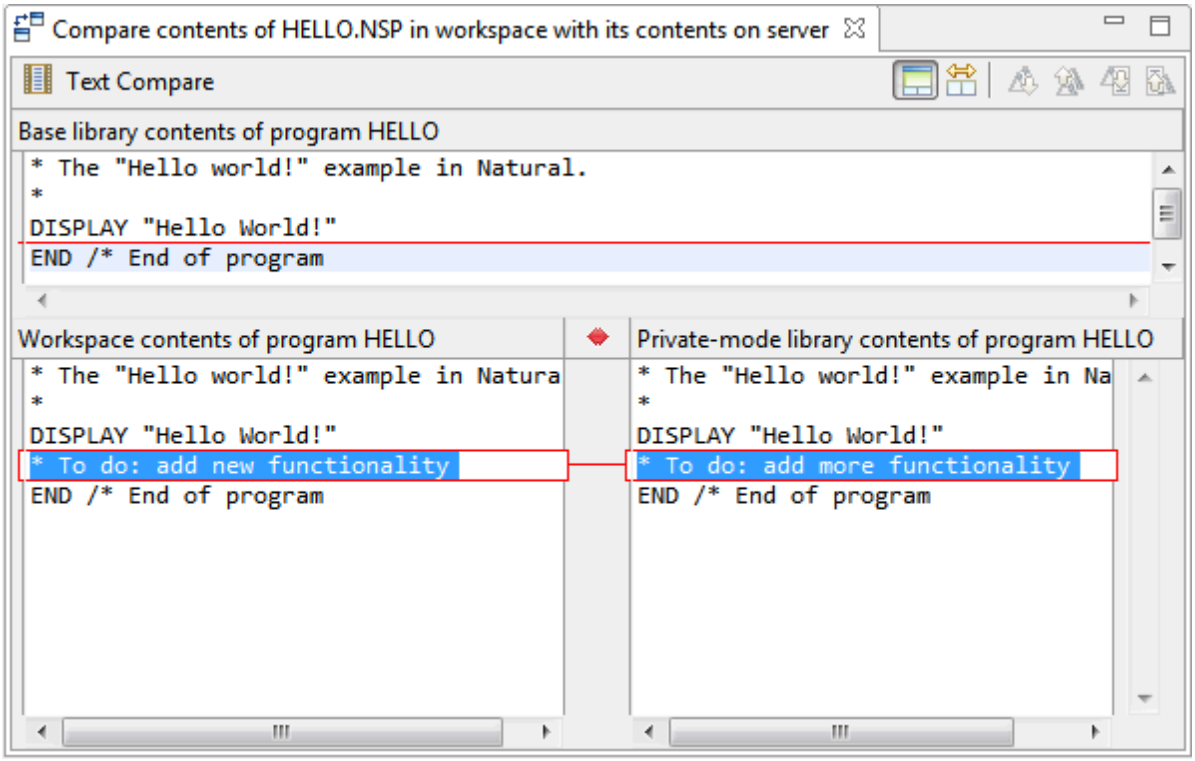
Select a node (such as a project or library node) which contains the sources that you want to compare.

2 Invoke the context menu and choose **Compare With > Private-Mode Library on Server**.

If there are no differences, a corresponding message is shown. If differences are found, the compare editor appears.

3 The following applies only, if the differences for a single source are shown in the compare editor: If you also want to display the content of the object in the server's base library, choose the (Three-Way Compare) button and then the (Show Ancestor Pane) button.

The content of the object in the base library is then shown in the upper pane of the compare editor. For example:



When comparing with a private-mode library, the compare mode is read-only. It is not possible to copy changes from one source to another.

Checking the Time Stamps in the Natural Environment

The following topics are covered below:

- General Information on Time Stamp Checking
- Time Stamp Conflicts During an Update of the Natural Environment
- Resolving a Time Stamp Conflict
- Comparing the Time Stamps Before an Update of the Natural Environment
- Time Stamp Conflicts View

- Logging Time Stamp Conflicts with Command Line Arguments

General Information on Time Stamp Checking

This feature is helpful, for example, when part of the development team works with Eclipse and another part of the development team works directly in the Natural environment (for example, on a mainframe server). You can find out whether the source has been modified on the server in the meantime. Thus, you can avoid that an object which another user has changed on the server is overwritten with your changes.

The objects are checked in the following way: The time stamp of the source in the local workspace is compared with the time stamp of the corresponding source on the server. When the time stamps are different, the contents of the sources are compared. When the sources are identical, no conflict occurs. When the sources are not identical, a time stamp conflict occurs. This approach makes sure that a time stamp conflict occurs only when the content of the source on the server has been modified. This makes sense, for example, when the system command `CATALL * STOW` has been executed on a mainframe server, which changes the time stamps of the sources but not their contents.

In order to use this feature, you must make sure that the **Check time stamp on server** option is enabled in the Natural preferences (see ***Natural > Builder*** in *Setting the Preferences*).

Notes:

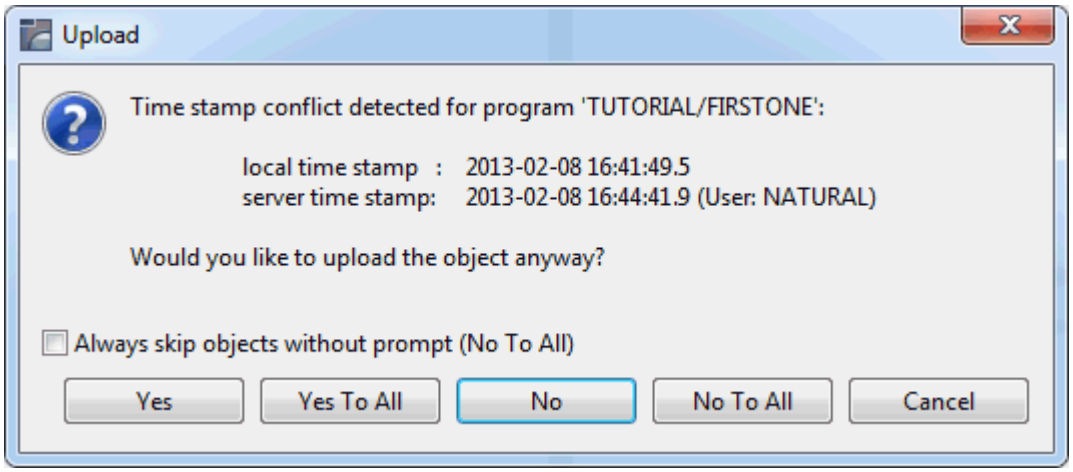
1. The time stamp of an object in the **Navigator** view or in the **Natural Navigator** view is shown in the properties of the object. See also ***Changing the Object Properties***.
2. The time stamps are not checked for objects in private-mode libraries. When the objects on a Natural server are stored in a private-mode library, a time stamp conflict cannot occur because the sources can only be modified by a single user via changes in the Natural project. For more information on private-mode libraries, see ***Steplibs*** in *Changing the Project Properties*.
3. The time stamps are not checked for **excluded** objects.
4. When you upload a data area to the server, the size of the data area on the server is approximately twice as large. The normalized data area (see also ***Editing Data Areas***) is stored as a comment in the source to avoid unnecessary content-related differences (such as empty lines and trailing blanks that might be missing after the normalization). **Caution:** When the source is later edited and saved on the server, the comment is automatically deleted. This causes a time stamp conflict afterwards.
5. When you upload a map to the server, the Map Editor generates a platform-specific header into the map source. **Caution:** When the source is later edited and saved on the server, the platform-specific header is changed. This causes a time stamp conflict afterwards.
6. In projects that are assigned to mainframe servers, the time stamps are not checked for error messages and Natural resources.

7. Due to the additional checks, a performance degradation may occur when updating the Natural environment if time stamp checking is enabled.

Time Stamp Conflicts During an Update of the Natural Environment

When the **Check time stamp on server** option is enabled in the preferences, the commands for updating the Natural environment (**Upload**, **Update**, **Build Natural Project** and **Rebuild Natural Project**) check for every object whether a time stamp conflict occurs. They do this before the object is updated in the Natural environment.

When the **Confirm time stamp conflict** option is also enabled in the Natural preferences, a dialog appears in case of a time stamp conflict, asking whether you want to upload the object anyway.



This dialog box offers the following options:

Command Button	Description
Yes	The time stamp conflict for the current object is ignored and the Natural environment is updated.
Yes To All	Same as **Yes**. In addition, this action is applied for all following time stamp conflicts.
No	The Natural environment is not updated with the current object.
No To All	Same as **No**. In addition, this action is applied for all following time stamp conflicts.
Cancel	The execution of the command is canceled.

The dialog box also contains the **Always skip objects without prompt (No To All)** check box. When you select this check box, **Confirm time stamp conflict** is automatically deselected in the preferences.

When the option **Confirm time stamp conflict** is not enabled in the Natural preferences, objects with time stamp conflicts are not updated in the Natural environment. This corresponds to **No** in the above dialog box.

All time stamp conflicts are collected in the **Time Stamp Conflicts** view (see **below**). In addition, a label decoration indicating the conflict is displayed in the **Navigator** view or in the **Natural Navigator** view. The following example shows this decoration for a program for which a conflict has been detected:

Note: The label decorations for the server problems are controlled by the preferences under **General > Appearance > Label Decorations**. If you do not want to have these label decorations, just go to the above mentioned preference page and deselect **Natural Compiler Problems**.

Resolving a Time Stamp Conflict

If a time stamp conflict occurs, it is your responsibility to find the differences and to resolve the conflict. You do this by looking at the contents of the local object and the object in the Natural environment.

When you decide to download the object from the Natural environment into your workspace, the object in the workspace receives the time stamp of the server and the conflict is resolved.

When you choose **Yes** or **Yes to All** in the dialog box (that is, when you ignore the conflict and update the Natural environment in spite of the conflict) the conflict is also resolved. Keep in mind that in this case it is no longer possible to compare the contents of the sources in the different environments. You should only update the Natural environment when you are absolutely sure that it is correct to overwrite the source with the content of your local source. It is therefore recommended that you first find out why the conflict has occurred - for example, you look at the code in the Natural environment, include any changed code from the Natural environment into your local source, resolve the conflict using the **Set Resolved** command and then build the project.

The **Set Resolved** command updates the time stamp of the object in the workspace with the time stamp of the object in the Natural environment. It also removes the label decoration in the **Navigator** view or in the **Natural Navigator** view which indicates the time stamp conflict.

To resolve a time stamp conflict in the Navigator view or Natural Navigator view

1. In the **Navigator** view or in the **Natural Navigator** view, select the object for which the time stamp conflict occurred.
2. Invoke the context menu and choose **NaturalONE > Set Resolved**.

To resolve a time stamp conflict in the Time Stamp Conflicts view

1. In the **Time Stamp Conflicts** view (see **below**), select the object for which the time stamp conflict occurred.

2 Invoke the context menu and choose **Set Resolved**.

Comparing the Time Stamps Before an Update of the Natural Environment

Before updating the Natural environment, you can compare the time stamps of the sources that are flagged for update in the **Navigator** view or in the **Natural Navigator** view with the time stamps of the sources in the Natural environment.

Note: The sources in the Natural environment are not locked when you compare the time stamps. Keep in mind that is possible that another user changes the source in the Natural environment between your time stamp comparison and your subsequent update.

To compare the time stamps

1 In the **Navigator** view or in the **Natural Navigator** view, select the object to be compared or the node containing the objects to be compared.

2 Invoke the context menu and choose **NaturalONE > Compare Time Stamps For >** *command*, where *command* can be one of the following, depending on your selection:

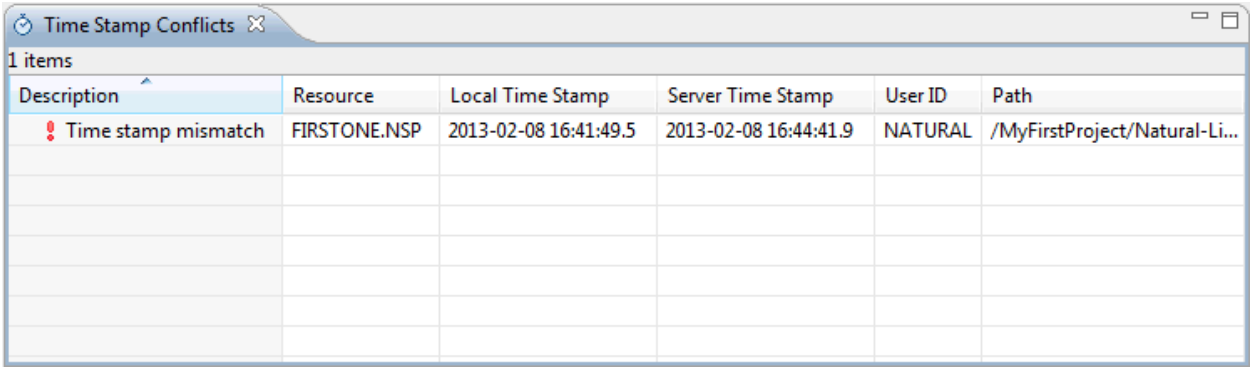
- **Upload** (compares the time stamps for all selected sources)
- **Build Natural Project** (compares the time stamps for new and changed sources of the current project)

Note: **Compare Time Stamps For** is only visible in the **NaturalONE** context menu if the **Check time stamp on server** option is enabled in the Natural preferences.

3 Go to the **Time Stamp Conflicts** view (see **below**) and check the output.

Time Stamp Conflicts View

The **Time Stamp Conflicts** view is not shown by default when you open the NaturalONE perspective. However, as soon as you enable the **Check time stamp on server** option in the Natural preferences, this view is automatically shown. If it is currently not shown, see *Showing a View of the NaturalONE Perspective*.



In the **Time Stamp Conflicts** view, "Time stamp mismatch" is shown for all sources which have changed in the Natural environment.

If an object in your workspace does not yet have a time stamp (for example, this is a new object or a Natural project has been checked out from a repository), the contents of the local source in the workspace is compared with the contents of the source in the Natural environment. If the sources are identical, no conflict occurs and the time stamp of the local source is set to the time stamp of the source in the Natural environment. If the sources are not identical, a time stamp conflict occurs. In this case, "No local time stamp - local source and server source are not identical" is shown in the **Time Stamp Conflicts** view.

For each object that is shown in the **Time Stamp Conflicts** view, you can invoke the compare editor of Eclipse. See *Using the Compare Editor* for further information.

Logging Time Stamp Conflicts with Command Line Arguments

You can start NaturalONE with special command line arguments. Using these arguments, you can write the time stamp conflicts for the objects in a project to a log file or, if no conflict occurs, update all objects of the project. In this case, NaturalONE is started as usual, with the Eclipse workbench user interface, but after handling the command line arguments, NaturalONE is shut down automatically. This feature works regardless of the setting of the Natural preference **Check time stamp on server**.

If you want to use this feature, you have to specify all of the following command line arguments:

`natural.TSProject "`*`project-name`*`"`
: Specifies the name of the Natural project for which time stamp checking and update is to be performed. This project must be located in the active workspace. If the project is not found in the workspace, an error is displayed in the **Error Log** view of Eclipse and NaturalONE is not shut down.

`natural.TSLogFile "`*`path-to-log-file`*`"`
: Specifies the path to the log file (including the log file name) in which all time stamp conflicts of the specified project are to be logged. If no time stamp conflicts are found, the log file is not created. If time stamp conflicts are found and a log file with the specified name already exists, it is overwritten (if no time stamp conflicts are found, an existing log file with the specified

name is deleted). If the directory specified in the path does not exist, an error is displayed in the **Error Log** view of Eclipse and NaturalONE is shut down.

`natural.TSCommand [ Compare | CompareAndUpdate ]`
Specifies the command to be executed. This can be one of the following:

Command	Description
`Compare`	Only compare the time stamps for all objects of the specified project.
`CompareAndUpdate`	Compare the time stamps for all objects of the specified project. If no time stamp conflicts are found, execute the **Update** command for all objects of the project.

If a wrong command is specified (that is, a command which is neither `Compare` nor `CompareAndUpdate`), an error is displayed in the **Error Log** view of Eclipse and NaturalONE is shut down.

The command line arguments are only executed if all of the above arguments are specified and if no error occurs in the definition of the arguments.

Like other Eclipse command line arguments, the NaturalONE arguments can be specified on the command line of your operating system or in the *eclipse.ini* file.

Example for the command line:

```
eclipse.exe natural.TSProject "My Sample Project" natural.TSLogFile ↩
"c:\temp\timestamp.log" natural.TSCommand CompareAndUpdate
```

Excluding Objects from Processing in the Natural Environment

You can exclude projects, folders, library folders or objects from being uploaded to the Natural environment. The names of the excluded items of a project are listed in the file *.excludes*. This file is stored in the root of a Natural project.

The *.excludes* file is created with the **Exclude** command (see below). Once this file has been created, it is possible to invoke an excludes editor (see below). Using the excludes editor, you can define to exclude, for example, specific Natural object types or files which match a particular pattern.

If you use the **Exclude** command in the **Natural Navigator** view, the content of the *.excludes* file depends on the type of node that you have selected before issuing the command:

- **Physical Node**
  If you select a node in the **Natural Navigator** view which physically exists in your Eclipse workspace, the path to the directory containing the files to be excluded is written to the *.excludes* file. If you add more objects to an excluded directory (for example, to a library), these objects are automatically excluded from processing.

- **Virtual Node**
  Virtual nodes in the **Natural Navigator** view are shown with a gray icon.

  When the objects in the **Natural Navigator** view are logically grouped according to their object types and you select the node for such a virtual group (for example, a **Programs** or **Subprograms** node), the name of each single object which is currently shown in that node is written to the *.excludes* file.

  On the other hand, when the folders which physically exist in your Eclipse workspace are not shown (because you have selected to show the objects of a library as they would be stored on the server) and you select a library node in the **Natural Navigator** view, the name of each single object which is currently shown in that node is written to the *.excludes* file.

  Any additional objects that will later appear in a virtual node are not automatically added to the *.excludes* file.

  Be careful when using the **Include** command, which removes the excluded objects from the *.excludes* file. If you include nodes in the **Navigator** view which have previously been excluded in the **Natural Navigator** view (or vice versa), the result might not be as expected.

Caution: You must not change *.excludes* file manually since this may result in damaging your project.

To exclude objects from processing in the Natural environment

1. In the **Navigator** view or in the **Natural Navigator** view, select the node(s) that you want to exclude from any processing in the Natural environment. This may be, for example, a Natural project, library, library folder or Natural object.
2. Invoke the context menu and choose **NaturalONE > Exclude**.

   In the **Navigator** view or in the **Natural Navigator** view, the label for each excluded node is shown with a gray color.

To include excluded objects in processing in the Natural environment

1. In the **Navigator** view or in the **Natural Navigator** view, select the previously excluded node(s) that you want to include again. This may be any node for which the label is currently shown with a gray color.
2. Invoke the context menu and choose **NaturalONE > Include**.

   In the **Navigator** view or in the **Natural Navigator** view, the label for each included node is no longer shown with a gray color.

To exclude objects using the excludes editor

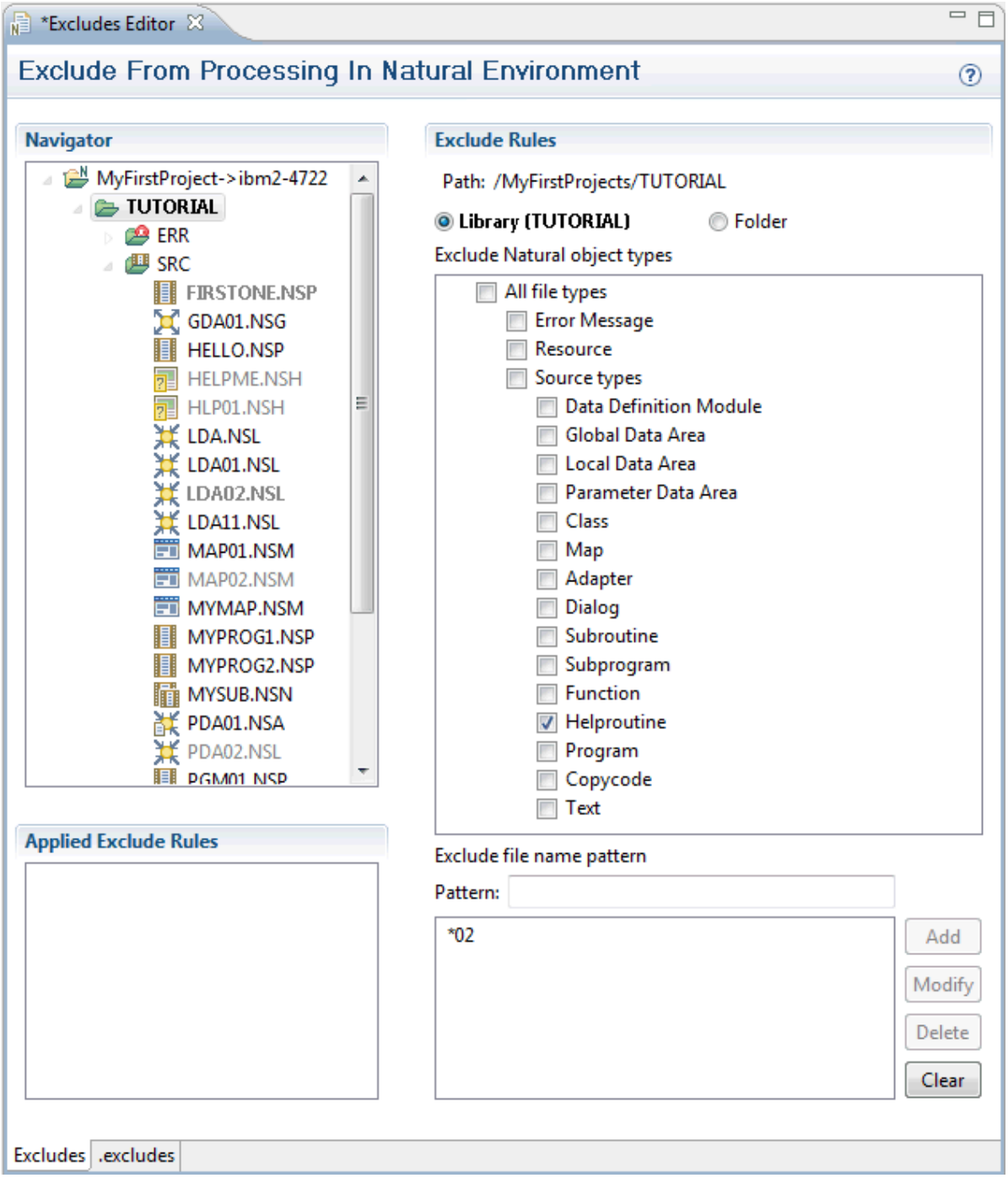
1 In the **Navigator** view or in the **Natural Navigator** view, go to the project in which you want to exclude objects and select the *.excludes* file.

2 Invoke the context menu and choose **Open With > Excludes Editor**.

The excludes editor is invoked. This is a multi-page editor which provides the following pages:

- **Excludes**
  Allows you to edit the *.excludes* file in a graphical way (as described below).
- **.excludes**
  Shows the actual content (code) of the *.excludes* file. It is not possible to modify the information on this page.

The **Navigator** section on the left side of the **Excludes** page shows the same tree which the **Navigator** view shows for a project. When you expand the nodes, all items which have been excluded are shown with a gray color. All nodes for which an exclude definition has been defined are shown in bold. When an exclude pattern has been defined for the selected node, this is shown at the bottom right. Example:



The **Exclude Rules** section on the right side of the **Excludes** page always shows the path for the selected library, folder or object. This is the path for which your exclude rules will be saved. A path is not shown when the project is selected; in this case, your exclude rules will apply to the entire project.

The option buttons **Library** and **Folder** specify the exclude range for the current path. When a library is selected, the **Library** option button is selected by default. In this case, your exclude

rule will be applied to the selected library, to all library folders that belong to the library, and to all subfolders (such as *SRC*) of the library. When the **Folder** option button is selected, your exclude rule will only be applied to the currently selected library folder and all of its subfolders; it does not apply to the entire library to which a library folder is assigned. When the label of the **Library** or **Folder** option button is shown in bold, this indicates that an exclude rule has been defined.

When you click a gray object in the **Navigator** section, the **Applied Exclude Rules** section indicates the node on which the corresponding exclude rule has been defined. For example, when an object name is shown, this means that an individual object has been excluded. Or when a project or library name is shown, this means that the exclude definition has been done on project or library level; when you click on the entry in the **Applied Exclude Rules** section, the corresponding definitions (including the path) are then shown in the editor.

3 On the **Excludes** page, you can exclude objects as follows:

- **To exclude a single object**
  Select a node in the **Navigator** section. A single check box is then shown in the **Exclude Rules** section. Select this check box to exclude the selected object.
- **To exclude all objects of a specific type**
  In the **Navigator** section, select the project, library or folder for which your exclude definition is to apply. The **Exclude Rules** section then shows check boxes for excluding special file types in the selected node. Select the check box for each type that you want to exclude.
- **To exclude all objects which match a particular pattern**
  In the **Navigator** section, select the project, library or folder for which your exclude definition is to apply. Enter a pattern of a file name in the **Pattern** text box and choose the **Add** button. Your pattern may contain the wildcards "?" and "*". For Natural object types and error messages, you must not enter an extension. If you do, the corresponding objects are not found. However, for Natural resources (that is, non-Natural files such as images or HTML files), you must enter the extension. Examples:

Pattern	Excludes the following files
*02	`LDA02.NSL`
02	`LDA02.NSL` and `PGM02OLD.NSP`
M????	`MAP01.NSM` and `MYPRG.NSP`
L??*	`LDA.NSL`, `LDA01.NSL`, and `LONGPRG.NSP`
N??APMSL	`N01APMSL.ERR`
*.PNG	All Natural resources with the extension PNG.

Using the **Modify** or **Delete** button, you can modify or delete an existing pattern which is currently selected in the list box. The **Clear** button deletes all patterns for the selected node.

4 Save your changes using the standard Eclipse functionality (for example, press CTRL+S).

In the **Navigator** view or in the **Natural Navigator** view, the label for each excluded node is now shown with a gray color.

Committing the Objects to the Repository of the Version Control System

You use the standard Eclipse functionality to commit the objects to your version control system.

It is important that you also keep the following in your version control system, in addition to the sources. Otherwise, your project will be corrupted.

Name	Type	Description
.project	File	Contains standard project information which is required by Eclipse.
.settings	Folder	Contains standard project information which is required by Eclipse.
.natural	File	Contains many settings which are written to the project when downloading libraries from a Natural environment or when changing the project properties.
.paths	File	Contains the mappings of the folder names to Natural library names.
.excludes	File	Contains all items which are to be excluded from any server operations.

The files *.paths* and *.excludes* are only available when the corresponding actions have been performed in your project. Therefore, these files need not necessarily exist in your project.

10 Understanding the Behavior of the Natural Builder

You can customize the NaturalONE environment to your specific needs. You can decide which of the Eclipse-specific build commands you want to use from the **Project** menu, and you can change the behavior of the Natural builder by changing the settings in the Natural preferences. How the Natural builder handles specific actions and commands depends on the following settings:

- The setting of the **Build Automatically** command in the **Project** menu of Eclipse (if disabled, the commands **Build All** and **Build Project** are enabled in the **Project** menu).
- The option **Build Natural projects automatically** in the Natural preferences.
- The option **Scratch server objects** in the Natural preferences.

For detailed information on the above-mentioned Natural preferences, see ***Natural > Builder*** in *Setting the Preferences*.

An overview is provided in the table below.

No.	Action	Build Automatically	Build Natural projects automatically	Scratch server objects	Results
1	Add object to workspace or modify existing object in workspace	On / Off	Off	On / Off	■ If the object is new, it is added to the Natural builder's object management. ■ The object is marked with an upload and stow flag (if appropriate). ■ Depending on the type of the object and the setting of the project's `PCHECK` parameter, other objects that call this object are marked with a catalog flag. ■ The **Build Natural Project** command of NaturalONE is enabled.
2		Off	On	On / Off	See the result for No. 1.

No.	Action	Build Automatically	Build Natural projects automatically	Scratch server objects	Results
3		On	On	On / Off	■ If the object is new, it is added to the Natural builder's object management. ■ The object is uploaded and stowed (if appropriate) in the Natural environment. ■ Depending on the type of the object and the setting of the project's `PCHECK` parameter, other objects that call this object are cataloged in the Natural environment. ■ The **Build Natural Project** command of NaturalONE is enabled.
4	Delete object from workspace	On / Off	On / Off	Off	■ The object is removed from the Natural builder's object management. ■ If no other object within the current Natural project has been modified, the **Build Natural Project** command of NaturalONE is disabled.
5		Off	On / Off	On	■ See the result for No. 4. ■ Depending on the type of the object and the setting of the project's `PCHECK` parameter, other objects that call this object are marked with a catalog flag. ■ The object waits to be scratched from the Natural environment.
6		On	Off	On	See the result for No. 5.
7		On	On	On	■ The object is removed from the Natural builder's object management. ■ The object is scratched from the Natural environment. ■ Depending on the type of the object and the setting of the project's `PCHECK` parameter, other objects that call this object are cataloged in the Natural environment.
8	Execute **Build Natural Project** command of NaturalONE	On / Off	Off	Off	■ All objects within the current Natural project that have an upload flag are uploaded to the Natural environment. If the upload was successful, the upload flag is removed. ■ All objects within the current Natural project that have a stow flag are stowed in the Natural

No.	Action	Build Automatically	Build Natural projects automatically	Scratch server objects	Results
					environment. If stowing was successful, the stow flag is removed. **Note:** An object is only stowed if the preceding upload was successful. ■ All objects within the current Natural project that have a catalog flag are cataloged in the Natural environment. If cataloging was successful, the catalog flag is removed. **Note:** A catalog flag is only shown if the object does not need to be uploaded.
9		On / Off	Off	On	■ See the result for No. 8. ■ The objects waiting to be scratched are scratched from the Natural environment.
10		Off	On	Off	See the result for No. 8.
11		Off	On	On	See the result for No. 9.
12		On	On	On / Off	The **Build Natural Project** command of NaturalONE is disabled.
13	Execute **Build Project** command in **Project** menu of Eclipse	On	On / Off	On / Off	The **Build Project** command in the **Project** menu is disabled.
14		Off	Off	On / Off	The **Build Project** command in the **Project** menu has no effect on Natural projects.
15		Off	On	Off	See the result for No. 8.
16		Off	On	On	See the result for No. 9.
17	Execute **Clean** command in **Project** menu of Eclipse	On / Off	Off	Off	■ All objects are removed from the Natural builder's object management. ■ All objects are added to the Natural builder's object management. ■ The upload, stow and catalog flags are preserved.
18		On / Off	Off	On	■ All objects are removed from the Natural builder's object management. ■ All objects are added to the Natural builder's object management. All objects within the current Natural project are marked with a scratch, upload and stow flag (if appropriate).

No.	Action	Build Automatically	Build Natural projects automatically	Scratch server objects	Results
19		On / Off	On	Off	■ See the result for No. 17. ■ See the result for No. 8.
20		On / Off	On	On	■ All objects are removed from the Natural builder's object management. ■ All objects are added to the Natural builder's object management. ■ All objects within the current Natural project are scratched from the Natural environment. ■ All objects within the current Natural project are uploaded to Natural environment. If the upload fails, the upload flag is not removed. ■ All objects within the current Natural project are stowed (if appropriate) in the Natural environment. If stowing fails, the stow flag is not removed. **Note:** An object is only stowed if the preceding upload was successful.
21	Execute **Upload** command of NaturalONE	not applicable	not applicable	not applicable	■ All selected objects are uploaded to the Natural environment. ■ If the upload was successful and if flags where previously shown for the objects, the upload and scratch flags of the uploaded objects are removed.
22	Execute **Update** command of NaturalONE	not applicable	not applicable	not applicable	■ All selected objects are uploaded to the Natural environment and stowed (if appropriate) in the Natural environment. **Note:** An object is only stowed if the preceding upload was successful. ■ If the upload and stowing/cataloging was successful and if the corresponding flags where previously shown for the objects, the upload, stow/catalog and scratch flags of the uploaded objects are removed.

III Working with Natural Objects in Natural Server Mode

In the so-called Natural server mode, you edit a Natural object directly on a mapped Natural server. See also *Different Modes for Developing Natural Applications* in the *Introduction*.

If you choose to work in Natural server mode, you use the **Natural Server** view to edit your Natural objects. You can also check, stow, catalog and execute Natural objects in this view. If you are working in parallel with other developers on the same library, the possibility to unlock locked objects may also be helpful for you.

Important: Keep in mind that the preferred way for developing or maintaining Natural applications is working in local mode (not in Natural server mode). This means that the entire project has been offloaded from the Natural server to the Eclipse workspace. If you have chosen to work in local mode, the **Natural Server** view is only required if you want to **download** libraries and objects to your Eclipse workspace. See ***Working with Natural Projects in Local Mode*** for further information.

This part describes how to manage Natural objects in the **Natural Server** view. It covers the following topics:

Accessing a Remote Development Environment

Managing Objects Directly on a Natural Server

Launching Natural Applications

Using the Natural Command Console for Mainframes

11 Accessing a Remote Development Environment

- Mapping a Natural Environment 142
- Dynamically Changing the CICS Transaction Name when Starting a Session 145
- Contents of the Natural Server View 145
- Filtering Libraries and Objects 147
- Properties for the Different Nodes 153
- System Information for a Natural Environment 153
- Unmapping a Natural Environment 158

Mapping a Natural Environment

To perform development directly on a Natural server on which Natural Development Server (NDV) is installed, you have to activate a Natural server environment. You do this by mapping the appropriate server in the **Natural Server** view. Each server provides all remote services (such as access or update) for a specific `FUSER` (Natural system file for user programs).

If you want to connect to a Natural environment for the first time, you have to map it as described below. Once you have mapped an environment, a node for this environment is automatically shown in the **Natural Server** view. It is possible to map the same environment more than once, for example, if you want to have Natural server sessions with different session parameters.

Note: When mapping to an environment, especially with Natural security installed, the `ETID` parameter should be set to `'  '` (Mainframe) or `$$` (UNIX). Otherwise an error might occur that the user with the same ETID is already active. This error is often raised during a refresh of a NaturalONE Server environment where multiple connections are internally being set up in order to perform the **Refresh** operation.

To map a Natural environment

1 Go to the **Natural Server** view.

2 Invoke the context menu and choose **Map**.

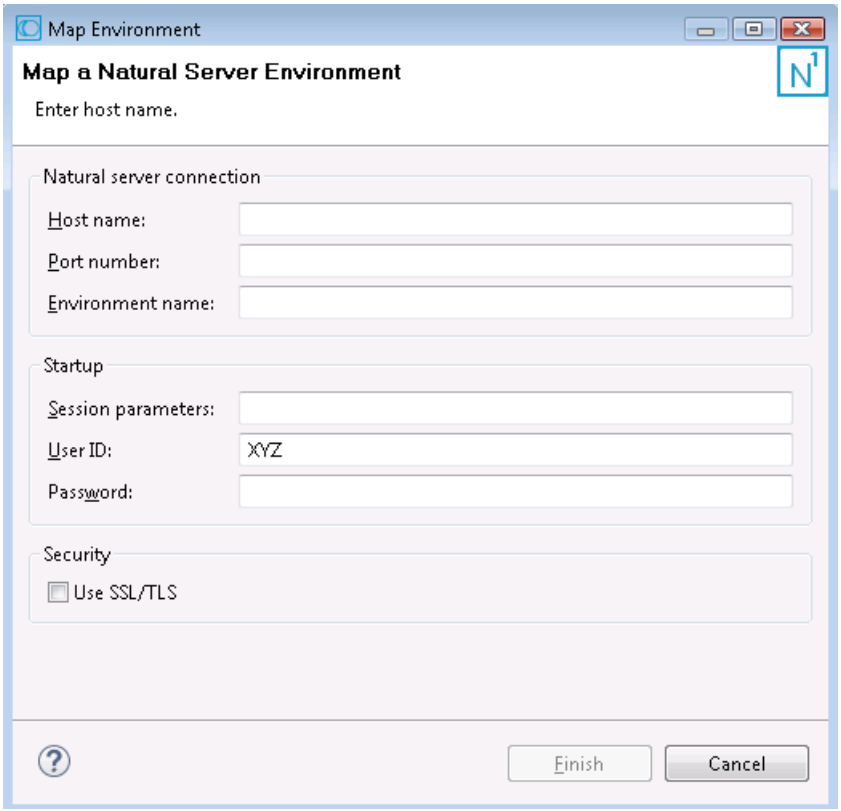
Or:

Choose the following icon in the local toolbar:

Or:

Press CTRL+ALT+M.

The following dialog box appears.



3 Specify the following information:

Option	Description
Host name	The name of the Natural server.
Port number	The TCP/IP port number of the Natural server.
Environment name	The name that is to appear in the **Natural Server** view. A default name will be created automatically. If you want, you can enter a more specific environment name.
Session parameters	Optional. If dynamic parameters are required for the Natural environment, specify them in this text box. **Tip:** For a mainframe environment, it is recommended that you specify `TMODEL=2`. Otherwise (with the default setting), the output in the **Natural I/O** window is shown with a very small font. **Note:** 1. When connecting to a Natural Development (NDV) server on a Linux, UNIX or Windows platform, you can use the predefined NDVPARM session parameter.

Option	Description
	2. When connecting to a Natural server on a Linux, UNIX or Windows platform, in order to use the **Predict Description and Generation** optional component of NaturalONE, you can use the predefined PRFPARM session parameter.
User ID	The user ID that is to be used for mapping the Natural environment. This text box is initially blank. When you have previously mapped an environment, the user name that you entered the last time is automatically provided.
Password	Optional. If Natural Security is active on the Natural server, specify the required password in this text box.
Use SSL/TLS	Optional. Must be enabled when a connection to an SSL/TLS-secured Natural Development (NDV) Server is performed.

Note: If you do not know the host name and port number for your Natural server, ask your administrator.

4 Choose the **Finish** button.

A node for the specified environment is now shown in the **Natural Server** view.

It is also possible to map a Natural environment from an existing Natural project located in the **Navigator View** or the **Natural Navigator View**. In this case the runtime properties of the selected Natural project are used as the mapping parameters.

To map a Natural environment from an existing Natural project

1 Go to either the **Navigator View** or the **Natural Navigator View**.

2 Select a single or multiple Natural project(s).

3 Invoke the **NaturalONE** context menu and choose **Map**.

Or:

Press CTRL+ALT+M.

A node for each specified environment is now shown in the **Natural Server** view.

Dynamically Changing the CICS Transaction Name when Starting a Session

The following description applies if you want to switch to a different CICS transaction on a mainframe.

You specify the CICS transaction name in the same text box in which you also specify the dynamic parameters for the Natural environment. So that the CICS transaction name can be evaluated, it is important that you specify it before any Natural parameters, using the following syntax:

```
<TA_NAME=name>
```

where *name* can be 1 to 4 characters long. This must be the name of an existing CICS transaction which applies to a CICS Adapter. It will override the transaction name which is currently defined in the configuration file for the CICS Adapter on the Natural Development Server (NDV). Ask your administrator for further information.

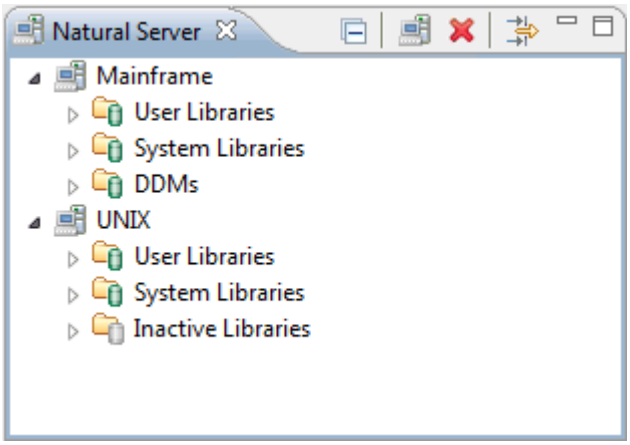
Make sure to put the entire definition in angle brackets. When this definition is followed by a Natural parameter, insert a blank before the Natural parameter. Example:

```
<TA_NAME=NA82> STACK=(LOGON SYSCP)
```

If the specified CICS transaction name cannot be found, an error message occurs and the session cannot be started.

Contents of the Natural Server View

The **Natural Server** view provides the logical representation of one or more Natural environments. The following example shows two Natural server nodes, one for a server on a mainframe and another for a server on UNIX. The names that are shown for the server nodes are the environment names that have been defined when the environments were mapped.



Specific top-level nodes are provided for user libraries and system libraries (also known as the `FUSER` and `FNAT` system files). Other top-level nodes correspond to the Natural architecture in the mapped environment. For example:

- In a mainframe environment, DDMs are always stored in the system file `FDIC`. Therefore, a **DDMs** node is available.

  In a UNIX, OpenVMS or Windows environment, DDMs are normally stored in libraries. However, if the parameter `FDDM` has been set, the DDMs are stored in the system file `FDDM`. In this case, a **DDMs** node is also available.
- Inactive libraries are only available in UNIX, OpenVMS and Windows environments. Therefore, such a node is not provided for a mainframe environment.

Alias names can be shown for the system files in a UNIX, OpenVMS or Windows environment. They are shown when they have been defined with Natural on the appropriate platform.

When you expand the node for a system file, the nodes for the libraries in this system file are shown. For detailed information on the system files and library types, see the Natural documentation for the corresponding platform.

The objects in a library are grouped into different nodes, according to their Natural object types. For example, all programs are shown in a node called **Programs**. Thus, if you want to view the available programs in a library, you have to expand the **Programs** node.

For subroutines, functions, classes and DDMs, the long names (which may exceed 8 characters) are shown. These are the names that have been defined in the program; the names that have been defined when the object has been saved are not shown for these objects.

If Natural Security is active, only the allowed libraries and objects are shown. In addition, the commands which are not allowed to be used are disabled in the context menus.

You can use the button in the local toolbar to collapse all expanded nodes.

Notes:

1. For an overview of the icons that are used for the different types of objects in the **Natural Server** view, see *Types of Natural Editors*.
2. Further information on the **Natural Server** view is provided in the following topics: *Downloading an Existing Library or Object from a Natural Server* and *Working with Natural Objects in Natural Server Mode*.
3. When optional components of NaturalONE have been installed, additional top-level nodes may be shown for a mapped environment in the **Natural Server** view. For example, when **Service Development** has been selected in the installer, a node with the name "Business-Services" is shown. For detailed information on how to use such a node, see the documentation for the corresponding optional component.

4. When a huge number of objects are contained in a library on the mainframe, expanding an object node can take a long time. Natural for Mainframes on z/OS provides a hyperdescriptor that can significantly improve the database access required for this purpose. For further information, see *Performance Aspects* in *NaturalONE in a Nutshell*.

Filtering Libraries and Objects

Using a filter, you can reduce the number of items that are shown in the **Natural Server** view. Filtering involves several steps: First you define a filter and then you apply the filter to a system file node or to a library node. Detailed information is provided in the following topics:

- Defining a Filter
- Setting a Filter
- Removing a Filter or Pattern

Defining a Filter

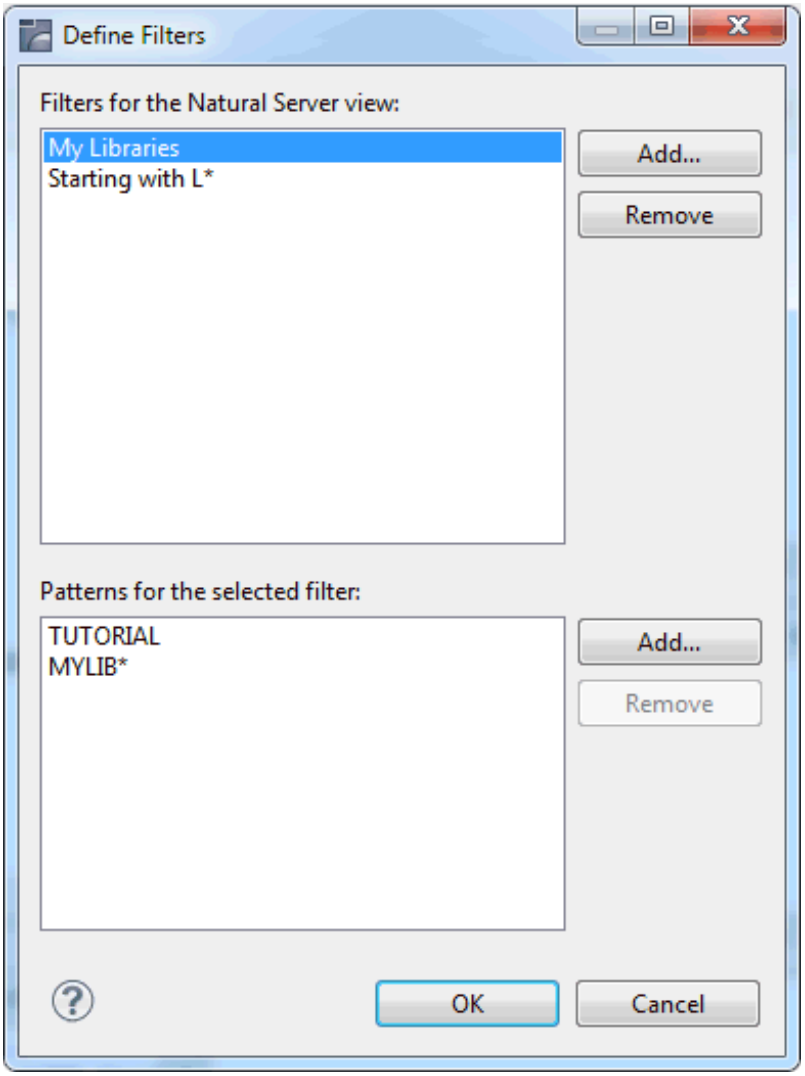
When you define a filter, you specify a pattern (for example, that only items are to be shown which start with the letter "L"). Each filter can hold an unlimited number of patterns.

The filters that you define can be applied to all mapped Natural environments.

To define a filter

1 Choose the following icon in the local toolbar of the **Natural Server** view:

The **Define Filters** dialog box appears.



When filters are already defined, they are shown in the upper part of the dialog box. The lower part of the dialog box shows the patterns, if defined, for the selected filter.

Note: This dialog box can also be invoked from the **Set Filters** dialog box (see *Setting a Filter*).

2 To define a new filter, choose the **Add** button in the upper part of the dialog box.

3 Enter the name for the filter in the resulting dialog box.

You can use any name for the filter (for example, "My Libraries").

4 Choose the **OK** button to close the dialog box in which you have defined the name of the filter.

Now, you have to define one or more patterns for the new filter.

5 Make sure that the filter is selected in upper part of the dialog box.

6 Choose the **Add** button in the lower part of the dialog box.

7 Enter the filter pattern in the resulting dialog box. A pattern is defined as follows:

- You can enter the names of all libraries or objects that are to be shown. All names must be separated by a semicolon.

  Note: Instead of entering all filter criteria in one pattern, you can also enter them by adding several patterns.

- You can use wildcards (? or *) within the names if you do not want to enter each name individually. The question mark (?) may be specified at any position inside the name. The asterisk (*) is only allowed at the end of a name.
- You can also enter a range of names. Ranges must be entered as follows:

  ```
  name1 - name2
  ```

  When defining a range, it is important that you enter a space before and after the hyphen. The spaces are necessary since names may contain hyphens. Without the spaces, `name1-name2` would be interpreted as the name of a single library or object. Each name in a range definition may contain wildcards (see above). Example:

  ```
  AL* - AM?TEST
  ```

8 Choose the **OK** button to close the dialog box in which you have defined the pattern.

9 In the **Define Filters** dialog box, choose the **OK** button to save your changes and to close the dialog box.

Setting a Filter

Each defined filter can be applied to a system file node or to a library node of a mapped Natural environment:

- When you select a system file node (for example, **User Libraries**), you can reduce the number of libraries that are shown for this system file.
- When you select a library node, you can reduce the number of objects that are shown in this library.

When a filter has been applied, the icons that are shown for the nodes contain an additional plus sign:

	System file with an active filter.
	Library with an active filter.

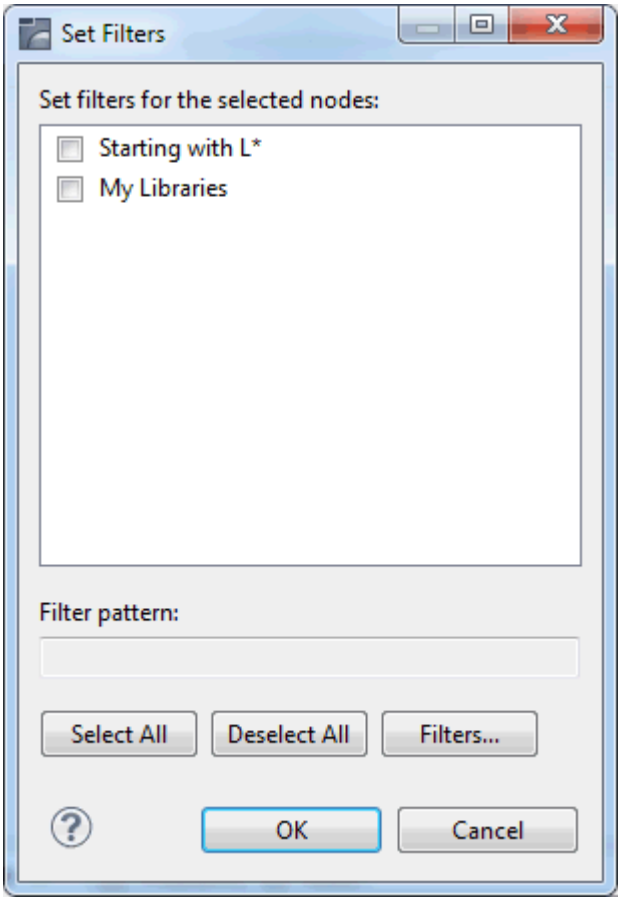
Note: The label decorations for the filters are controlled by the preferences under **General > Appearance > Label Decorations**. If you do not want to have label decorations for the filters, just go to the above mentioned preference page and deselect **Natural Server View Filter**.

All filters that you set for a mapped Natural environment are stored with the environment name that is shown in the **Natural Server** view. For example, when you have defined "MyMainframe" as the environment name and you unmap this environment, the filters will be set again when you map the same environment once more with the same environment name (that is, with the name "MyMainframe"). If you map the same environment with a different environment name (for example, with the name "MyServer"), the filters that you have set for "MyMainframe" are not considered.

To set a filter

1. In the **Natural Server** view, select the node(s) for which you want to set a filter (either a system file node or a library node).
2. Invoke the context menu and choose **Set Filter**.

   The **Set Filters** dialog box appears. It lists all filters that are currently defined.



3 Activate the check box for each filter that you want to set.

You can also use the following command buttons:

Command Button	Description
Select All	Activates all filters in the dialog box.
Deselect All	Deactivates all filters in the dialog box.
Filters	Invokes the **Define Filters** dialog box in which you can define additional filters or change the existing filters. See *Defining a Filter*.

Note: When you select a filter (not the check box), the pattern for this filter is shown at the bottom of the dialog box.

4 Choose the **OK** button.

When an automatic refresh has been defined in the Natural preferences (see *Runtime Execution* in *Setting the Preferences*), the content of the **Natural Server** view changes automatically so that just the items that match your filter pattern are shown (depending on your selection).

When an automatic refresh has not been defined in the Natural preferences, you have to refresh the display manually. See *Refreshing the Display*.

Notes:

1. If a filter is changed in the **Define Filters** dialog box and if this filter is already used, an automatic refresh does not occur.
2. The group nodes for a library (for example, **Programs**) are always shown, even if all objects of the corresponding type are filtered out.

Removing a Filter or Pattern

You can either deactivate a filter for a node or you can remove a filter completely so that it is no longer available.

You can also remove a single pattern if you no longer need it.

To deactivate a filter

1 In the **Natural Server** view, select the node(s) for which you want to deactivate a filter.

2 Invoke the **Set Filters** dialog box as described above (see *Setting a Filter*).

3 Deselect the check box for each filter that you want to deactivate.

4 Choose the **OK** button.

To remove a filter

1 Invoke the **Define Filters** dialog box as described above (see *Defining a Filter*).

2 Select the filter that you want to delete.

3 Choose the **Remove** button in the upper part of the dialog box.

Note: When the selected filter is currently active, you are asked whether you really want to delete the filter.

To remove a pattern

1 Invoke the **Define Filters** dialog box as described above (see *Defining a Filter*).

2 Select the filter which contains the pattern that you want to delete.

3 Select the pattern that you want to delete.

4 Choose the **Remove** button in the lower part of the dialog box.

Properties for the Different Nodes

When you select a node in the **Natural Server** view, the corresponding properties are automatically shown in the **Properties** view. The information that is shown in the **Properties** view depends on the type of node that is currently selected:

- **Natural Environment**
  Information on the selected Natural environment is shown. This includes the mapping information.

  Note: More detailed information on a Natural environment is provided in the **Properties** dialog box. See *System Information for a Natural Environment*.

- **System File, Library or Object Type**
  Information on the selected system file, library or object type is shown, for example, database ID, file number, the number of different objects (such as sources and cataloged objects) and their sizes.
- **Object**
  Information on the selected object is shown, for example, short name and long name, programming mode, encoding, and date and time when the source was last modified or cataloged.

Brief information on the selected node is also shown in the status line of the Eclipse window. For example, when a Natural object is selected, the following information is shown: environment name (with host name and port number), system file name (with database ID and file number), library name, object type (for example, "P" for program), and object name.

Note: You can also invoke a **Properties** dialog box for a node, using the **Properties** command from the context menu.

System Information for a Natural Environment

In the **Natural Server** view, you can display detailed system information for each mapped Natural environment.

To display system information for a Natural environment

1. In the **Natural Server** view, select the top-level node for a mapped Natural environment.
2. Invoke the context menu and choose **Properties**.

   The **Properties** dialog box appears, showing the system information for this environment.

3 In the tree on the left side of the dialog box, choose the type of system information that you want to display.

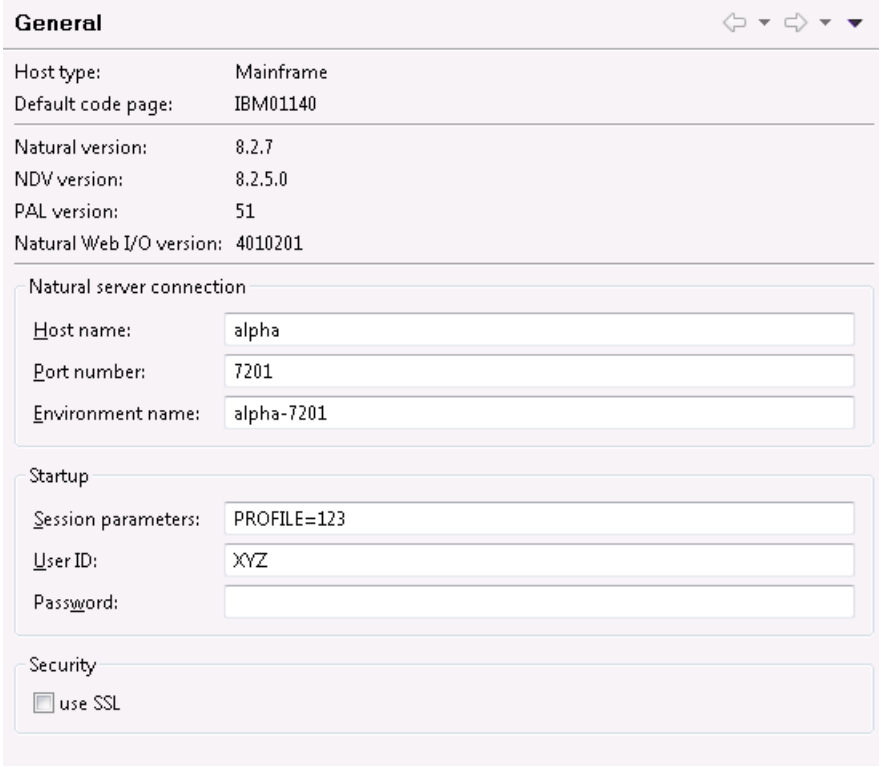
Information on the resulting page is provided in the topics below.

- General Information
- Product Information (SYSPROD)
- System Files (SYSPROF)
- Work and Print Files (SYSFILE)

Note: The information provided in the **Properties** dialog box corresponds to the output of the Natural system commands `SYSPROD`, `SYSPROF` and `SYSFILE`.

General Information

When you choose **General** in the tree, general information about the current environment is shown.



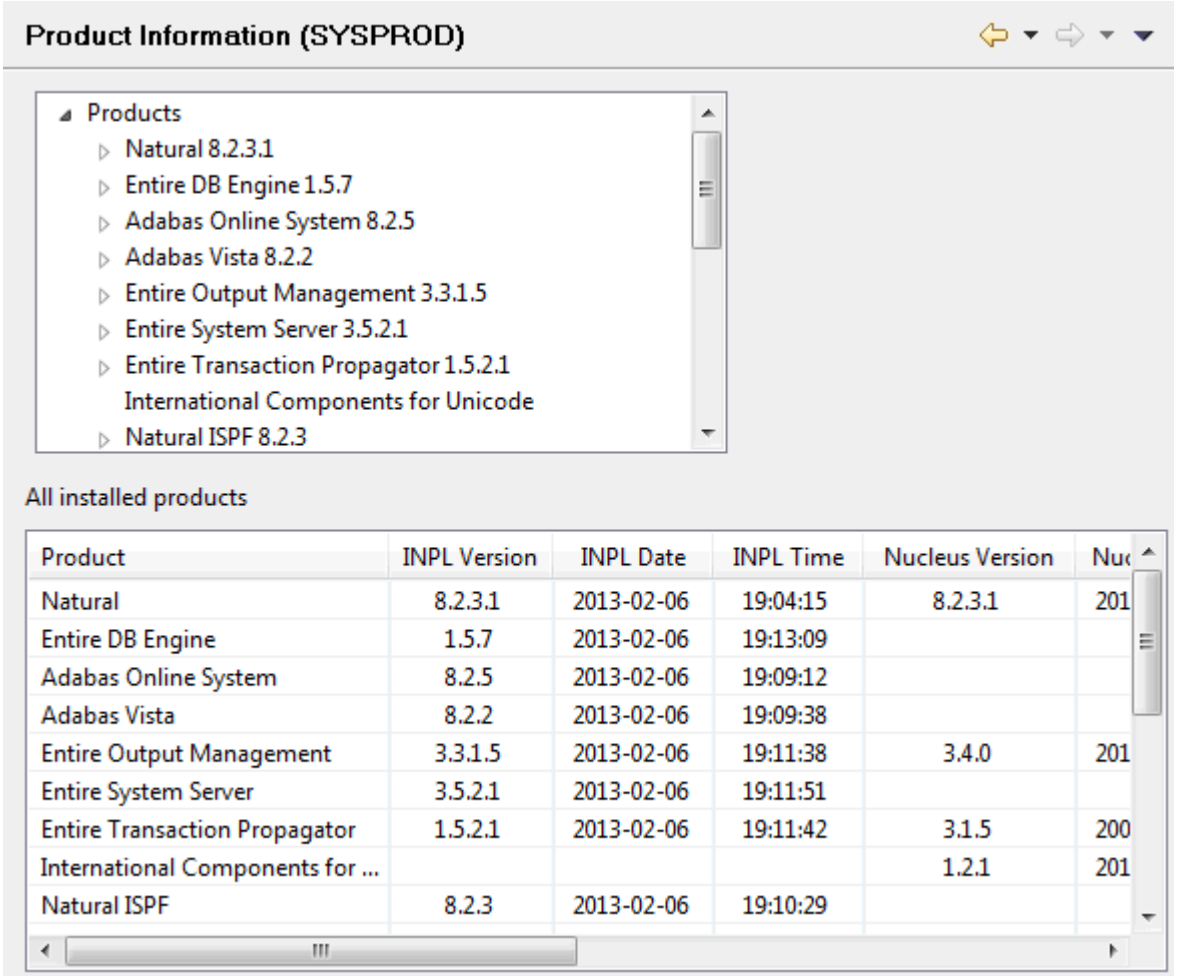
This information includes, for example, the Natural Development Server (NDV) version and the name of the default code page.

If you want to change the mapping (for example, in order to specify different session parameters or a different user ID and password), you can do this here. See *Mapping a Natural Environment* for information on the options that can be specified.

Note: General information is also shown in the **Properties** view when the node for a mapped Natural environment is selected. However, it is not possible to modify information in this view. See also *Properties for the Different Nodes*.

Product Information (SYSPROD)

When you choose **Product Information (SYSPROD)** in the tree, a list of all products that are installed in the Natural environment is shown.



Two different areas are available, a tree and a table. The information that is shown depends on the selected Natural environment.

- The tree at the top lists all installed products in the Natural environment.

  When you move the mouse pointer over a product name in the tree, a tool tip appears, providing information about this product. The same information is shown in the table at the bottom when you select the product in the tree.

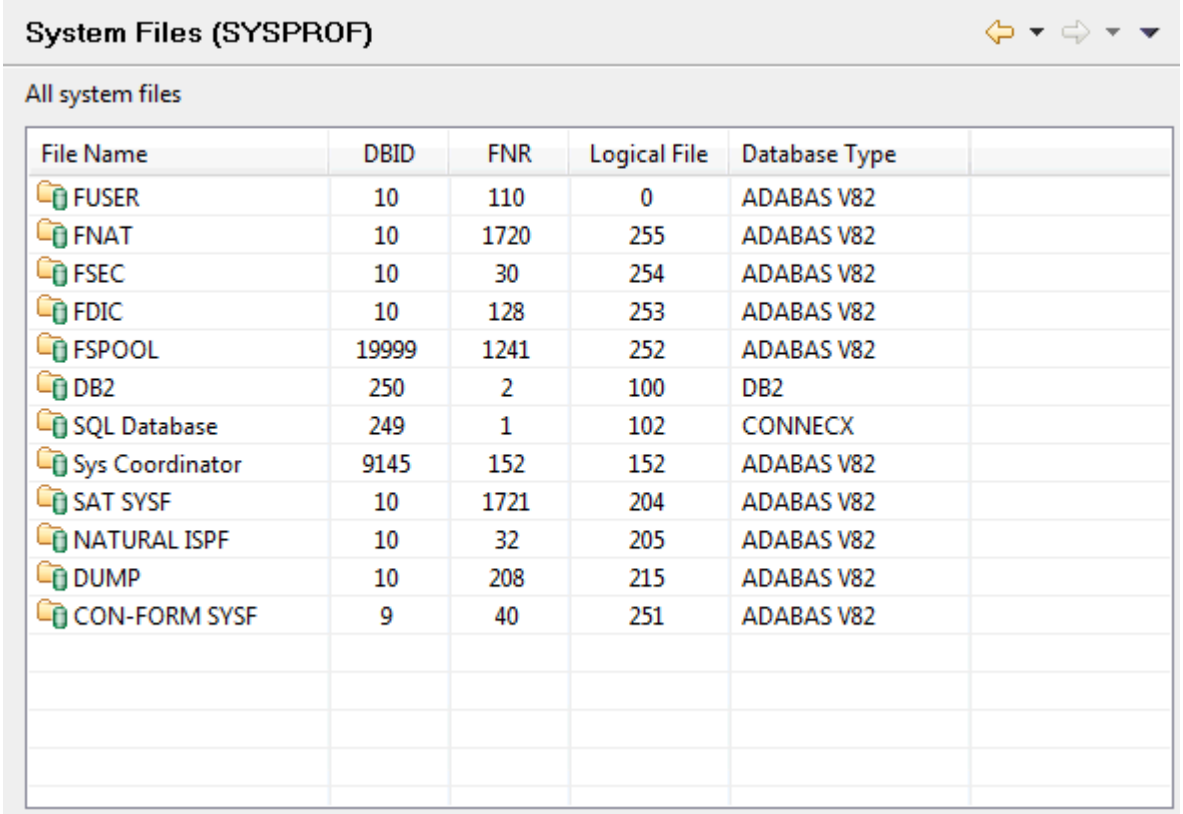
For some products, it is possible to expand the product node in the tree:

- **Mainframe environments**
  You can display information on history records and/or subcomponents.
- **Windows, UNIX and OpenVMS environments**
  You can display information on hotfixes.

- The table at the bottom shows information about the entry which is currently selected in the tree. For example:
  - When you select the top-level entry **Products** in the tree, a list of all installed products is shown in the table. In this case, you can see the details for all products (such as INPL version number, INPL date and product ID) at a glance.
  - When you select a product in the tree, only the details for the selected product are shown in the table.
  - When you select, for example, a history record in the tree, the corresponding information is shown in the table.

System Files (SYSPROF)

When you choose **System Files (SYSPROF)** in the tree, the current assignments for all Natural system files in the Natural environment are shown.



System Files (SYSPROF)

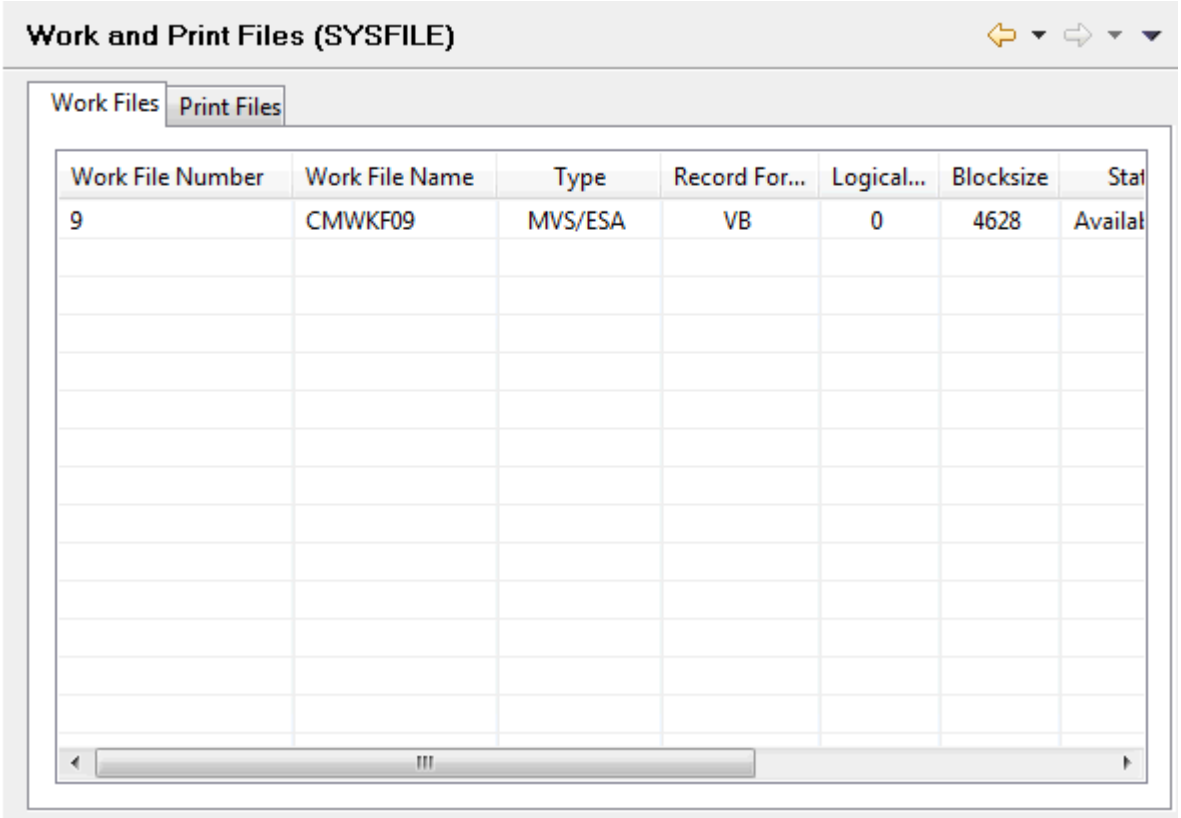
All system files

File Name	DBID	FNR	Logical File	Database Type
FUSER	10	110	0	ADABAS V82
FNAT	10	1720	255	ADABAS V82
FSEC	10	30	254	ADABAS V82
FDIC	10	128	253	ADABAS V82
FSPOOL	19999	1241	252	ADABAS V82
DB2	250	2	100	DB2
SQL Database	249	1	102	CONNECX
Sys Coordinator	9145	152	152	ADABAS V82
SAT SYSF	10	1721	204	ADABAS V82
NATURAL ISPF	10	32	205	ADABAS V82
DUMP	10	208	215	ADABAS V82
CON-FORM SYSF	9	40	251	ADABAS V82

For each system file, detailed information such as database ID and file number is provided. The information that is shown depends on the selected Natural environment. The above example shows the system files in a mainframe environment. For Windows, UNIX and OpenVMS environments, additional entries for inactive system files can also be shown.

Work and Print Files (SYSFILE)

When you choose **Work and Print Files (SYSFILE)** in the tree, information about work files and printer settings is shown.



Several tabs are provided. The information that is shown depends on the selected Natural environment. The above example shows the tabs for a mainframe environment (work files and print files). In a Windows, UNIX and OpenVMS environment, tabs are provided for work files, reports and logical devices.

Unmapping a Natural Environment

When you unmap a Natural environment, its node is removed from the **Natural Server** view.

Note: Any editor windows that have been opened for an environment are not closed when unmapping this environment. Even though the environment has been unmapped, it is still possible to save to this environment any modifications that you make in the editor windows.

To unmap a Natural environment

1 In the **Natural Server** view, select the node for the Natural environment that you want to unmap.

2 Invoke the context menu and choose **Unmap**.

Or:

Choose the following icon in the local toolbar:

12 Managing Objects Directly on a Natural Server

- Editing Objects 160
- Listing Objects 161
- Saving Objects 161
- Checking Objects 162
- Stowing Objects 162
- Cataloging Objects 163
- Executing Objects 163
- Refreshing the Display 164
- Copying and Moving Objects and Libraries 164
- Renaming Objects and Libraries 167
- Deleting Objects and Libraries 168
- Unlocking Locked Objects 168
- Working with DDMs 171
- Working with Dialogs 172
- Working with Resources 173
- Working with Private-mode Libraries 173

Editing Objects

When you edit a source directly on a Natural server, a lock is applied in order to prevent concurrent updating of objects, the source is automatically downloaded into a temporary project in the Eclipse workspace and the editor is opened. You can then apply changes to the source code and save them; the source is then changed on the server. When you close the editor, the object is automatically unlocked and removed from the temporary project. When the temporary project no longer contains any objects, it is removed from the Eclipse workspace.

The following information applies to Natural for Windows, UNIX and OpenVMS. It also applies to Natural for Mainframes when the profile parameter `SLOCK` has been set to "SPOD".

- On a Natural server, locking information is kept in the development server file (`FDIC`). Therefore, locking is only possible when such a file exists on the server.

  Note: Natural for Mainframes only: `SLOCK=PRE` is the recommended setting when working in mixed environments. In this case, locking information is kept in `FUSER` or `FNAT` (depending on where the source member to be edited is located).

- All Natural servers must have a development server file. An exception is a Windows server which can be run with or without development server file.
- If a Windows server runs without development server file, NaturalONE never locks any objects. When an object is edited on a Windows server using Natural Studio, the object is locked and can therefore not be edited or deleted with NaturalONE. However, when you edit an object on a Windows server using NaturalONE, the object will not be locked and can thus be edited or deleted by any other user. It is also possible to delete a file on a Windows server which is currently edited in the same NaturalONE session; however, when the object is saved in the editor, the object is stored again on the Windows server.
- If a server runs with a development server file, the above described situation cannot occur.

To edit objects

1. In the **Natural Server** view, select the object(s) that you want to edit.
2. Invoke the context menu and choose **Edit**.

   Or:

   Double-click each object that you want to edit.

   An editor window appears for each selected object and you can now edit the sources. See *Using the Natural Editors* for further information.

When an object is currently locked, an editor is not invoked. Instead, the following information is shown: the date and time when the object was locked, and by whom it was locked.

Note: For information on how to edit resources, see *Working with Resources*.

Listing Objects

When you "list" an object, you display its contents but you cannot modify it. However, you can copy its contents. The object is automatically downloaded into a temporary project in the Eclipse workspace and the editor is opened in read-only mode. This is helpful, for example, if you want to view the contents of an object that you are not allowed to modify, or if you want to view the contents of an object and do not want to lock the object (as done with the **Edit** command).

To list objects

1 In the **Natural Server** view, select the object(s) that you want to list.

2 Invoke the context menu and choose **List**.

An editor window appears for each selected object.

Saving Objects

Object sources are saved using the standard Eclipse functionality. The object source is saved directly on the Natural server.

To save an object

1 Activate the editor window for the source that you want to save.

2 From the **File** menu, choose **Save**.

Or:

Press CTRL+S.

Checking Objects

You can check the source code of an object (except classes) for syntax errors.

Note: When the option **Console output** is enabled on the **Runtime Execution** page of the Natural preferences, information about success or failure of the **Check** command is shown in the **Console** view.

To check objects

1 In the **Natural Server** view, select the object(s) that you want to check.

2 Invoke the context menu and choose **Check**.

If an error was found, a dialog box appears, providing information on the error.

Stowing Objects

When you stow an object, it is compiled and saved. If no errors are found, the source form of the object is saved and the resulting generated program is stored (in addition to the source).

Note: When the option **Console output** is enabled on the **Runtime Execution** page of the Natural preferences, information about success or failure of the **Stow** command is shown in the **Console** view.

To stow the current object in the editor

1 Activate the editor window for the source that you want to stow.

2 Invoke the context menu and choose **Stow**.

Or:

Press CTRL+T.

If an error is found, information on the error is shown in the **Problems** view. In this case, you first have to correct the error before it is possible to stow the object.

To stow several objects

1 In the **Natural Server** view, select the object(s) that you want to stow.

2 Invoke the context menu and choose **Stow**.

If an error is found, a dialog box appears, providing information on the error. In this case, you first have to correct the error before it is possible to stow the object.

Cataloging Objects

When you catalog an object, it is compiled. If no errors are found, the resulting generated program is stored.

In contrast to the **Stow** command, the source code is not saved when you catalog an object. When the editor is open and the latest changes have not yet been saved, they are not considered by the **Catalog** command.

Note: When the option **Console output** is enabled on the **Runtime Execution** page of the Natural preferences, information about success or failure of the **Catalog** command is shown in the **Console** view.

To catalog objects

1 In the **Natural Server** view, select the object(s) that you want to catalog.

2 Invoke the context menu and choose **Catalog**.

If an error was found, a dialog box appears, providing information on the error. In this case, you first have to correct the error before it is possible to catalog the object.

Executing Objects

When you execute an object, the default configuration settings in the Natural preferences are used (see ***Natural I/O > Runtime*** in *Setting the Preferences*).

To execute objects

1 In the **Natural Server** view, select the object that you want to execute.

Note: It is only possible to execute one object at a time.

2 Invoke the context menu and choose **Execute**.

The output is either shown in the internal browser or in an external browser, depending on your launch configuration.

Refreshing the Display

Usually when something changes in Natural, the **Natural Server** view is automatically refreshed. This happens, for example, when objects are created, renamed or deleted. The automatic refresh requires that the corresponding option has been set in the Natural preferences; see ***Runtime Execution*** in *Setting the Preferences*.

There are, however, situations where an automatic refresh does not take place, since Natural is not aware of a modification. For example, two Natural processes are currently active and both are working on the same system file. When one Natural is applying a change to the system file (such as creating a new object), the second Natural is not aware of this modification. In this case, you have to refresh the display manually.

To refresh the display manually

1 In the **Natural Server** view, select the node to be refreshed (for example, a library).

2 Invoke the context menu and choose **Refresh**.

   Or:

   Press F5.

   The most recent information is fetched from the Natural server. The tree remains expanded at the same place.

Copying and Moving Objects and Libraries

You can copy and move nearly all of the nodes in the **Natural Server** view. This can be either a group node or a node for a single object. For example, you can copy all programs of a library by copying the **Program** node. The following restrictions apply:

- It is not possible to copy or move an object as long as the object is locked on the server.
- It is not possible to copy or move a system file node.
- The target node of a copy or move operation will only accept the objects of the selected source node when *all* objects can be copied or moved to the target node.
- When you copy an object (for a example, a program), it is not possible to paste it in the same library. Likewise, when you move an object, it is not possible to move it within the same library.

Libraries are copied and moved in the same way as any other node in the **Natural Server** view. The target node for a copied or moved library can be any system file node. It can even be any

other library node; in this case, all objects of the source library are copied. The following applies for libraries:

- When a library is copied or moved to a system file node for user libraries (`FUSER`), the new library must conform to the library naming conventions. Thus, when you copy or move a system library which starts with "SYS" to the user libraries, a dialog box appears and you have to specify another name. The default library name "USR-LIB" that is offered in this dialog box can be overwritten.
- The same applies when copying or moving a user library to the system libraries (`FNAT`) where the library names must start with "SYS". In this case, the dialog box mentioned above offers the default name "SYS-LIB".
- In all other cases, the name of the source library is taken as the name of the target library.

You can also copy and move nodes from one Natural server environment to another Natural server environment, for example, from a mainframe environment to a UNIX environment.

Note: Using drag-and-drop, it is possible to download libraries and objects to an existing project in the Eclipse workspace. For further information, see *Downloading an Existing Library or Object from a Natural Server*.

To copy objects or libraries using menu commands

1. In the **Natural Server** view, select one or more nodes.
2. Invoke the context menu and choose **Copy**.
3. Select the target node in the **Natural Server** view.
4. Invoke the context menu and choose **Paste**.

To copy objects or libraries using drag-and-drop

1. In the **Natural Server** view, select one or more nodes.
2. Click and hold down the left mouse button.
3. Drag the mouse to the node in the **Natural Server** view to which you want to copy the objects.
4. Hold down CTRL.
5. Release the mouse button and then CTRL.

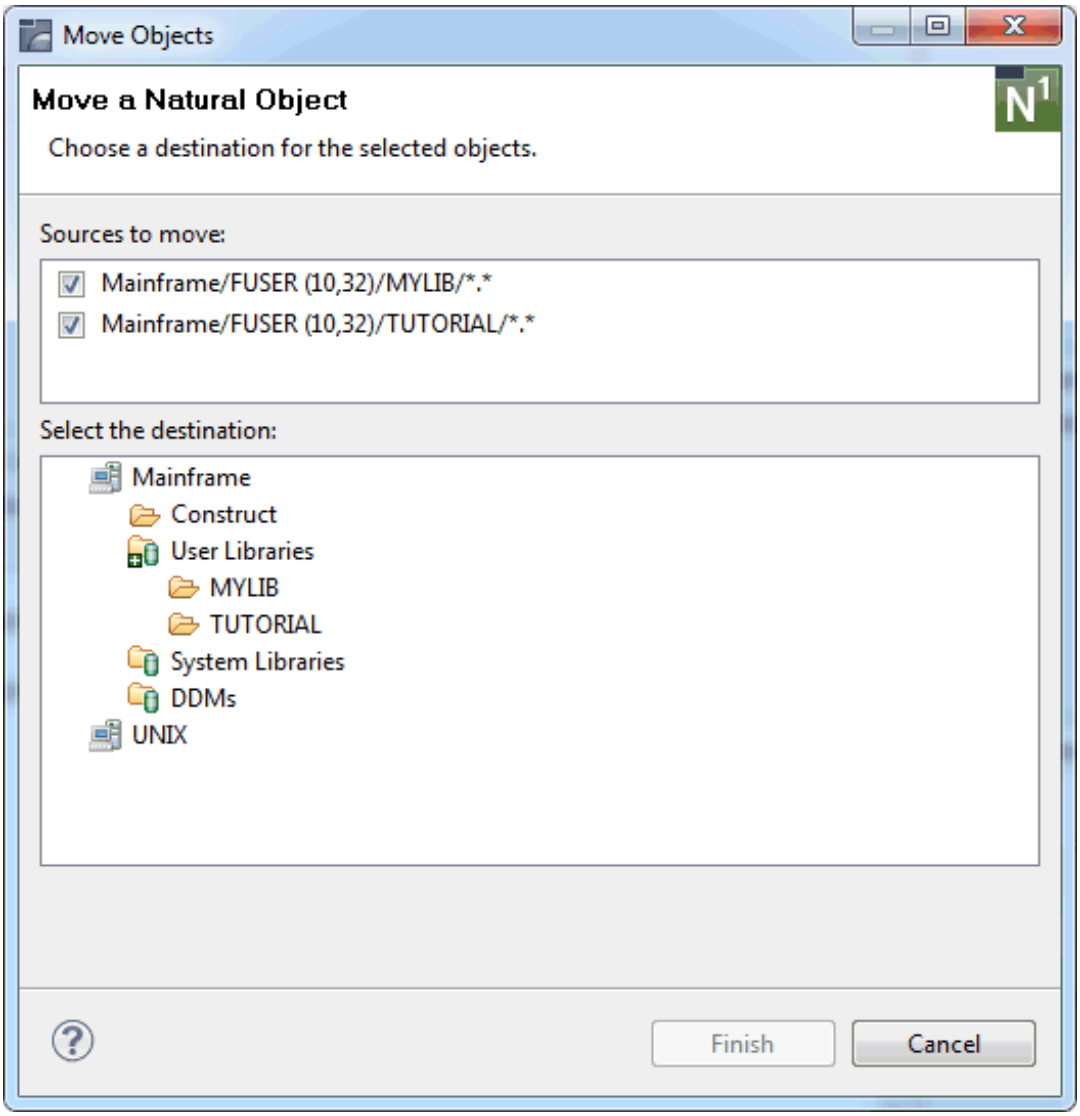
   Caution: If you do not hold down CTRL before releasing the mouse button, the objects are moved.

To move objects or libraries using a menu command

1. In the **Natural Server** view, select one or more nodes.

2 Invoke the context menu and choose **Move**.

The following dialog box appears.



The upper part of the dialog lists the path(s) to the selected node(s). If you decide that you do no longer want to move one of the listed nodes, simply deselect the corresponding check box.

3 In the lower part of the dialog, select the target node for the moved object(s).

Note: Any filters that have been set are also used in the **Move Objects** dialog box.

4 Choose the **Finish** button.

To move objects using drag-and-drop

1 In the **Natural Server** view, select one or more nodes.

2 Click and hold down the left mouse button.

3 Drag the mouse to the node in the **Natural Server** view to which you want to move the objects.

4 Release the mouse button.

When you drag an object to a different environment (for example, from a UNIX environment to a mainframe environment), the object is copied and not moved. The object is not deleted at its original position.

Notes:

1. When you copy or move an object to another library in which an object of the same type and with the same name exists already, and when the corresponding option has been set in the Natural preferences (see ***Runtime Execution*** in *Setting the Preferences*), you are asked whether you want to overwrite the object.
2. When you are dragging nodes, the mouse pointer always indicates whether the objects can be dropped or not.
3. If the automatic refresh does not take place, refresh the source and/or target node manually to see the result of a copy-and-paste, move or drag-and-drop operation. See ***Refreshing the Display***.

Renaming Objects and Libraries

When renaming an object or library, make sure to adhere to the Natural naming conventions.

Note: It is not possible to rename error messages, mainframe DDMs, and the library `SYSTEM`.

To rename an object

1 In the **Natural Server** view, select the node that is to be renamed.

2 Invoke the context menu and choose **Rename**.

A dialog box appears.

3 Enter a new name and choose the **OK** button.

Deleting Objects and Libraries

When you delete an object in the **Natural Server** view, both source and generated program are deleted. It is not possible to delete an object when it is currently locked on the server.

A library is deleted in the same way as a Natural object. If you are working in a multiple-user environment, you should only delete a Natural library if you have exclusive access to the library involved. It is not possible to delete the library `SYSTEM`.

To delete objects or libraries

1 In the **Natural Server** view, select one or more nodes.

2 Invoke the context menu and choose **Delete**.

   Or:

   Press DEL.

   When the corresponding option has been set in the Natural preferences (see ***Runtime Execution*** in *Setting the Preferences*), you are prompted to confirm the deletion.

Unlocking Locked Objects

When you are working in Natural server mode, it may happen that the connection to the Natural server is lost. In this case, any objects that you are editing directly on the Natural server remain locked until they are manually unlocked.

You can view the Natural objects that are locked. If required, you can unlock them.

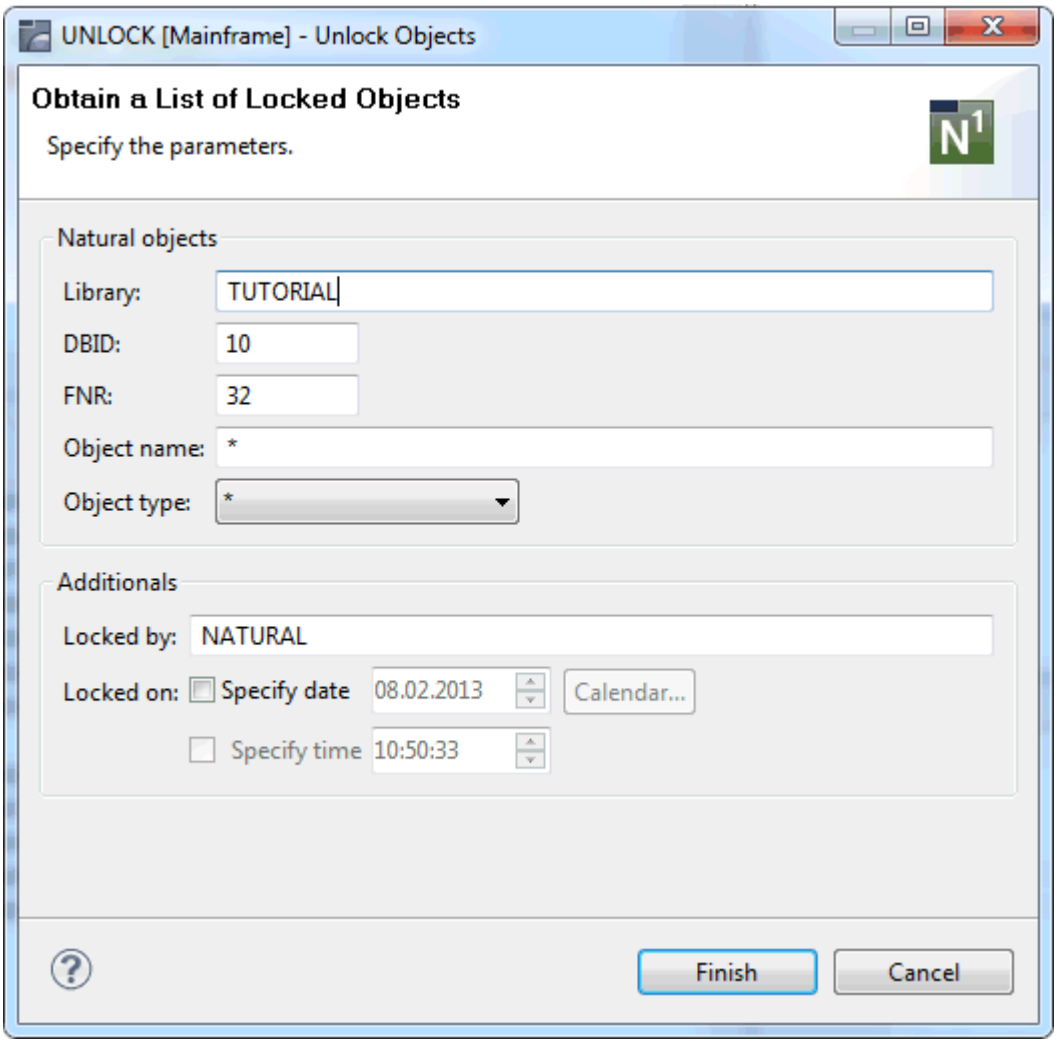
To unlock locked objects

1 In the **Natural Server** view, select a node in the appropriate Natural server environment.

   For example, select the node for a library containing locked objects. The library name and all available information will then be provided in the dialog box which is used for unlocking.

2 Invoke the context menu and choose **Unlock**.

   A dialog box appears. When a library was selected, its name and the appropriate database ID and file number are automatically provided.



Notes:

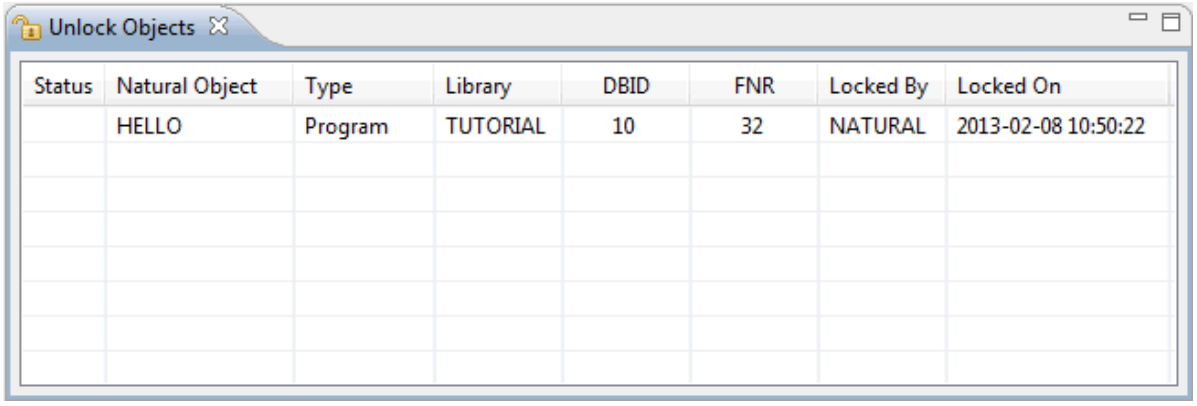
1. Depending on the selected node, all available information is filled into the dialog box as default information.
2. If you have selected a single object, the object is unlocked immediately and a corresponding message appears in the status line. The above dialog box does not appear in this case.
3. If several nodes (for example, several libraries) are selected, only the first node is considered by the **Unlock** command.
4. If Natural Security is active on the Natural server, you can only unlock objects if you are allowed to do so.

3 You can limit your search by specifying further options (for example, an object type or a specific date and time). The following options are available:

Option	Description
Library	The name of the library which contains the locked object. Asterisk notation (*) can be used.
DBID	The database ID of the specified library. Asterisk notation (*) can be used. On mainframe servers with parameter `SLOCK=PRE`, the following applies: When asterisk notation (*) is used, only the current `FNAT`, `FUSER` and `FDIC` files are scanned.
FNR	The file number of the specified library. Asterisk notation (*) can be used. On mainframe servers with parameter `SLOCK=PRE`, the following applies: When asterisk notation (*) is used, only the current `FNAT`, `FUSER` and `FDIC` files are scanned.
Password	This option is available only for a mainframe server and when the profile parameter `SLOCK=PRE` has been set in the mainframe environment. The password for the specified system file (DBID and FNR). Needs not be specified when the DBID and FNR of the current `FNAT` or `FUSER` are used.
Cipher	This option is available only for a mainframe server and when the profile parameter `SLOCK=PRE` has been set in the mainframe environment. The cipher key for the specified system file (DBID and FNR). Needs not be specified when the DBID and FNR of the current `FNAT` or `FUSER` are used.
Object name	The name of the object to be unlocked. Asterisk notation (*) can be used. For a list of all objects from a specific start value, the notation ">" can also be used.
Object type	If you want to restrict the objects to be unlocked to a specific object type, select the corresponding object type from the drop-down list box. The asterisk (*) in the drop-down list box means that all locked objects will be found.
Locked by	The ID of the user who caused the object to be locked. Asterisk notation (*) can be used. Your own user ID is provided by default. If Natural Security is used, you can only specify another user ID if the security unlock flag is set to "F" (forced unlock) in the Natural Security user profile.
Locked on	When the **Specify date** check box is selected, you can specify a date (either in the spin box or in the dialog box which appears when you choose the **Calendar** button). All locked objects between this date and the current date will then be found. When you select the **Specify time** check box, you can also define a time. All locked objects between this time on the specified date and the current date will then be found.

4 Choose the **Finish** button.

When locked objects exist, they are shown in the **Unlock Objects** view.



Note: This view is automatically shown when locked objects are found. It is not shown by default when you open the NaturalONE perspective. See also *Showing a View of the NaturalONE Perspective*.

5 In the **Unlock Objects** view, select the object(s) that you want to unlock.

6 Invoke the context menu and choose **Unlock Objects**.

All selected objects are unlocked. For each unlocked object, the following icon is shown in the **Status** column: .

Working with DDMs

How DDMs are shown in the **Natural Server** view depends on the type of server environment (see also *Contents of the Natural Server View*).

In previous versions of NaturalONE, the way in which protected libraries were shown depended on the setting of the **Display DDMs in library** option in the Natural preferences. The **Display DDMs in library** is no longer available. Instead, the following behavior applies:

- If the server has Natural Security installed, all DDMs which are accessible to the current user in a specific library are shown in a **DDMs** group node of this library, regardless of whether the DDMs are public or protected. The **DDMs** system file node is no longer available in this case.
- If the server has no Natural Security installed:
  - For Mainframe servers and other servers with `FDDM`, the DDMs will be displayed below the **DDMs** system file node.
  - For UNIX, OpenVMS or Windows environments where the DDMs are stored in libraries (that is, the `FDDM` parameter has not been set), the DDMs will be displayed below the **DDMs** group node in the libraries.

The commands which can be executed on DDMs which are displayed in a library are restricted to **Catalog**, **Stow**, **Edit**, **List** and **Unlock**. File operation commands such as **Delete** or **Rename** cannot be used.

Working with Dialogs

You can work with dialogs when NaturalONE has been installed in a Windows environment.

Dialogs cannot be edited with a NaturalONE editor. However, when Natural Studio is installed on the same machine as NaturalONE, you can invoke Natural Studio from the **Natural Server** view and then edit the dialog directly in Natural Studio, using the dialog editor.

The prerequisites are:

- You have a license for Natural for Windows. This must be a development environment, not a runtime environment.
- Natural for Windows, which includes Natural Studio, is installed on the same machine as NaturalONE.

  Important: This feature only works as of Natural Version 6.3.9 for Windows. Therefore, Natural Version 6.3.9 or a later version must be installed.

- Natural Development Server (NDV) has been installed together with Natural for Windows.
- You have **mapped** the development server so that it is available in the **Natural Server** view. With a default installation, this server can be mapped with the name "localhost" and the port number "2700".

When these prerequisites are fulfilled and Natural Studio can be found in the development environment, the **Edit with Natural Studio** command is visible in the context menu for a selected dialog in the **Natural Server** view.

Natural Studio's dialog editor can only be invoked for dialogs that are accessible via the **Natural Server** view. Dialogs which are stored in the Eclipse workspace (in the **Navigator** view or in the **Natural Navigator** view) can only be edited with NaturalONE's source editor (however, this is not recommended). If you want to edit a dialog that is stored in the Eclipse workspace with Natural Studio's dialog editor, you have to upload the dialog to an appropriate development server.

To edit a dialog

1. In the **Natural Server** view, select the dialog that you want to edit.
2. Invoke the context menu and choose **Edit with Natural Studio**.

Natural Studio's dialog editor is invoked for the selected object and you can now edit it. For detailed information on the dialog editor, see the Natural for Windows documentation.

Note: If you want, you can also use any other features of Natural Studio. All installed functionality will be available to you.

Working with Resources

In Natural terminology, resources are non-Natural files (such as images or HTML files) that are used in a Natural application.

In the **Natural Server** view, the resources in a library are contained in a node called **Resources**. When you expand this node, additional nodes are shown for the available types of resources. For example, all bitmaps (**.bmp*) are shown in a node called **Bitmap Images**, and all icons (**.ico*) are shown in a node called **Icons**.

If you want to modify a resource, you have to double-click it. This downloads the resource into a temporary project in your Eclipse workspace and opens the appropriate internal editor. You can then apply changes to the resource and save them; the resource is then changed on the server. When you close the internal editor, the resource is automatically removed from the temporary project. When the temporary project no longer contains any resources or Natural objects, it is automatically removed from the Eclipse workspace.

Caution: There are some resource types (for example, icons) for which an internal editor is not available. In this case, an external program is started which is registered as the system default editor for that file type. When you close an external program, Eclipse (and thus NaturalONE) is not aware of this. As a result, the resource is not removed from the temporary project. In this, case you have to delete the resource manually from the temporary project and, if the resource was the last object in the temporary project, you also have to delete the temporary project manually.

Working with Private-mode Libraries

When you switch on private mode in the properties of a Natural project which is located in the Eclipse workspace, one or more private-mode libraries are automatically created on the associated Natural server when you build the project. These private-mode libraries are visible in the **Natural Server** view.

It is not possible to change a private-mode library in the **Natural Server** view, and it is not possible to change any object in a private-mode library (for example, it is not possible to edit such an object). The corresponding commands in the context menu are unavailable (gray).

For further information, see *Private-mode Libraries* in *NaturalONE in a Nutshell*, and the description for **private mode** in *Changing the Project Properties*.

13 Launching Natural Applications

In order to launch a Natural Application (for **executing**, **debugging** or **profiling**), the Eclipse Launch configuration framework is used with the standard Eclipse commands **Run As**, **Debug As** and **Profile As**.

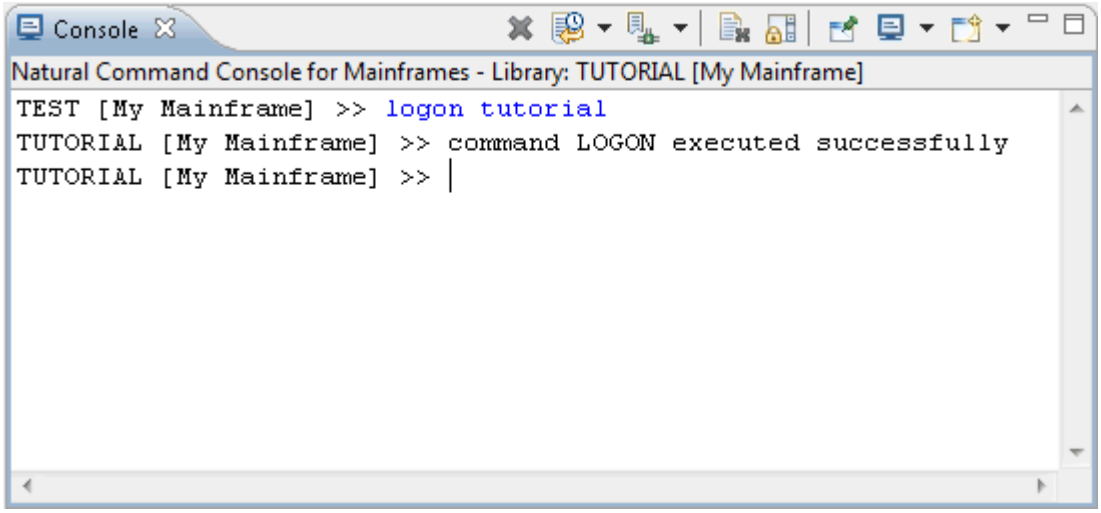
For further details, please refer to ***Launching Natural Applications*** in *Working with Natural Projects in Local Mode*.

14 Using the Natural Command Console for Mainframes

- General Information 178
- Opening the Natural Command Console for Mainframes 179
- Active Environment 179
- Server Connections 180
- Commands in the Natural Command Console for Mainframes 180

General Information

Using the Natural command console for mainframes, you can enter Natural system commands directly in the **Console** view.



When a system command provides output (for example, when you execute the `LIST` or `TECH` command), the output (Natural I/O) is either shown in the internal browser or in an external browser, depending on the settings in the Natural **preferences**.

Application-dependent information (for example, "command LOGON executed successfully") and error messages (for example, "NAT0082 Invalid command, or Object XY does not exist in library.") are shown directly in the Natural command console, behind the input prompt.

Notes:

1. For detailed information on the Natural system commands, see the Natural for Mainframes documentation. The most up-to-date Natural documentation is always available at ***http://documentation.softwareag.com/*** (Empower login required).
2. The Natural command console can only be used with Natural servers on mainframes. It is not possible to use it with Natural servers on UNIX, OpenVMS or Windows.

Opening the Natural Command Console for Mainframes

The Natural command console for mainframes is available in the **Console** view, in the list of registered consoles.

To open the Natural command console for mainframes

1 Go to the **Console** view.

2 Click the "Open Console" icon ( ) which is shown in the local toolbar of the **Console** view.

3 From the resulting menu, choose **Natural Command Console for Mainframes**.

Active Environment

When you select a node in the **Natural Server** view, the name of the corresponding server environment and library is shown in the header line of the Natural command console (for example, "Library: TUTORIAL [My Mainframe]". When a library is currently not selected in the **Natural Server** view, the Natural command console uses the default logon library.

As soon as you switch from the **Natural Server** view to the Natural command console, the environment information which is part of the command console's input prompt is adapted so that it shows the environment that was last selected in the **Natural Server** view. The environment that is shown in the input prompt is always used as the active environment for any subsequent Natural system commands that you enter.

In the Natural command console, you can change the environment with the Natural system commands `LOGON`, `MAP` and `UNMAP`. The change is reflected in the header line and input prompt of the Natural command console. The new environment is considered to be the active environment, and any system command that you will now enter will be executed in the active environment. The system command `MAP` adds a Natural environment to the **Natural Server** view. The system command `UNMAP` removes a mapped server environment from the **Natural Server** view. For detailed information on the system commands `MAP` and `UNMAP` (and only for these two system commands which are normally not available in a mainframe environment), see the Natural for Windows documentation.

The following MAP command syntax applies if you want to establish a connection to a **Natural Development Server**, using the Natural command line:

```
MAP ENVIRONMENT=environment-name server-name port-name [SSL[userid [password
['parm=value;...']]]]
```

Example:

```
MAP IBM2 4722 SSL
```

The current selection in the **Natural Server** view is not affected when you change the environment in the Natural command console. For example, when the library `TUTORIAL` is currently selected in the **Natural Server** view and you log on to the library `TEST` using the Natural command console, the library `TUTORIAL` is still selected in the **Natural Server** view. However, when you select a different library in the **Natural Server** view, this change is immediately reflected in the Natural command console and this library will become the active environment.

Server Connections

The Natural command console for mainframes establishes permanent connections to the Natural servers that are accessed from the Natural command console. A connection to a server is established when the first Natural system command is issued in this server environment. Existing connections are activated and deactivated when the current server environment changes. All connections are closed when the NaturalONE session ends.

Commands in the Natural Command Console for Mainframes

In addition to the functionality common to all Eclipse consoles (such as "Pin Console"), the Natural command console for mainframes provides the following commands:

Icon	Command	Description
	Close Console	Closes the Natural command console for mainframes. When you close the Natural command console, the command history (see below) is cleared.
	Natural Command History	Lists all commands that you have entered since the Natural command console has been opened. When you select a command, it is copied to the input prompt. You can then edit the command (if required) and execute it once more. Using the UP-ARROW and DOWN-ARROW keys, you can scroll through the command history.
	Set Selected Environment	Lists all server connections that have been established in the current Natural session. The currently active environment is marked. When you select a server environment from this list, this environment becomes the active environment and subsequent commands will be executed in this environment.

Note: As long as a Natural system command is being processed, the console is set to read-only, a wait cursor is shown, and the above icons are disabled.

IV Using the Natural Editors

This part describes the Natural editors that are available with NaturalONE. It covers the following topics:

General Information

Using the Source Editor

Using the Map Editor

Using the DDM Editor

Note: The above editors are used to develop the basic functionality of a Natural application. The map editor is used to create character-based user interfaces. If you want to create complex graphical user interfaces, you will use Ajax Developer instead. Ajax Developer includes a number of tools. Its central tool is the Layout Painter which is used to define layouts for HTML pages. For further information, see the *Ajax Developer* documentation.

15 General Information

- Types of Natural Editors 186
- Invoking a Natural Editor 187
- Editing Data Areas 187
- Viewing Dialogs, Classes and Adapters 188
- Problems in Your Natural Sources 188
- Parsing Dependent Objects 190
- Unicode and Code Page Support 191
- Bidirectional Language Support 192
- Source Header 196
- Line Numbers 196

Types of Natural Editors

The following table indicates which type of editor is used for a specific object type:

Icon	Object Type	Extension	Editor
	Program	NSP	Source editor
	Class	NS4	
	Subprogram	NSN	
	Subroutine	NSS	
	Function	NS7	
	Copycode	NSC	
	Helproutine	NSH	
	Text	NST	
	Dialog	NS3	
	Adapter	NS8	
	Global data area (GDA)	NSG	
	Local data area (LDA)	NSL	
	Parameter data area (PDA)	NSA	
	Map	NSM	Map editor
	Data definition module (DDM)	NSD	DDM editor

The above icons are used for the sources in the **Navigator** view, in the **Natural Navigator** view and in the **Dependencies** view.

They are also used in the **Natural Server** view. On a Natural server, however, Natural objects can consist of the source, the generated object (for example, a generated program) or both. The difference is reflected in the icon. Therefore, in the **Natural Server** view, the above icons can be shown as follows (example for a program):

Icon	Description
	Without a green ball and not gray: only the source of the object is available.
	With a green ball and not gray: source and generated object are available.
	With a green ball and gray: only a generated object is available and no source.

Invoking a Natural Editor

A Natural editor is invoked when you open an existing Natural object or when you create a new Natural object.

You can open an existing object either from a Natural project in the Eclipse workspace (**Navigator** view or **Natural Navigator** view) or directly on the Natural server (**Natural Server** view). For further information, see the following sections:

- ***Editing Objects*** in *Working with Natural Projects in Local Mode*.
- ***Editing Objects*** in *Working with Natural Objects in Natural Server Mode*.

You can create a new object in a Natural project in the Eclipse workspace (**Navigator** view or **Natural Navigator** view). For further information, see the following section:

- ***Creating Natural Objects*** in *Working with Natural Projects in Local Mode*.

Editing Data Areas

NaturalONE makes use of the Natural source editor for data area editing. The wizard for creating new objects generates a skeleton for the `DEFINE DATA` statement. Within this skeleton, you are able to write the variable definition. Editing of the data area is well supported through the Natural parser.

Data areas that are downloaded from a Natural server are automatically normalized. This means that the internal data area format is converted into a `DEFINE DATA` statement. The benefit is that the data area can be defined quickly and that versioning of the data area source is possible. When you use the **Build Natural Project** or **Upload** command, the normalization is reversed: the internal data format of the Natural server is then generated from the `DEFINE DATA` statement.

Viewing Dialogs, Classes and Adapters

NaturalONE does not support dialogs and classes (see also *Are all Natural object types supported?* in *Frequently Asked Questions*). However, it is possible to view (and even edit, although this is not recommended) the source code of such an object using the source editor. With NaturalONE, it is not possible to create these objects from scratch. The same is true for adapters which are generated by Natural for Ajax.

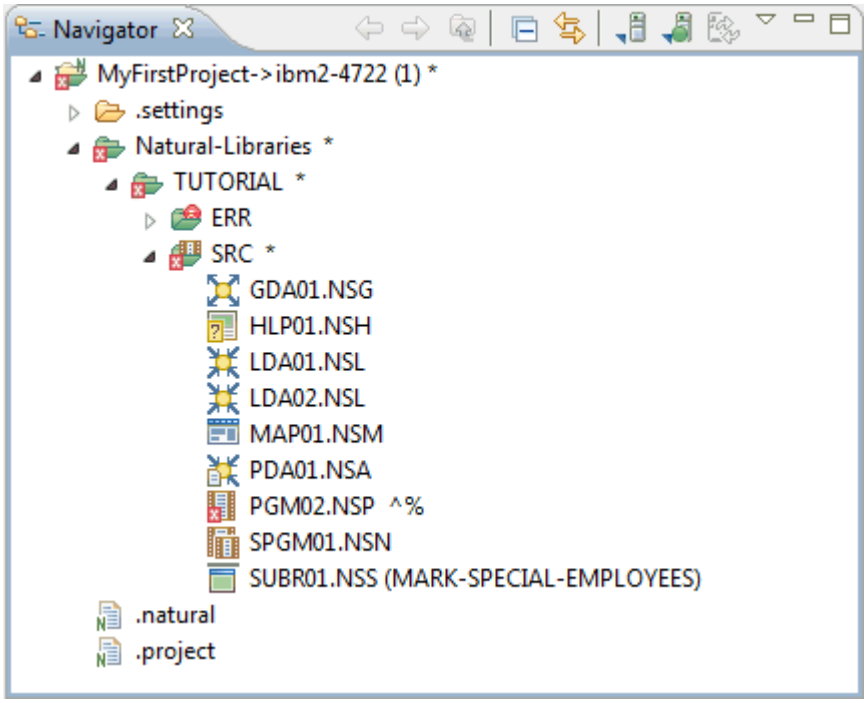
Note: When certain prerequisites are fulfilled, Natural Studio's dialog editor can be accessed directly from NaturalONE. See ***Working with Dialogs***.

Problems in Your Natural Sources

When a syntax error is detected in a project after an object has been saved, or when a compiler problem is detected in a project during a **server update** (that is, when a source could not be cataloged on the server), this is indicated in the **Navigator** view or in the **Natural Navigator** view.

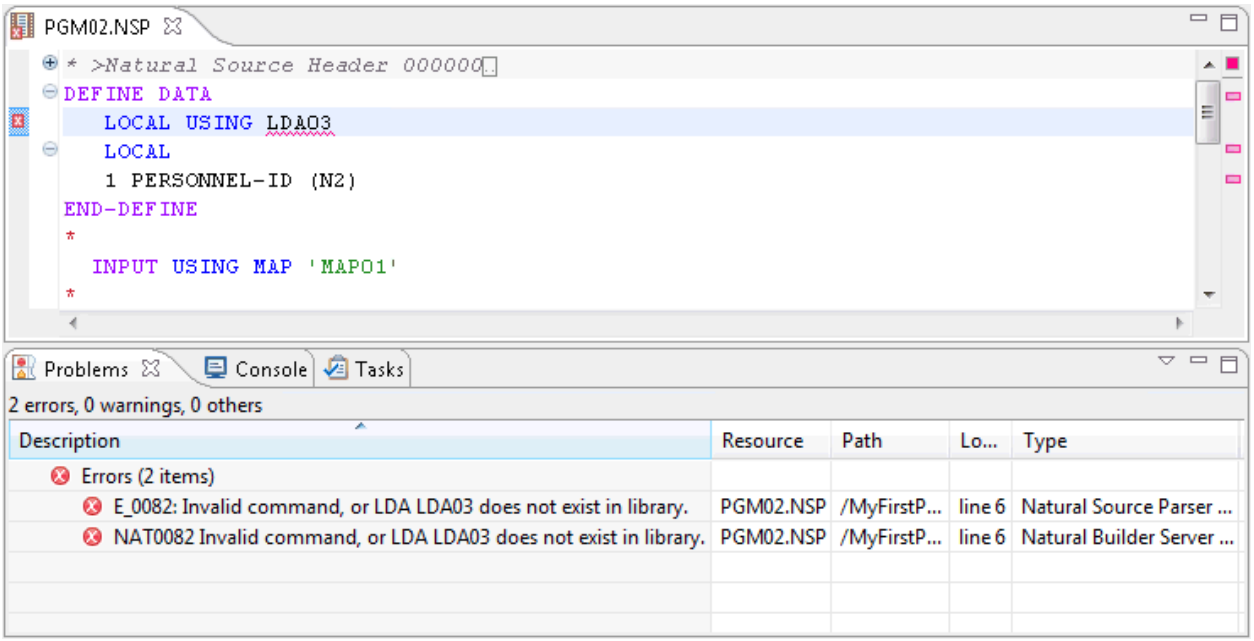
Note: Syntax parsing of dependent objects can be enabled and disabled. For further information, see ***Parsing Dependent Objects***.

In the case of a problem, the icon of a project contains a corresponding decoration. The subnodes containing the problem also contain the decoration in their icons.



Note: The label decorations for the server problems are controlled by the preferences under **General > Appearance > Label Decorations**. If you do not want to have these label decorations, just go to the above mentioned preference page and deselect **Natural Compiler Problems**.

The **Problems** view shows the corresponding error message(s). When you double-click an error in the **Problems** view (or when you select the error and then choose **Go to** from the context menu), the corresponding line in the editor is selected.



Two types of errors are shown in the **Problems** view. Errors starting with "NAT" are Natural system error messages which are issued by Natural on the server. Errors starting with "E_" are issued by the local NaturalONE parser when an error is found in the editor.

When you select an error in the **Problems** view, the error message is also shown at the bottom left of the Eclipse window.

In the Natural preferences, you can specify whether information is to be shown in the **Problems** view. See *Source Editor* in *Setting the Preferences*.

Parsing Dependent Objects

When syntax parsing of dependant objects is enabled in the Natural preferences (see ***Natural > Builder*** in *Setting the Preferences*), any dependent objects containing syntax errors are shown with an error marker in the **Navigator** view or in the **Natural Navigator** view. In addition, a corresponding error message is shown in the **Problems** view.

A dependent object is, for example, a program which uses the fields from a data area. If the data area is changed, this change must also be considered in the dependent program.

In the Natural preferences, you can also specify that syntax parsing is to be stopped after the first error (see ***Natural > Builder*** in *Setting the Preferences*).

Depending on your settings, syntax parsing is either done automatically or can be started manually. This is explained in the topics below:

- Automatic Parsing
- Manual Parsing

Automatic Parsing

When automatic parsing is enabled, all dependent objects are automatically parsed after a Natural source has been saved.

Automatic parsing can be enabled and disabled using the **Parse Syntax Automatically** command or by setting the corresponding option in the Natural preferences (see ***Natural > Builder*** in *Setting the Preferences*). When you choose the **Parse Syntax Automatically** command, the setting in the preferences is adapted accordingly, and vice versa.

When automatic parsing is enabled, a checkmark is shown next to the **Parse Syntax Automatically** command.

To enable or disable automatic parsing of dependent objects

- From the **Project** menu, choose **Parse Syntax Automatically**.

Manual Parsing

When automatic parsing is disabled, the **Parse All** command is enabled and can be used to invoke the parser manually. All dependent objects which need parsing since you enabled syntax parsing are then considered by the parser.

When you correct and save your sources, the error markers in the **Navigator** view or in the **Natural Navigator** view and the entries in the **Problems** view are not automatically removed. To remove them, you have to invoke the **Parse All** command once more (or you have to switch on automatic parsing).

To parse dependent objects manually

- From the **Project** menu, choose **Parse All**.

  Or:

  Press CTRL+ALT+P.

Unicode and Code Page Support

When a source is downloaded from a Natural server into the Eclipse workspace, the source is always converted from the server encoding to the Eclipse text file encoding UTF-8.

The server encoding of the source is stored in the source header (for further information on the source header, see **below**).

Caution: When you edit a source in the Eclipse workspace and this source has UTF-8 encoding, you can enter any Unicode characters, however, only characters which are contained in the server encoding can be uploaded to the server.

In the **Navigator** view or in the **Natural Navigator** view, you can change the server encoding of the source in the **Properties** dialog box of the source. The supported ICU encodings depend on the Natural server. See *Changing the Object Properties*.

When a source is transferred back to a Natural server, it is converted to the server encoding that is defined in the **Properties** dialog box of the source. When the source contains "unsupported" characters, it cannot be saved on the Natural server and the server returns an error.

Note: The `SRETAIN` profile parameter (set on the Natural server) is not evaluated when uploading a source to a Natural server. You have to define a suitable project encoding in the Natural project properties or a suitable server encoding in the object properties of the individual Natural objects. For example, if you leave the text box for the project encoding blank in the project properties, a code page will not be assigned to all newly created Natural objects.

This is also true for a temporary project which is created when you edit a source in the **Natural Server** view.

In the **Natural Server** view (on UNIX, Windows and OpenVMS servers only), you can also change the encoding of a Natural object in the **Properties** dialog box (on the **Object Details** page). Changing the encoding of an object means that the encoding information which is stored on the server is changed; it does not mean that the content of the object is converted in any way.

For more information, see *Unicode and Code Page Support* in the Natural documentation for the appropriate platform.

Error Messages

Error Messages which are created in/or downloaded to Natural projects in the Eclipse workspace are stored in UTF-8 file encoding. However, the error messages do not contain the original server encoding.

When downloaded from a Natural server, the error message will be converted from the server encoding to UTF-8 and when an error message is uploaded to a Natural server it will be converted from UTF-8 to the server encoding. All characters which are not contained in the server encoding will be lost.

So the runtime environment of your Natural project should be started with the correct encoding.

Bidirectional Language Support

Some languages, for example Arabic and Hebrew, are written from right-to-left (RTL), whereas the majority of the languages, for example English and German, are written from left-to-right (LTR). Text which contains both left-to-right and right-to-left characters is called bidirectional text.

In addition to the bidirectional language support of the Eclipse workbench (see the Eclipse online help for further information), NaturalONE offers special bidirectional support which is described in the topics below:

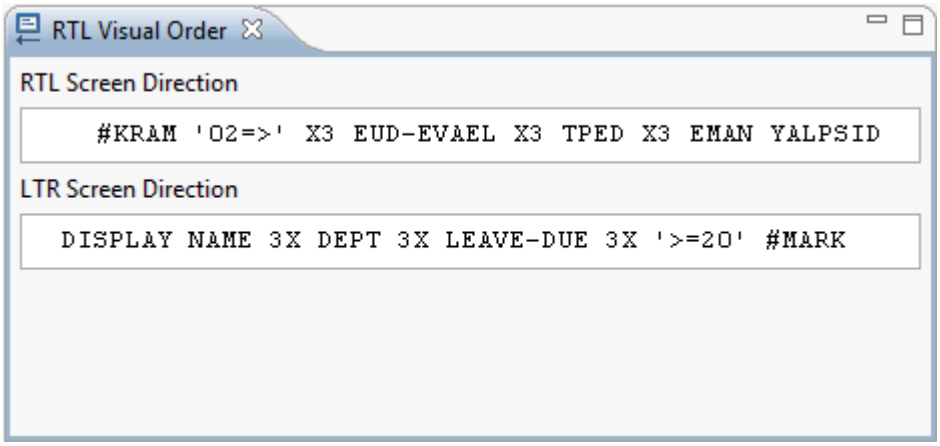
- RTL Visual Order View
- Visual Order Option
- Map Editor and PM=I

- Arabic Shaping

RTL Visual Order View

The **RTL Visual Order** view is used to support applications that have been originally written for terminals which support inverse (right-to-left) print mode, but no bidirectional data. These applications create the display order of bidirectional data in the application code, that is, the application data is stored in visual order.

Note: This view is not shown by default when you open the NaturalONE perspective. For information on how to display it, see *Showing a View of the NaturalONE Perspective*.



The **RTL Visual Order** view can be used in the following cases:

- With the **source editor**.
- With the **Source** page of the **map editor**, which shows the code for a **processing rule**.

The selected line in the active editor (either in the source editor or on the **Source** page of the map editor) is converted from visual to logical format and is displayed in right-to-left screen direction (first text box in the view) and left-to-right screen direction (second text box in the view).

If you want to change the currently selected line in the active editor, you can do this in the **RTL Visual Order** view. Every change is immediately reflected in the editor.

Visual Order Option

When the **Visual order** option is selected in the Natural preferences (see ***Regional Settings > RTL Languages*** in *Setting the Preferences*), the application data (that is, Natural sources and data from databases) is assumed to be in visual order. In this case, several editors and tools convert the data from visual order to logical order before displaying the data. When the data is saved, it is converted back to visual order.

Map Editor and PM=I

The settings for bidirectional language support are defined in the Natural preferences. See ***Map Editor*** in *Setting the Preferences*.

Inverse print mode is supported for maps and data fields. The corresponding parameter (`PM=I`) can be set in the **Properties** view. See ***Changing the Properties for the Map*** in the description of the map editor.

Arabic Shaping

Important: With the current version of NaturalONE, support for Arabic shaping is provided as a preview version. There will be no support for this preview version. Any feedback from our customers on this new feature is greatly appreciated. Since support for Arabic shaping is still under development, a later version of NaturalONE may include substantial changes to this type of support.

In Arabic text, all characters of a string are normally connected with each other. For this reason, Arabic characters have up to 4 presentation forms: the isolated, the final, the initial and the medial form. The form that will be used depends on the position of the character in the string. For example, the Arabic character "MEEM" has the following forms in Unicode:

U+0645	م	ARABIC LETTER MEEM
U+FEE1	م	ARABIC LETTER MEEM ISOLATED FORM
U+FEE2	م	ARABIC LETTER MEEM FINAL FORM
U+FEE3	ـمـ	ARABIC LETTER MEEM INITIAL FORM
U+FEE4	ـمـ	ARABIC LETTER MEEM MEDIAL FORM

Moreover, some characters are combined to a new form if they appear consecutively in a string. This is called a “ligature”. For example, the characters

U+0644	ل	ARABIC LETTER LAM
U+0627	ا	ARABIC LETTER ALEF

have the following combined form:

U+FEFB	ﻻ	ARABIC LIGATURE LAM WITH ALEF ISOLATED FORM

Unicode strings should include only the Arabic characters in the Arabic block (U+0600 through U+06FF) or the Arabic Supplement block (U+0750 through U+077F); it is not recommended to use the presentation forms in regular Arabic text. It is up to the user interface to display the correct shapes of the characters. However, applications which have been originally written on simple terminals often use Arabic presentation forms in the strings.

If data containing characters from the Arabic presentation forms has to be handled in the Eclipse context, it is necessary to unshape the Arabic characters for getting full bidirectional support from Eclipse. "Unshape" means that every Arabic presentation form character is converted to its base character. For example, U+FEE2 (ARABIC LETTER MEEM FINAL FORM) is converted to U+0645 (ARABIC LETTER MEEM).

When the data is transferred back to the original server, it is necessary to shape the Arabic strings again for getting full bidirectional support on the server platform. "Shape" means that every Arabic base character is converted to the appropriate Arabic presentation form. For example, if U+0645 (ARABIC LETTER MEEM) is used as the last character of a string, it is converted to U+FEE2 (ARABIC LETTER MEEM FINAL FORM).

NaturalONE supports Arabic shaping during source transfer. Using the Natural preferences for Arabic shaping (see ***Regional Settings > RTL Languages*** in *Changing the Project Properties* and ***Regional Settings > RTL Languages*** in *Setting the Preferences*), Arabic shaping conversion can be controlled. The unshape operation is performed when sources are downloaded from a server and the shape operation is performed when sources are uploaded to a server.

Notes:

1. It is not guaranteed that a source will be absolutely the same as the original source after performing an unshape/shape operation. The shape operation will always use ligature forms if possible, even if the original source contains the single character forms. Moreover, if the original source contains an "improper" shape (for example, a final form where an initial form is expected), the shape operation will not restore the original character.
2. Sources which contain Arabic strings both in visual left-to-right (LTR) order and in logical order cannot be handled with NaturalONE. NaturalONE assumes that all strings in a source are stored in the same order.

Source Header

A source header is used to maintain some source properties (such as programming mode or code page).

The source header is part of a Natural source. It is removed when the source is transferred to the Natural server and is reinserted when the source is read from the server into the Eclipse workspace. Since the header belongs to the source, it remains in the source when it is stored in the repository of your version control system.

When using a Natural editor in Eclipse, the source header is protected. In this case, it cannot be modified or deleted. The background color for the protected lines which make up the source header can be changed in the Natural preferences. See ***Syntax Coloring*** in *Setting the Preferences*.

Caution: When using a non-Natural editor, do not modify or delete the source header. This may result in compile errors or properties which can no longer be assigned correctly.

Line Numbers

By default, Eclipse has no line number information associated with a source. Line numbers make it almost impossible to version sources in a version control system: each new or removed line would result in a renumbering of the subsequent lines and these lines would be marked as changed. Therefore, NaturalONE strips off the Natural line numbers from the source.

However, it is possible to switch on the display of the Eclipse line numbers (**Preferences > General > Editors > Text Editors > Show line numbers**). This may be especially useful in the case of an error because the parser indicates the error position by its line number. This may also be useful when executing a Natural application: in the case of an error, the Natural runtime uses the line number to inform the user about the error position.

Eclipse uses an increment of 1 for line numbering. When Natural objects are downloaded to the Eclipse workspace, all line number references in the source code are renumbered accordingly. When the objects are uploaded to the Natural server, the line number references are all changed back to the original increment.

The following patterns are recognized as being valid line number references (where *nnnn* is a four-digit number):

```
(nnnn)
(nnnn/
(nnnn,
```

The source editor supports the line number references.

Caution: If the Natural sources are modified with an editor other than the source editor, line number references may be destroyed.

By default, line number references in constants are not renumbered when you are editing a source. If they are to be renumbered while editing, you have to enable the **Renumber line numbers in constants** option in the project properties. See ***Editor*** in *Changing the Project Properties*. However, when you are downloading a source using the **Add to New Project** or **Add to Existing Project** command from a server where `RNCONST` is set to `ON`, the line number references in constants are always renumbered.

During the download, it is possible to replace the line number references with labels. This can be enabled in the Natural preferences. See ***Natural > Options*** in *Setting the Preferences*.

Important: When the above mentioned option is set in the Natural preferences, all valid line numbers (as described above) will be replaced with labels. It will not be checked whether the result is valid Natural syntax. Therefore, it is up to you to check the result of the replacement.

16 Using the Source Editor

- About the Source Editor 200
- Associated Views 200
- Using Content Assist 203
- Using Context-Sensitive Help 204
- Using Code Snippets from the Natural Community 205
- Working with Tasks 209
- Working with Bookmarks 212
- Error Handling in the Source Editor 213
- Going to a Specific Natural Line Number 213
- Opening Referenced Objects and Jumping to Variable and Internal Subroutine Definitions 214
- Externalizing Code Fragments 216
- Inserting Call or Include Statements 219
- Importing Data Fields 221
- Generating Counter Fields 224
- Adding and Removing Comments 226
- Toggle Comments 226
- Protecting Source Code Lines 227
- Protected Lines in Sources Generated by Construct or Code Generation 228
- Translating to Upper Case or Lower Case 228
- Indenting the Source Code Lines 229

About the Source Editor

The Natural source editor is based on the Eclipse text editor and supports, for example, syntax highlighting and content assist. In addition, it provides Natural-specific features.

The behavior of the source editor can be influenced by changing the Natural preferences. See ***Source Editor*** in *Setting the Preferences*.

An intelligent parser is used. During the input of the source code, you will receive a message in the case of a syntax error. In the project properties, you can determine for which platform the Natural syntax is to be parsed (see the description of the **parser options** in the section *Changing the Project Properties*).

For information on the Natural programming language (for example, on statements which can be specified in the Natural source editor), see the Natural documentation for the appropriate platform.

Note: For your convenience, the NaturalONE documentation (and help) includes the documentation for the Natural programming language on the different platforms. See *Natural Language for Mainframes*, *Natural Language for UNIX*, *Natural Language for OpenVMS* and *Natural Language for Windows*.

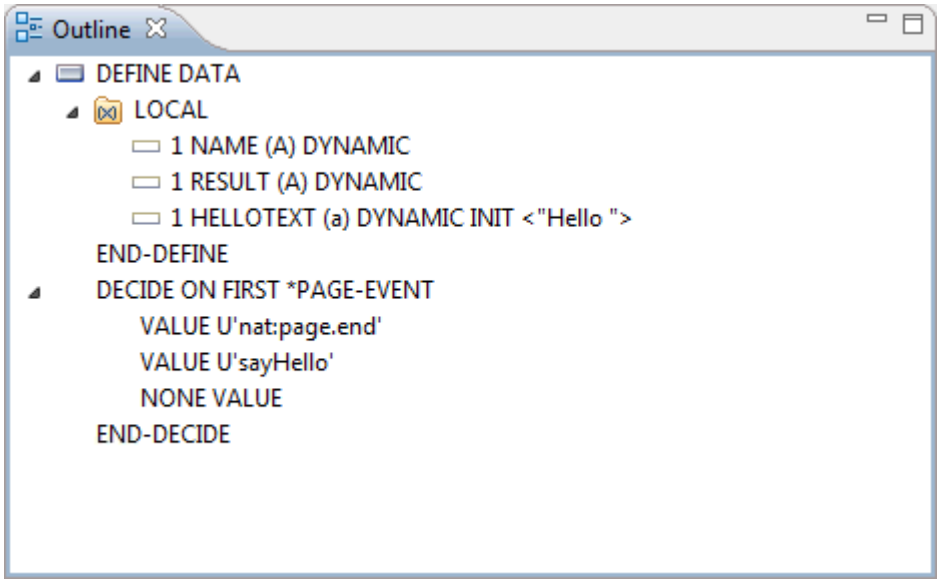
Associated Views

The source editor uses the following views of the NaturalONE perspective:

- Outline View
- Dependencies View

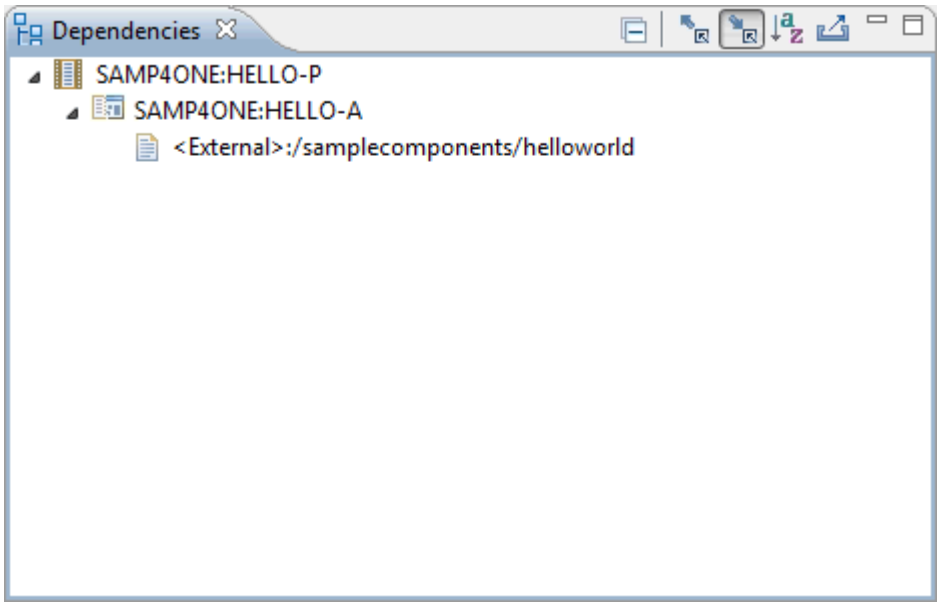
Outline View

This view shows all control structures that are used in the source which is currently shown in the active editor window. This includes all structures which can be displayed in the editor window in a collapsed way, such as `DEFINE DATA`, `IF` and `DECIDE`. When you select an object in the **Outline** view, the corresponding position is shown in the editor.



Dependencies View

This view shows the dependencies between the Natural object which is currently shown in the active editor window and other objects. It is refreshed each time the object is saved.



Other than what is shown in the **Navigator** view or in the **Natural Navigator** view, the **Dependencies** view always shows the Natural object names (not the alternative file names which may be up to 255 characters long). However, for subroutines, functions and DDMs, it always shows the so-called long names (that is, the name that is defined in the source itself, not the name under which the source was saved).

Displaying Active and Passive Cross-References

You can display the information in the **Dependencies** view in two different modes:

Icon	Mode	Description
	Active cross-references (callees)	Shows the objects that are referenced in the current source.
	Passive cross-references (callers)	Shows the objects in which the current source is referenced.

You can switch between these modes by choosing the corresponding icon in the local toolbar or by choosing the corresponding command (**Active XRefs** or **Passive XRefs**) from the context menu.

Opening Objects

When you select an object in the **Dependencies** view, you can choose the **Open** command from the context menu. Or you can double-click the object. This opens the object in the appropriate editor.

If you have enabled the option **Mark occurrences of the selected called object in the current source** in the Source Editor preferences, the cursor is positioned to the first occurrence of the corresponding called object and all references are marked.

When you have added a reference to an object in your code (for example, when you have added an `INCLUDE object-name` statement) and the referenced object is not yet available in your workspace, the object is indicated as follows in the **Dependencies** view (this example assumes that you have specified "TEST" as the object name):

```
<Unknown>:TEST
```

When you use the **Open** command (or a double-click) with an unknown object, a dialog box appears asking whether you want to download the object from the Natural server. If you agree, NaturalONE tries to establish a connection to the server (using the connection properties stored in the associated Natural project). When the connection can be established, the object is downloaded and then opened in the appropriate editor.

Note: Using the **Handling of missing objects** option in the Natural preferences, you can determine that specific types of unknown objects are automatically made available to the parser. See ***Natural > Project*** in *Setting the Preferences*.

Sorting Alphabetically

You can sort the objects in the **Dependencies** view alphabetically using the icon in the local toolbar. The sort order is library name first, then object name.

- **Copying Nodes as Text to the Clipboard**
  When you select a node in the **Dependencies** view, you can choose the **Copy** command from the context menu. Or you can press CTRL+C. This copies the selected node and all of its visible subnodes as text to the clipboard, depending on the current mode (active or passive cross-references). Invisible nodes (that is, nodes which are not expanded) are not copied. You can then paste the text from the clipboard into any other application (for example, a text editor), using the paste functionality that is provided by the target application (for example, CTRL+V).

  Other than what is shown in the **Dependencies** view, the copied text also includes the file extensions of the Natural objects. The file extensions are shown in brackets behind the Natural object names. In case of subroutines, functions, and DDMs, they are shown behind the long names of the objects.

- **Exporting Nodes as Text to a File**
  You can also export the contents of the **Dependencies** view as text to a file, depending on the current mode (active or passive cross-references). This file is stored in a project of your Eclipse workspace. In this case, it is not required that you select a node first. All visible nodes are written to the file. Invisible nodes (that is, nodes which are not expanded) are not written. To start the export, choose the icon in the local toolbar. In the resulting **Save As** dialog box, select the project in which you want to store the file (if you want, you can select any folder in this project), enter a file name and then choose the **OK** button. If the specified file name already exists at the selected location, a dialog box appears, asking whether you want to replace the existing file.

  Other than what is shown in the **Dependencies** view, the exported text also includes the file extensions of the Natural objects. The file extensions are shown in brackets behind the Natural object names. In case of subroutines, functions, and DDMs, they are shown behind the long names of the objects.

Using Content Assist

Content assist is a standard feature in Eclipse. With NaturalONE, basic content assist support is provided. It offers Natural syntax and variable support, based on the current context of the position inside the source code. This feature uses the templates which are defined in the Natural preferences. See ***Source Editor > Templates*** in *Setting the Preferences*.

Content assist can be switched off in the Natural preferences. See ***Source Editor*** in *Setting the Preferences*.

To invoke content assist

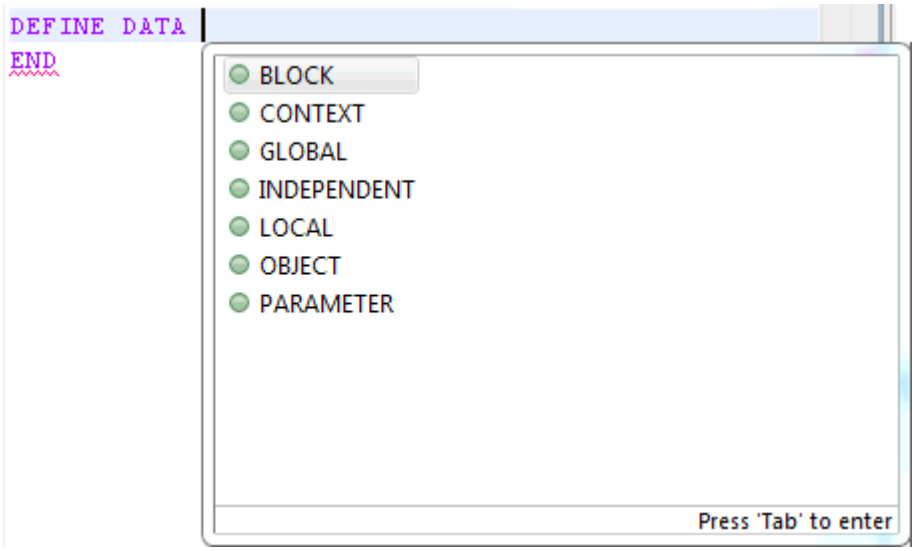
1 Enter a keyword or part of it and press CTRL+SPACEBAR.

  Or:

Do not enter anything and press CTRL+SPACEBAR in an empty line.

A window appears providing a number of choices from which you can select the code that is to be included in your program. The shown choices depend on the context and can be, for example, variable names, statements, program names or templates.

For example, when you type "DEFINE DATA " (including the blank after the letters), all valid keywords are shown.



2 Choose the code that you want to insert into your program.

For more information on content assist, see the Eclipse online help.

Using Context-Sensitive Help

In addition to content assist, the source editor provides context-sensitive help for the following syntax elements:

- Statements (for example, `WRITE`).
- System variables (for example, `*USER`).
- System functions (for example, `COUNT`).
- Parameters (for example, `AD`).

To invoke context-sensitive help

1 In the source editor, move the insertion point to the keyword within a syntax element for which you require help.

2 Press F1.

The corresponding section in the Natural language documentation is listed in the **Help** view.

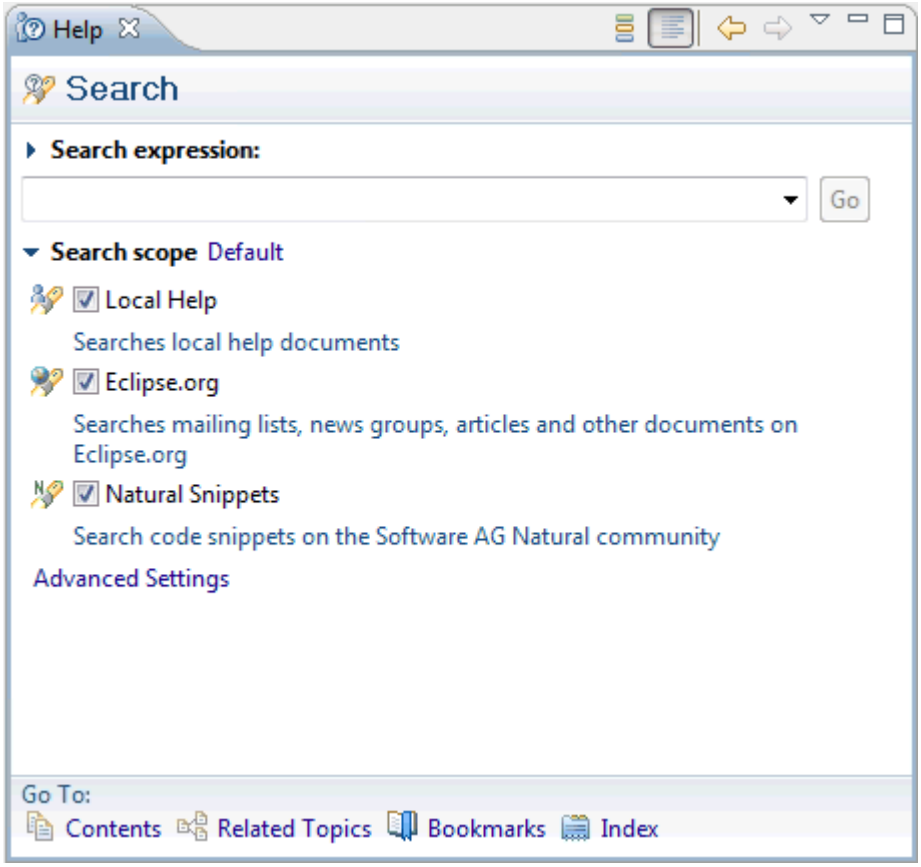
NaturalONE uses the host type that is currently defined on the **Runtime** page of the project properties to determine whether the Natural language documentation for the mainframe, UNIX, OpenVMS or Windows platform is to be shown. In case the host type is "<unknown>", the language documentation for the mainframe is shown by default.

Note: The descriptions in the Natural language documentation may contain references to parts of the Natural documentation which are not included in the NaturalONE documentation set. If you want to view the referenced pages, refer to the Natural documentation at ***http://documentation.softwareag.com/*** (Empower login required).

Using Code Snippets from the Natural Community

The Natural Code Samples Forum of the Natural community contains Natural code samples (also called “code snippets”) that you can include in your own source code. Code samples are written by Software AG employees, customers or partners. If you have code samples that you want to share, you can join the Natural Code Samples Forum at ***http://techcommunity.softwareag.com/ecosystem/communities/public/adanat/products/natural/code-samples/***.

You can search the Natural community for code snippets - directly from Eclipse. To do so, you use the standard search function of the Eclipse **Help** view. NaturalONE automatically adds a **Natural Snippets** entry to the search scope. Using copy-and-paste, you can copy information from a found code snippet into your own source code.



To search for code snippets

1 From the **Help** menu, choose **Search**.

2 Click the **Search scope** link to expand the search scope section.

3 Make sure that the check box next to the **Natural Snippets** entry is enabled.

> **Tip:** If you only want to return Natural snippets in the search result, disable all other check boxes in the search scope section.

4 In the **Search expression** text box, enter the string for which you want to search.

For example, if you are looking for code snippets that deal with the `REQUEST DOCUMENT` statement, you can enter the string "request".

5 Choose the **Go** button.

If code snippets are found, they are listed under the heading **Natural Snippets**. The number of found hits is shown next to this heading.

6 Click the heading of a search result.

This opens the code snippet in the **Help** view.

7 Copy all or part of the coding and paste it into your own source code.

To modify the search parameters

1 From the **Help** menu, choose **Search**.

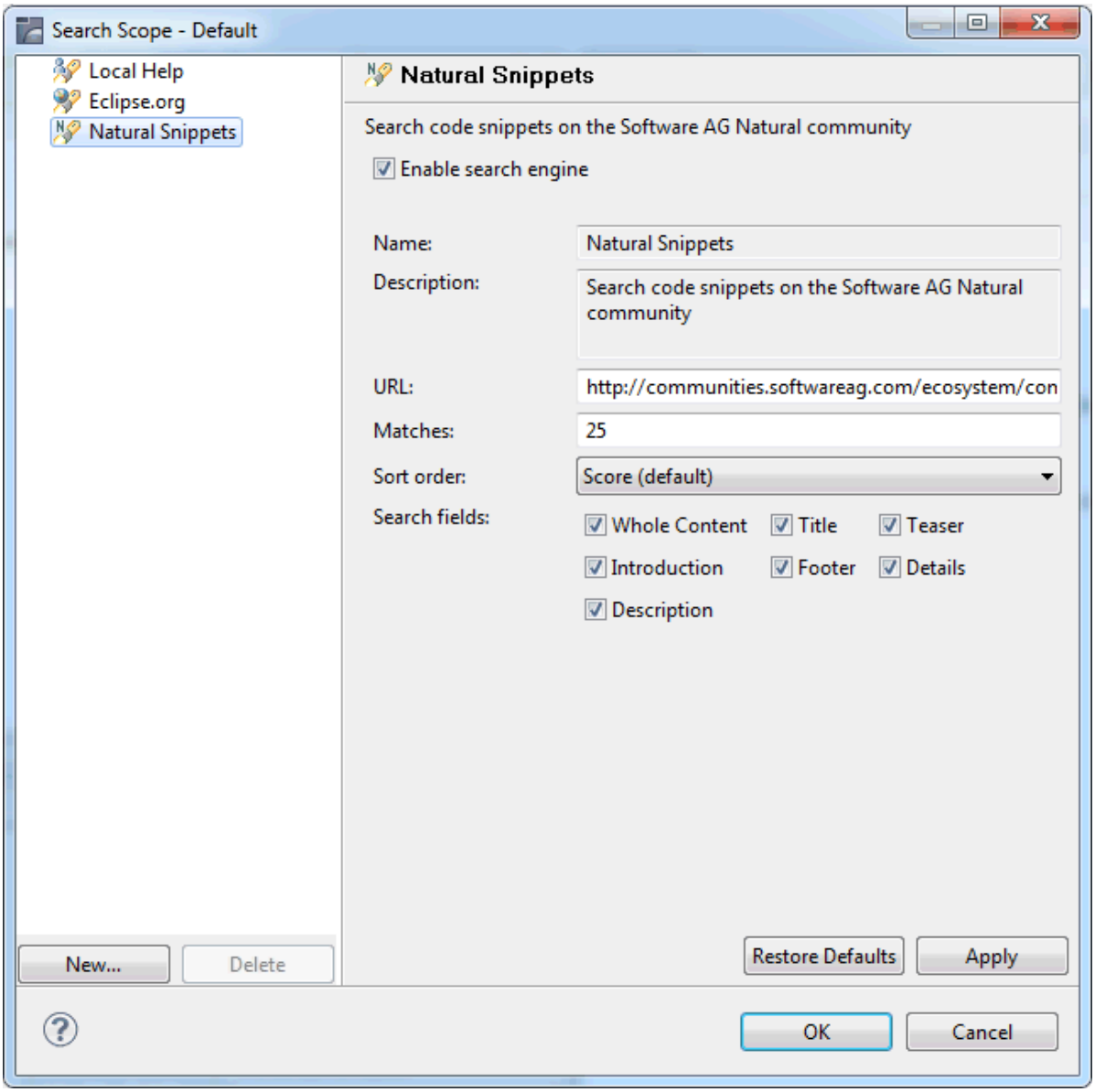
2 Click the **Search scope** link to expand the search scope section.

3 Click the **Advanced Settings** link.

The **Search Scope** dialog box appears.

4 In the tree on the left, select **Natural Snippets**.

The following information is shown.



5 Make sure that the **Enable search engine** check box is selected.

> **Caution:** Do not modify the provided URL. Otherwise, you will not be able to search for code snippets. Using the **Restore Defaults** button, you can always return to the default settings.

6 In the **Matches** text box, specify the maximum number of code snippets that are to be returned. It is recommended that you do not return more than 50 code snippets.

7 From the **Sort order** drop-down list box, select the order in which the search results are to be sorted. By default, the "Score" order is used.

8 Select the check box for each search field that is to be considered during the search, or deselect the check box for each search field that is not to be considered during the search.

9 Choose the **OK** button.

Working with Tasks

When you are working with the source editor and you want to set reminders (for example, for programming steps that still need to be done), you can add tasks to your source code. The tasks are also shown in the **Tasks** view. You can add tasks in different ways, as described below. For more information on tasks, see the Eclipse online help.

The following topics are covered below:

- Adding a Task Tag to the Source Code
- Adding a Task Using a Dialog Box
- Using the Tasks View

Adding a Task Tag to the Source Code

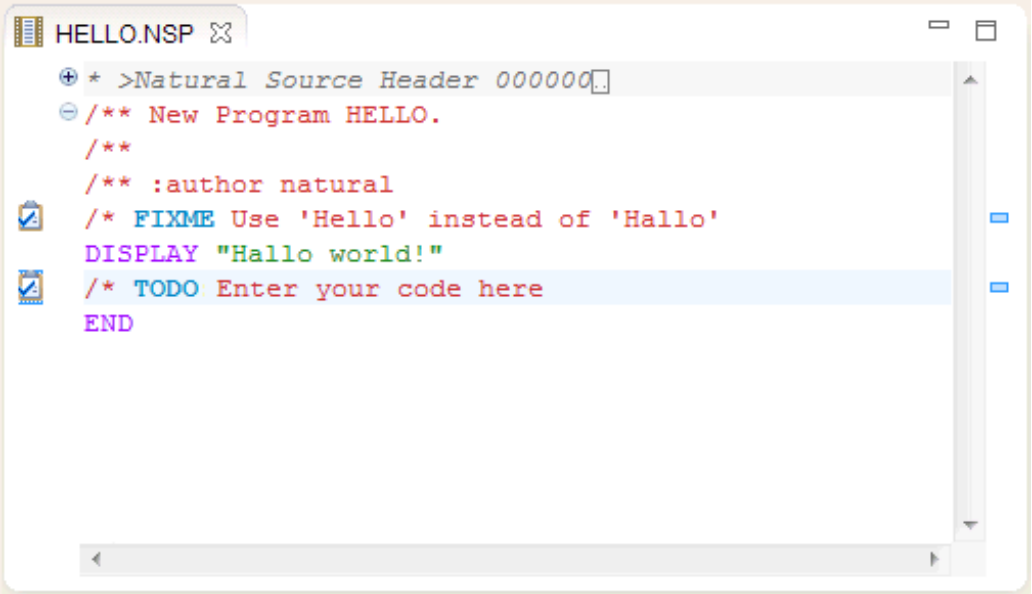
NaturalONE uses the task tags that are defined for the Java compiler (**Preferences > Java > Compiler > Task Tags**). Standard Java task tags are `TODO` and `FIXME`, but you can also define your own task tags. See also the **Parser > Task Tags** page in the Natural preferences.

Note: Other than the Java compiler, NaturalONE does not support project-specific configurations.

The color for the task tags used in the Natural sources can be changed in the Natural preferences. See ***Source Editor > Syntax Coloring*** in *Setting the Preferences*.

To add a task tag to the source code

1 In the source editor, enter a comment containing a task tag, followed by any text you want. Example:



2 Save your changes to display the new task tag in the **Tasks** view. Such a task is of type "Natural Task".

Adding a Task Using a Dialog Box

If you do not want to add a visible task tag to your source code, you can add a task using a dialog box. In this case, only the task marker is shown on the left side of the source editor.

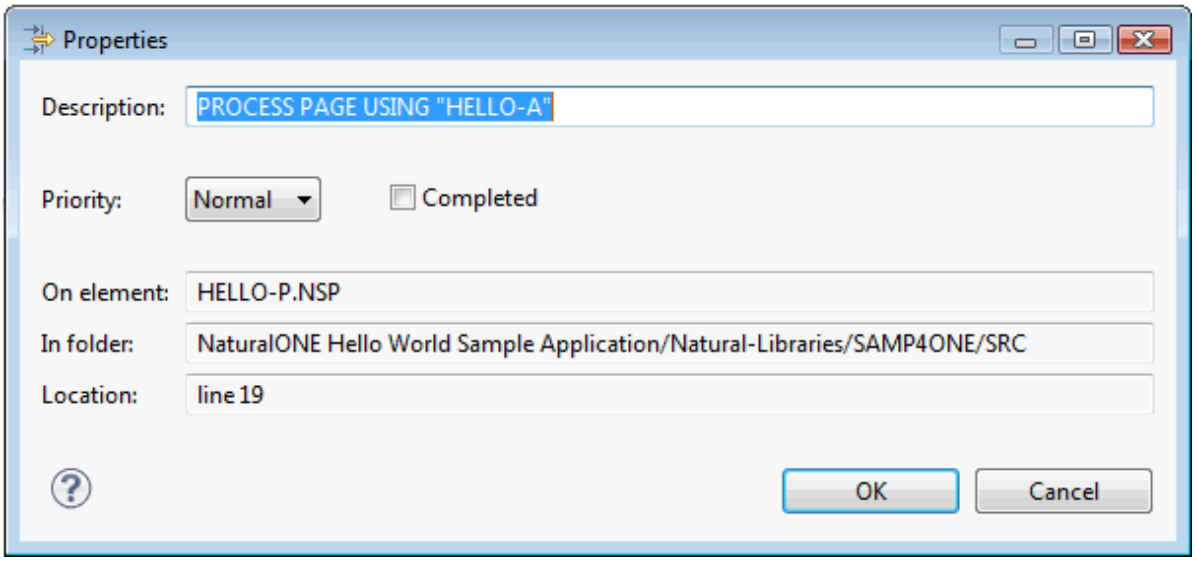
To add a task using a dialog box

1 In the source editor, position the mouse on the marker bar, directly next to the line for which you want to add a task.

> **Note:** The marker bar is the narrow gray area at the very left of the source editor.

2 Open the context menu for the marker bar and choose **Add Task**.

A dialog box appears.



Initially, the **Description** text box contains the text from the selected line. This is the text which will also be shown in the **Tasks** view. You can change this to any text you want.

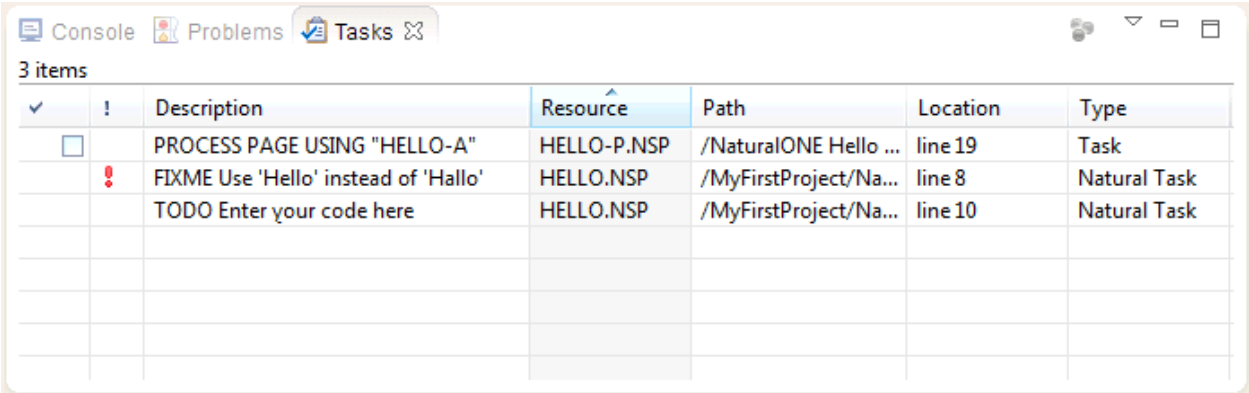
The file name, path and line number are automatically provided.

3 Choose the **OK** button.

The task is now shown in the **Tasks** view. This task is of type "Task", that is, it is a standard Eclipse task.

Using the Tasks View

The **Tasks** view is a standard Eclipse view. It is part of the NaturalONE perspective. If it is currently not shown, you can redisplay it with **Window > Show View > Other > General > Tasks**.



When you double-click a task in the **Tasks** view, the corresponding file is automatically opened and the appropriate line is selected. For more information on the **Tasks** view, see the Eclipse online help.

Working with Bookmarks

When you are working with the source editor, you can add a bookmark for a source code line. A bookmark icon for this line will then appear in the marker bar, and a new entry is added to the **Bookmarks** view. The **Bookmarks** view enables you to return to a file quickly.

The **Bookmarks** view is a standard Eclipse view. It is not part of the NaturalONE perspective. You can display it with **Window > Show View > Other > General > Bookmarks**.

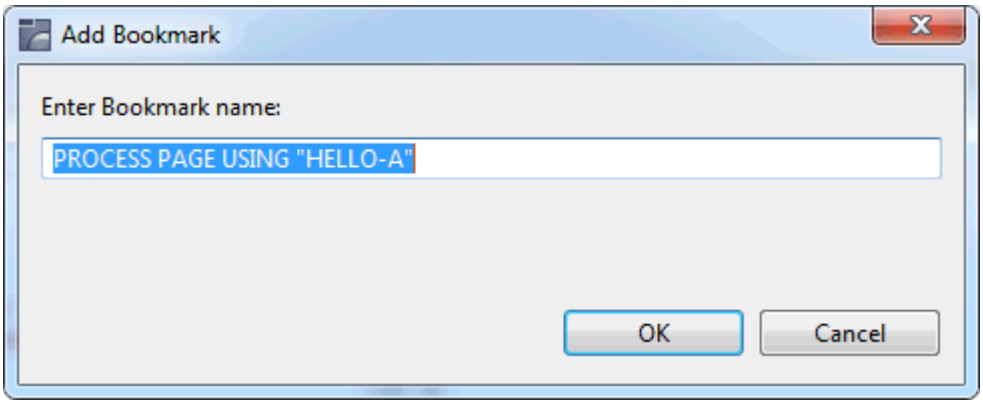
To add a bookmark for a specific source code line

1 In the source editor, position the mouse on the marker bar, directly next to the line for which you want to add a bookmark.

Note: The marker bar is the narrow gray area at the very left of the source editor.

2 Open the context menu for the marker bar and choose **Add Bookmark**.

A dialog box appears.



Initially, the text box contains the text from the selected line. This is the text which will also be shown in the **Bookmarks** view. You can change this to any text you want.

3 Choose the **OK** button.

For more information on bookmarks, see the Eclipse online help.

Error Handling in the Source Editor

In case of a syntax error, the vertical ruler on the left side of the editor contains a red error marker, indicating the error position. You may have several errors in a source. The incremental parser recovers from different error situations and parses the code until the end of the source. If the source contains multiple syntax errors, you can directly go to a different error location using one of the red error markers in the overview ruler on the right side of the editor.

The general preferences for the annotations (**Preferences > General > Editors > Text Editors > Annotations**) also apply for NaturalONE. For example, when configured accordingly, squiggly lines appear under the word which causes the error and an annotation is shown next to the affected line. After a program containing errors has been saved, the errors are shown persistently in the **Problems** view.

A tooltip with the Natural error number and error text appears when you position the mouse over an annotation.

See also: ***Problems in Your Natural Sources***

Going to a Specific Natural Line Number

The **Go to Natural Line** command is similar to the standard Eclipse **Go to Line** command. The **Go to Line** command refers to the Eclipse line numbers. The **Go to Natural Line** command, however, refers to the four-digit Natural line numbers which are used on a Natural server. Due to the existence of a source header in the Eclipse workspace and to different increments that may be used on a Natural server, the Natural line numbers are not identical with the Eclipse line numbers. See also ***Line Numbers***.

The **Go to Natural Line** command is helpful, for example, when a runtime error occurs in the production environment and the corresponding Natural sources are located in an Eclipse workspace. You can use the information that was provided with the runtime error (that is, the object name and four-digit Natural line number): you open the corresponding Natural source in your workspace and then use the **Go to Natural Line** command in order to jump to the corresponding line in the source editor.

To go to a specific Natural line number

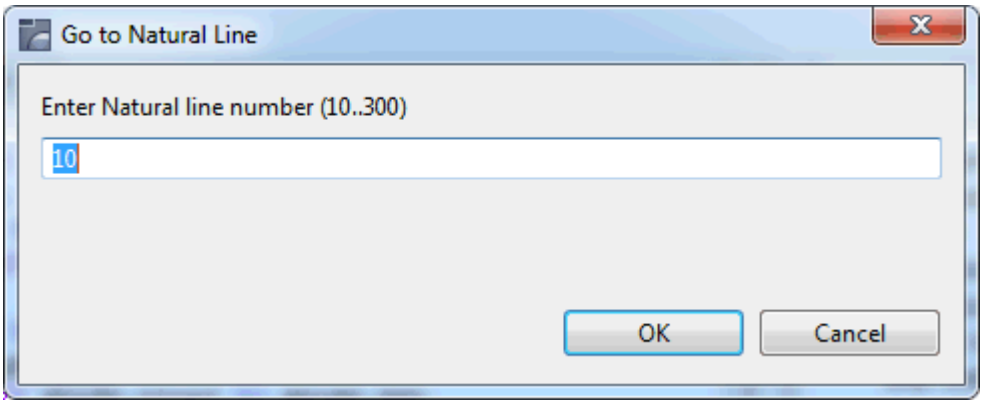
1 Open the source which contains the required line in the source editor.

2 From the **Navigate** menu, choose **Go to Natural Line**.

Or:

Press CTRL+G.

The **Go to Natural Line** dialog box appears.



3 Enter the Natural line number.

You need not specify all four digits of the Natural line number, leading zeros can be omitted.

4 Choose the **OK** button.

Note: The **OK** button is not enabled when the specified Natural line number does not exist in the current source or when a character other than a number has been specified.

Opening Referenced Objects and Jumping to Variable and Internal Subroutine Definitions

Applies only to local mode (that is, when the sources are available in a project in the Eclipse workspace).

You can open a referenced object directly from the source editor. For example, when your source code contains the following statement, you can directly open the local data area with the name "LDA01":

```
LOCAL USING LDA01
```

When the referenced object is not yet available in the Eclipse workspace (that is, when it is listed as "<Unknown>" in the **Dependencies** view), a dialog box appears asking whether you want to download the object from the server.

Note: You can also open a referenced object from the **Dependencies** view.

In addition to opening referenced objects, you can also jump to variable definitions and you can use hyperlinking.

To open a referenced object

1 In the source editor, put the cursor on the name of the referenced object.

2 Invoke the context menu and choose **Open**.

   Or:

   Press F3.

   The referenced object is now opened in the appropriate editor.

To jump to a variable definition

1 In the source editor, put the cursor on the name of the variable.

2 Invoke the context menu and choose **Open**.

   Or:

   Press F3.

   When the variable definition is located in the `DEFINE DATA` block of the current object, it is selected there. When it is located in a different object (for example, in a local data area), the object is opened and the variable definition is selected there.

To jump to an internal subroutine definition

1 In the source editor, put the cursor on the name of the internal subroutine.

2 Invoke the context menu and choose **Open**.

   Or:

   Press F3.

   The subroutine name of the internal subroutine definition (`DEFINE SUBROUTINE` statement) is selected.

To use hyperlinking

1 In the source editor, press the modifier key (by default, this is CTRL) and move the mouse over your source code.

When hyperlinking is enabled, a hyperlink appears when the mouse is positioned over a referenced object or variable name.

Note: By default, hyperlinking is enabled in the general preferences (**Preferences > General > Editors > Text Editors > Hyperlinking**). A special option is available for NaturalONE: **Natural Element**.

2 Click the hyperlink.

The corresponding action is performed: either a referenced object is opened or you jump to the variable definition.

How the Jump to a Variable or Subroutine Definition in a Copy Code is Realized

If a copy code is opened, it can only be parsed without any context. In order to realize a jump to variable and internal subroutine definitions, the first of probably multiple possible root sources which include this copy code is selected via the **Dependencies** view.

This selected root source is then parsed internally. Based on the result of the internal parsing, the location of the variable or internal subroutine definition is determined.

Externalizing Code Fragments

Applies only to local mode (that is, when the sources are available in a project in the Eclipse workspace).

You can move a code fragment from the current source to a new object in the same library. This is helpful, if you want to reuse this piece of code. The new object can be of the following object type: subprogram, subroutine, copycode or program.

As long as you do not close the editor from which you have externalized the code fragment, you can undo/redo your last externalization (this is standard Eclipse functionality). The editor must be active for this purpose.

To externalize a code fragment

1 In the source editor, select the code fragment that you want to externalize.

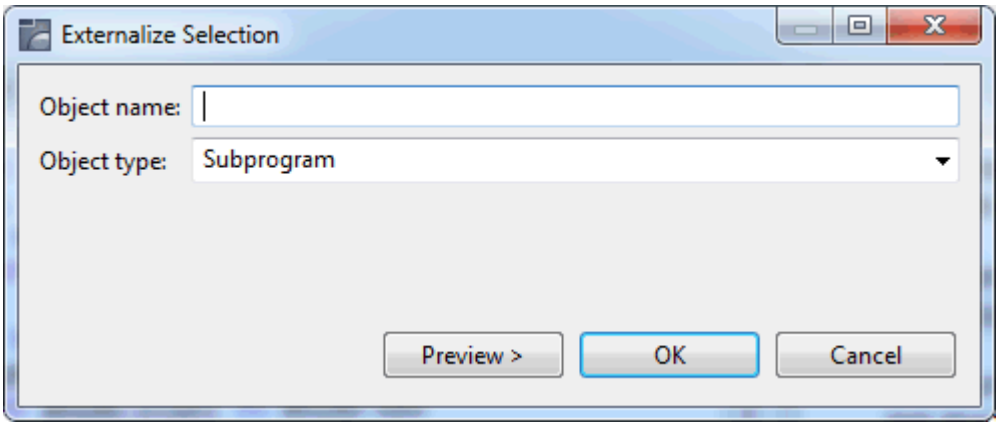
Note: When you have not selected all lines which belong, for example, to a loop, the selection is automatically expanded in order to include all required lines.

2 Invoke the context menu and choose **Advanced Edit Features > Externalize Selection**.

Or:

Press ALT+SHIFT+S.

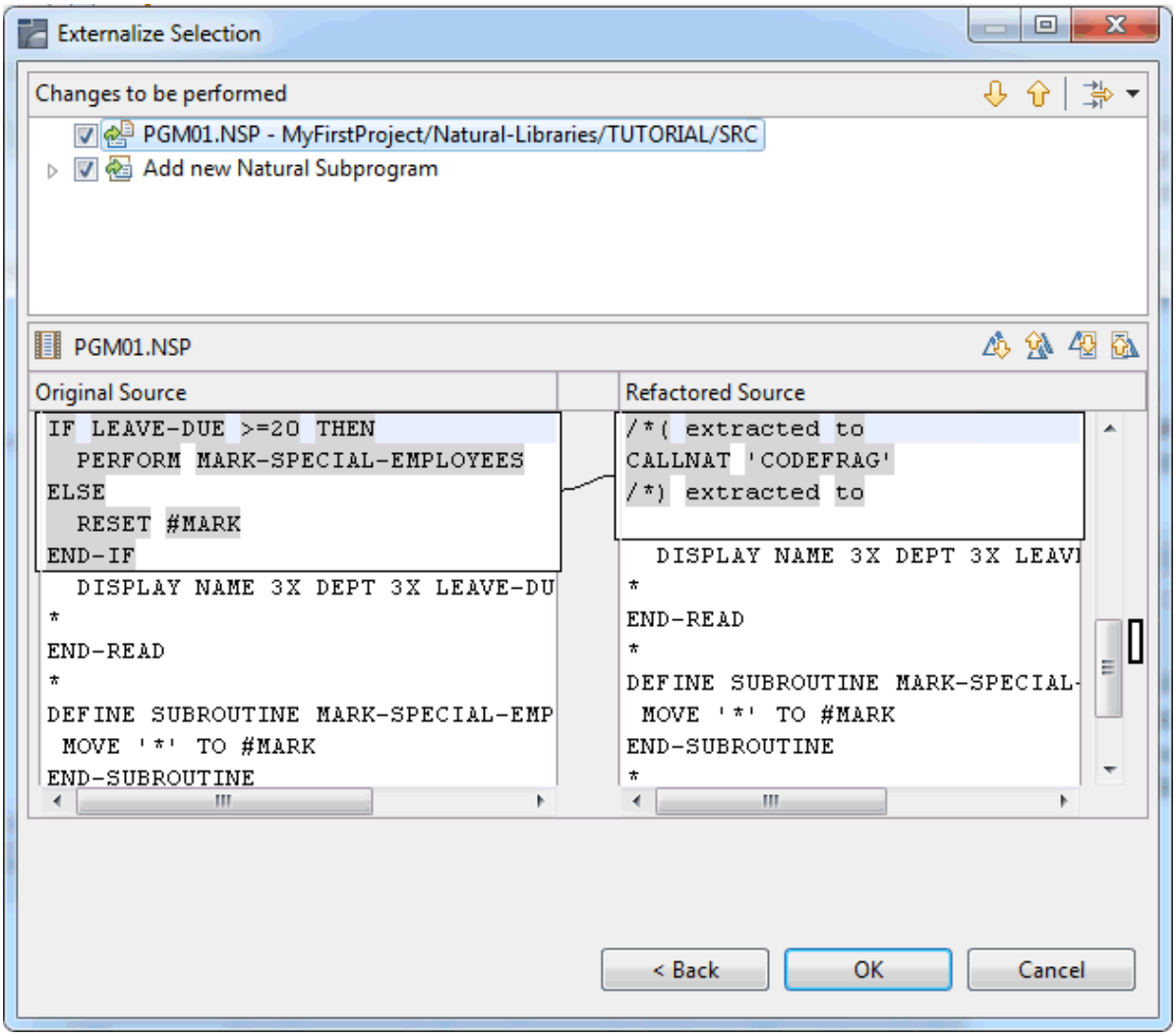
The following dialog box appears.



3 Specify a name for the new object and select an object type.

4 Optional. Choose the **Preview**.

The following dialog box appears.



This dialog box shows how your code will be changed. At the position of the selected code fragment, the corresponding call will be inserted. For example, if you want to externalize code to a subprogram, a `CALLNAT` statement will be inserted. A comment will also be inserted, indicating that code has been extracted.

Note: When you deselect a check box at the top of the dialog box, the corresponding change will not be performed.

5 Choose the **OK** button.

The selected code is removed from the current source and a new object containing this code is shown in the **Navigator** view or in the **Natural Navigator** view.

Inserting Call or Include Statements

Applies only to local mode (that is, when the sources are available in a project in the Eclipse workspace).

You can invoke a dialog box in which you can specify a Natural object that is to be referenced in the current source. The corresponding statement is then automatically inserted in your source code.

The following table gives an overview of the different statements that are inserted for the different object types:

Object Type	Statement

Adapter	`PROCESS PAGE USING "adapter-name"`
	...
	`END-DECIDE`

Copycode	`INCLUDE copycode-name`
Function	`function-name(< [parameter]... >)`
Global data area	`GLOBAL USING gda-name`
Local data area	`LOCAL USING lda-name`
Map	`INPUT USING MAP "map-name"`
Parameter data area	`PARAMETER USING pda-name`
Program	`FETCH RETURN "program-name"`
Subprogram	`CALLNAT "subprogram-name" [parameter]...`
Subroutine	`PERFORM subroutine-name [parameter]...`

To insert a call or include statement

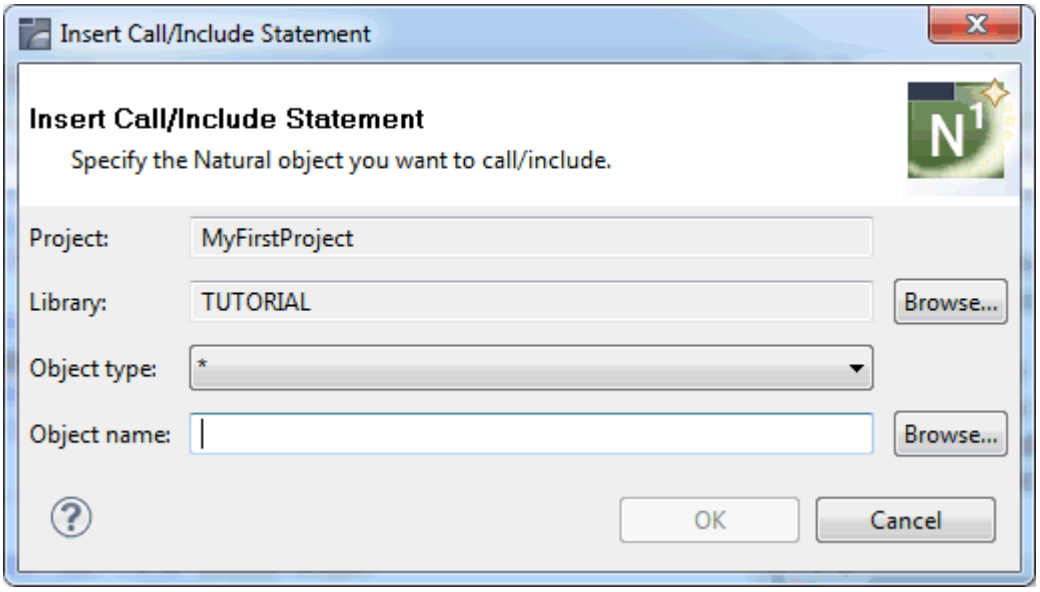
1 In the source editor, position the cursor where you want to place the object reference.

2 Invoke the context menu and choose **Advanced Edit Features > Insert Call/Include Statement**.

Or:

Press ALT+SHIFT+C.

The following dialog box appears.



3 Optional. Use the **Browse** button to select the library that contains the object that you want to reference.

4 From the **Object type** drop-down list box, select the type of object that you want to reference.

The asterisk (*) which is shown by default is used as a filter for the **Select Natural Object** dialog box that can be accessed using the **Browse** button for the object name. If you do not want to use this **Browse** button, it is important that you select the appropriate object type for the object name that you specify. Otherwise, the **OK** button will not be enabled.

5 In the **Object name** text box, enter the name of the object that you want to reference.

If you want to use the **Browse** button, you can enter wildcard characters: an asterisk (*) for optional multiple characters, and/or a question mark (?) for a single character. This specification is only used as a filter for the **Select Natural Object** dialog box.

Or:

Choose the **Browse** button to select the object from the **Select Natural Object** dialog box.

Unless a Natural object is currently selected, the **Select Natural Object** dialog box only shows the objects matching the type (if any) displayed in the **Object type** drop-down list box. In addition, if the **Object name** text box contains wildcard characters, the **Select Natural Object** dialog box only shows the objects matching this name pattern.

6 Choose the **OK** button.

The corresponding statement is inserted at the cursor position.

Importing Data Fields

Applies only to local mode (that is, when the sources are available in a project in the Eclipse workspace).

You can invoke a dialog box in which you can select data field definitions from another object that are to be imported into the current source.

Note: The data field definitions are only available for Natural objects that contain a `DEFINE DATA` statement.

To import data fields

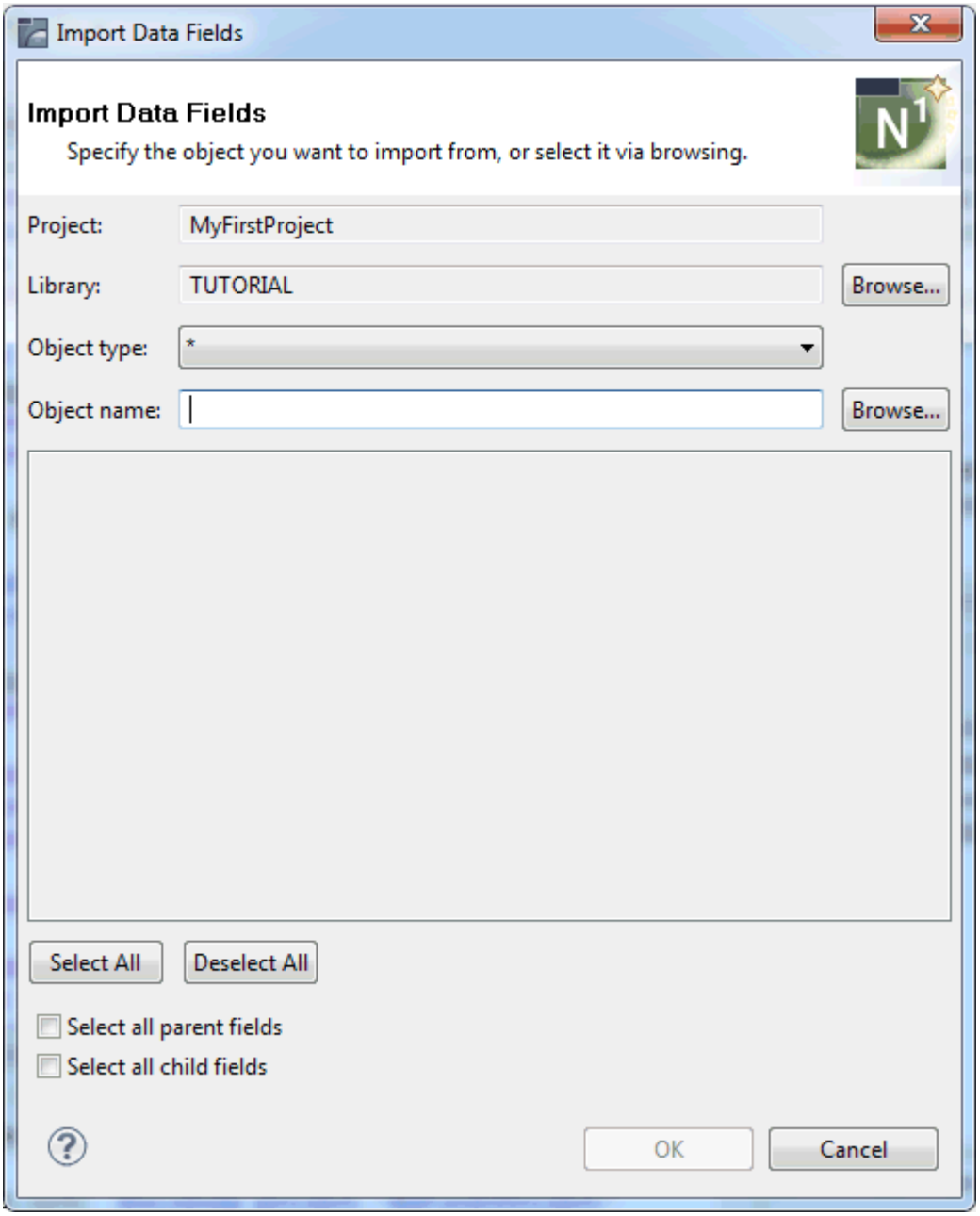
1 In the source editor, position the cursor where you want to place the imported data field(s). This is usually the first position of a line within the `DEFINE DATA` block.

2 Invoke the context menu and choose **Advanced Edit Features > Import Data Fields**.

   Or:

   Press ALT+SHIFT+I.

   The following dialog box appears.



3 Optional. Use the **Browse** button to select the library that contains the object from which you want to import data fields.

4 From the **Object type** drop-down list box, select the type of object from which you want to import data fields.

The asterisk (*) which is shown by default is used as a filter for the **Select Natural Object** dialog box that can be accessed using the **Browse** button for the object name. If you do not want to use this **Browse** button, it is important that you select the appropriate object type for the object name that you specify. Otherwise, the fields that can be imported will not be shown.

5 In the **Object name** text box, enter the name of the object that you want to import.

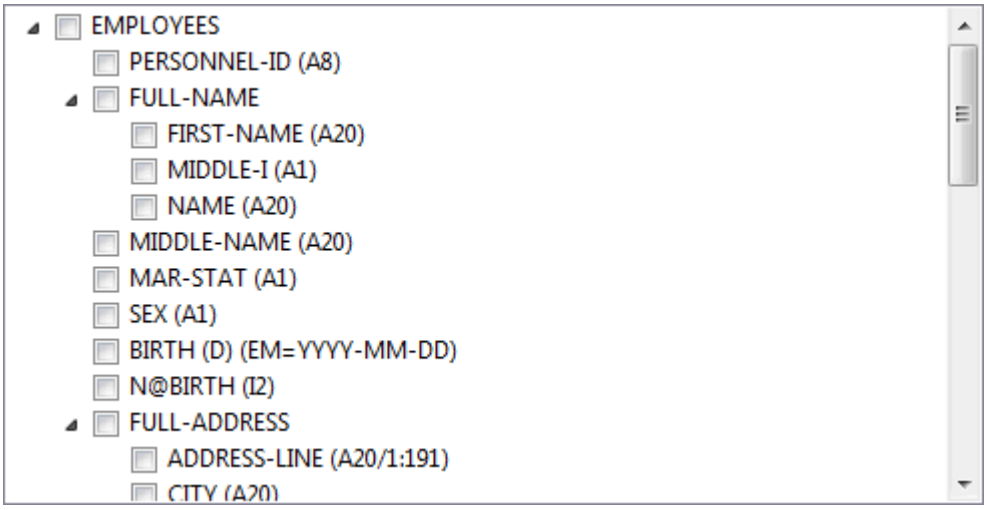
If you want to use the **Browse** button, you can enter wildcard characters: an asterisk (*) for optional multiple characters, and/or a question mark (?) for a single character. This specification is only used as a filter for the **Select Natural Object** dialog box.

Or:

Choose the **Browse** button to select the object from the **Select Natural Object** dialog box.

Unless a Natural object is currently selected, the **Select Natural Object** dialog box only shows the objects matching the type (if any) displayed in the **Object type** drop-down list box. In addition, if the **Object name** text box contains wildcard characters, the **Select Natural Object** dialog box only shows the objects matching this name pattern.

All data fields that can be imported from the selected object are listed in the **Import Data Fields** dialog box. For example:



6 Activate the check box for each data field that you want to import.

When the **Select all parent fields** check box is selected, all parents fields are automatically selected when you select a child field.

When the **Select all child fields** check box is selected, all child fields are automatically selected when you select a parent field.

You can also use the command buttons **Select All** and **Deselect All** to select or deselect all check boxes.

7 Choose the **OK** button.

Note: This button is only enabled when at least one data field to be imported has been selected.

The selected data fields are imported into the source code.

Generating Counter Fields

Applies only to local mode (that is, when the sources are available in a project in the Eclipse workspace).

A counter field (`C*` notation) is used to retrieve the number of occurrences of a multiple-value field or a periodic group from an Adabas database. For example, the counter field `C*INCOME` returns the count of the number of occurrences for the periodic group `INCOME`. For more information on referencing the internal count for a database array, see the *Programming Guide* for your Natural platform.

You can generate counter fields for periodic groups and multiple-value fields. You can do this in all objects where a `DEFINE DATA` statement may be used (for example, in programs, data areas, or processing rules in maps).

To generate a counter field

1. In the source editor, position the cursor in the name of a multiple-value field or periodic group within the `DEFINE DATA` block.
2. Invoke the context menu and choose **Advanced Edit Features > Generate Counter Field**.

   Note: This command is only visible for a multiple-value field or periodic group.

   Or:

   Press ALT+SHIFT+G.

   A new line is generated into the `DEFINE DATA` block, before the selected multiple-value field or periodic group.

   The following examples show where a generated counter field occurs.

   - Periodic group:

```
DEFINE DATA LOCAL
   1 EMP-VIEW VIEW OF EMPLOYEES
     2 NAME
     2 CITY
     2 C*INCOME
     2 INCOME (1:12)
       3 SALARY
       ...
```

- Multiple field:

```
DEFINE DATA LOCAL
   1 EMP-VIEW VIEW OF EMPLOYEES
     2 NAME
     2 C*LANG
     2 LANG (1:3)
     ...
```

- Multiple field within a periodic group:

```
DEFINE DATA LOCAL
   1 EMP-VIEW VIEW OF EMPLOYEES
     2 NAME
     2 C*BONUS (1:12)
     2 INCOME (1:12)
       3 SALARY
       3 BONUS (1:2)
       ...
```

In this case, the counter is generated directly above the periodic group, with the same index as the periodic group.

The index of the periodic group is always used, even if the periodic group is not defined within the view:

```
DEFINE DATA LOCAL
   1 EMP-VIEW VIEW OF EMPLOYEES
     2 NAME
     2 C*BONUS (1:12)
     2 BONUS (1:12,1:2)
     ...
```

Adding and Removing Comments

You can mark several lines in your source code as comment lines all at once, or you can remove the comment marks from several lines.

To mark several lines as comment lines

1 In the source editor, select all lines that you want to turn into comment lines.

2 Invoke the context menu and choose **Add Natural Comment Lines**.

   Or:

   Press CTRL+ALT+C.

   All selected lines are turned into comment lines, where each comment line starts with an asterisk (*) followed by one blank character.

To remove the comment marks from several lines

1 In the source editor, select all comment lines from which you want to remove the comment marks.

2 Invoke the context menu and choose **Remove Natural Comment Lines**.

   Or:

   Press CTRL+ALT+U.

   The comment marks (* , ** or /*) are removed from the beginnings of all selected lines.

Note: When you select part of a line, the selection is automatically expanded in order to include the entire line.

Toggle Comments

You can select several lines in your source code and toggle between commenting or uncommenting the selected lines.

To toggle comment lines

1 In the source editor, select all lines where you want to toggle between commenting and uncommenting.

2 Invoke the context menu and choose **Toggle Natural Comment Lines**.

Or:

Press CTRL+7.

All selected lines will be turned into comment lines.

3 If you press CTRL+7 again on the same selection of lines, you revert the previous action and the original status of the selected lines is displayed again.

Note: When you select part of a line, the selection is automatically expanded in order to include the entire line.

Protecting Source Code Lines

You can protect one or more lines in your source code by using read-only tags. Protected source code lines cannot be edited.

Protected lines have a special background color. This background color can be changed in the Natural preferences. See ***Syntax Coloring*** in *Setting the Preferences*.

To protect a single line

- At the end of the line to be protected, enter a blank character and append the following tag:

```
/*<RO>
```

To protect a code block

1 At the end of the line where the block to be protected begins, enter a blank character and append the following tag:

```
/*<RO>>
```

2 At the end of the line where the block to be protected ends, enter a blank character and append the following tag:

```
/*<<RO>
```

Protected Lines in Sources Generated by Construct or Code Generation

When the first line of a source (that is, the line directly below the Natural source header) starts with `**SAG GENERATOR`, this indicates that the source has been generated by Construct or by the Code Generation component of NaturalONE. Most lines in such a source are protected and cannot be edited. The following rules apply:

- You can edit the lines that start with `**SAG`.
- You can edit the lines between `**SAG DEFINE EXIT` and `**SAG END-EXIT`.
- All other lines are protected.

The protection can be enabled or disabled in the Natural preferences. See ***Protection*** in *Setting the Preferences*.

Translating to Upper Case or Lower Case

When case translation is enabled in the Natural preferences, you can translate source code from upper case to lower case, or from lower case to upper case. See ***Case Translation*** in the section *Setting the Preferences*.

Case translation can either be performed manually using the commands described below, or (when the corresponding option is enabled in the preferences) automatically when you save or stow your source code.

To translate text from lower case to upper case

1. In the source editor, select the text you want to translate.
2. Invoke the context menu and choose **Upper Case**.

   Or:

   Press ALT+U.

To translate text from upper case to lower case

1. In the source editor, select the text you want to translate.
2. Invoke the context menu and choose **Lower Case**.

Or:

Press ALT+L.

Notes:

1. When you select the entire source code and then choose one of the above commands, only those parts of the source code are translated to a different case for which the corresponding option has been enabled in the preferences.
2. When you select, for example, only a string within single quotes and the corresponding option has not been enabled in the preferences, the above commands are not enabled.

Indenting the Source Code Lines

The Natural source editor makes use of the standard Eclipse source formatter. However, it is possible to indent the source code lines in such a way that the indentation reflects the structure of the program. This feature corresponds to the Natural system command `STRUCT`.

The number of positions that a line is indented is determined in the Natural preferences. See ***Struct*** in *Setting the Preferences*.

The following types of statements are affected by a structural indentation:

- processing loops (`READ`, `FIND`, `FOR`, etc.),
- conditional statement blocks (`AT BREAK`, `IF`, `DECIDE FOR`, etc.),
- `DO`/`DOEND` statement blocks,
- `DEFINE DATA` blocks,
- inline subroutines.

You can exclude sections of your source code from structural indentation by using the special statements `/*STRUCT OFF` and `/*STRUCT ON`. These statements must be entered at the beginning of a source code line. The source code lines between these two statements will remain as they were before you performed the indentation.

To indent the source code lines

- In the source editor, invoke the context menu and choose **Struct**.

  Or:

  Press CTRL+ALT+S.

  The entire source is indented, according to the settings in the Natural preferences.

17 Using the Map Editor

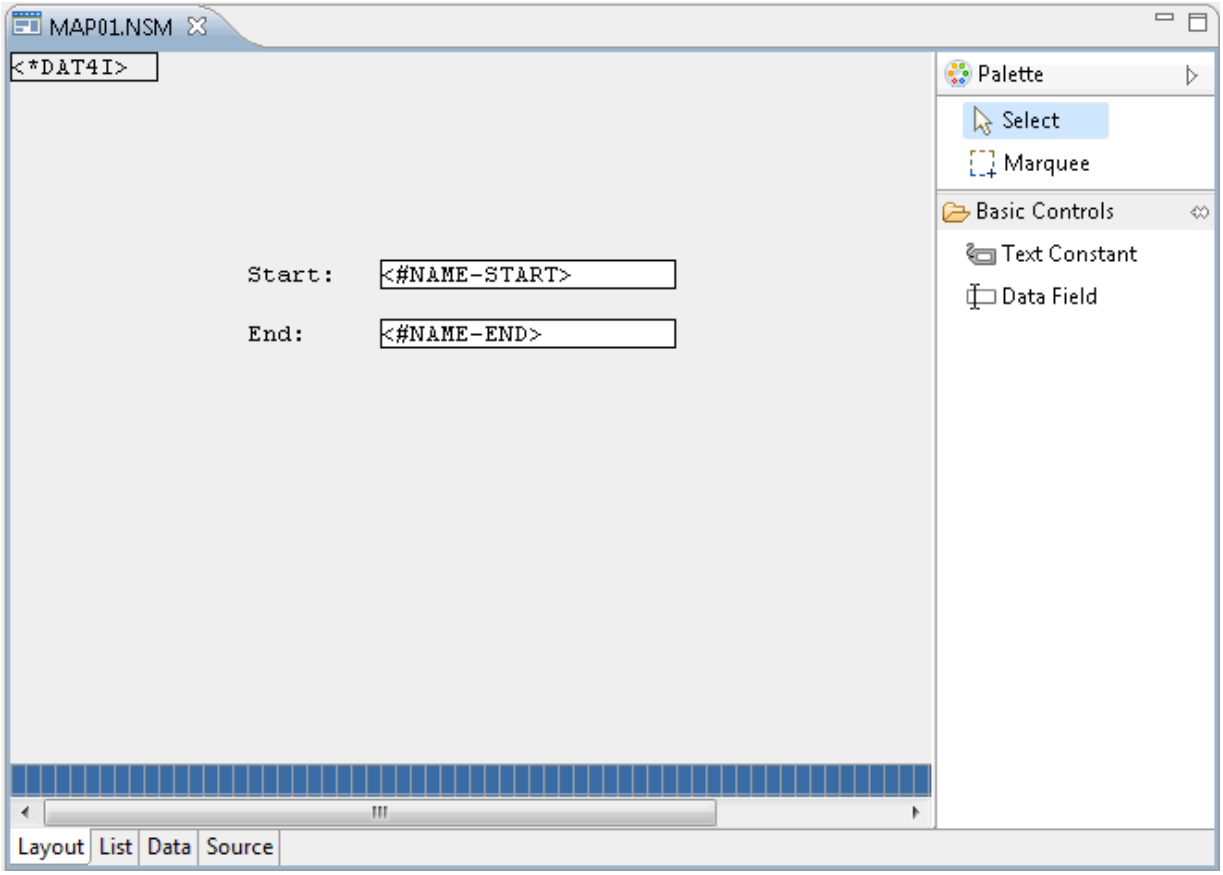
- About the Map Editor 232
- Associated Views 233
- Adding Controls to the Map 236
- Selecting Controls in the Map 238
- Changing the Reference Control for Selection 239
- Managing the Controls in the Map 240
- Defining Rules 241
- Moving Data Definitions to Data Fields 243
- Editing the Code of a Map 243
- Changing the Properties for the Map 244
- Changing the Properties for a Text Constant 246
- Changing the Properties for a Data Field 247
- Changing the Properties for a Rule 253
- Defining Arrays 254
- Viewing the Status Properties 256
- Saving Maps 257
- Stowing Maps 258
- Validating Maps 258
- Error Handling in the Map Editor 259

About the Map Editor

The `INPUT` statement offers the possibility to use predefined map layouts.

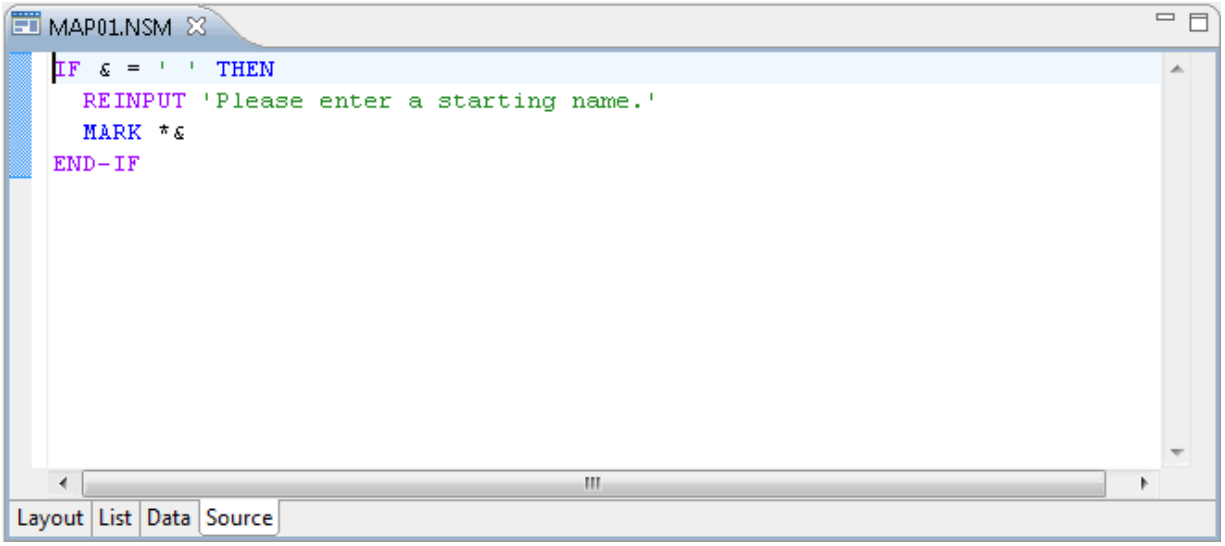
When you create or open a map, the map editor is invoked. The map editor is a multi-page editor which provides the following pages:

- **Layout**
  Shows the graphical representation of the map and provides a palette for inserting controls into the map.



- **List**
  Displays the Natural code corresponding to the current state of the entire map.
- **Data**
  Displays the data definition for the data area that is currently selected in the **Outline** view.

- **Source**
  Only shown when at least one processing rule has been defined. Shows the code for a processing rule.



The behavior of the map editor can be influenced by changing the Natural preferences. See ***Map Editor*** in *Setting the Preferences*.

For further information on maps, see the *Programming Guide* in the Natural documentation for the appropriate platform.

Associated Views

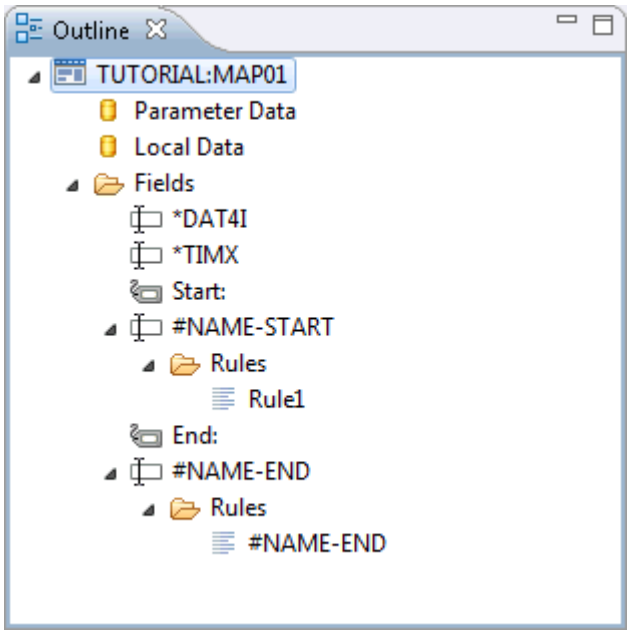
The map editor uses the following views of the NaturalONE perspective:

- Outline View
- Dependencies View

- Properties View

Outline View

This view shows the various components of the map in a tree. This includes data definitions, field definitions and processing rules (map-based and field-based). If a dynamic layout is present, its components are also shown in a separate hierarchy.



You can invoke a context menu, for example, to delete the selected item or to create a new processing rule.

You can navigate to a particular item in the map by selecting it in the **Outline** view, and vice versa.

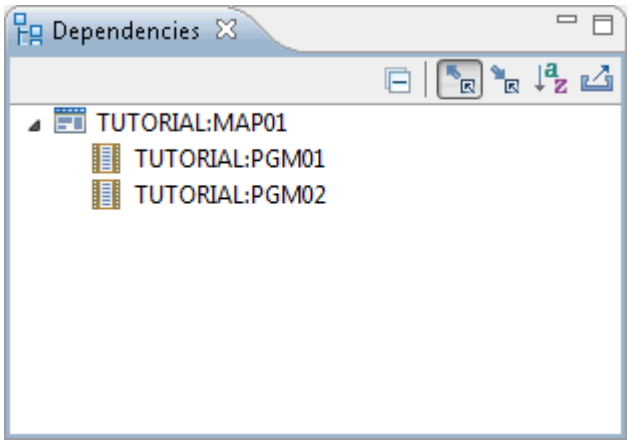
When the definition for an element in the tree has been modified or when a new element is created, a label decoration in the form of an asterisk is shown for this element.

Note: The label decorations for modifications in the map editor are controlled by the preferences under **General > Appearance > Label Decorations**. If you do not want to have these label decorations, just go to the above mentioned preference page and deselect **Natural Map Editor Modifications**.

Errors and warnings are also displayed in the form of label decorations. See *Error Handling in the Map Editor* for further information.

Dependencies View

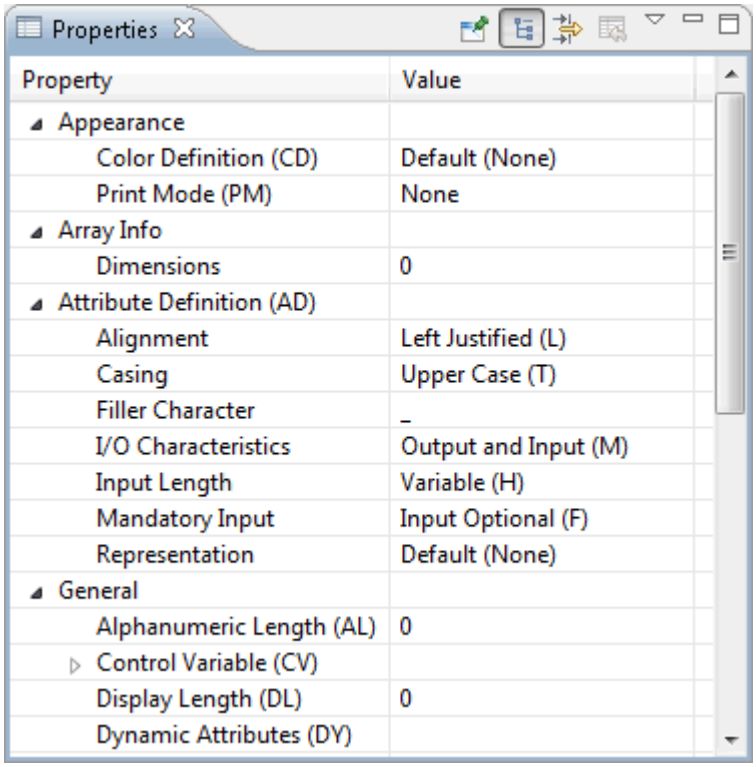
This view shows the dependencies between the map which is currently shown in the active editor window and other objects. For example, when the passive cross-references are currently displayed, you can see all Natural objects that reference the map being edited.



For further information on this view, see *Dependencies View* in the description of the source editor.

Properties View

This view shows the properties of the item that is currently selected in the active map editor window or in the **Outline** view. You can change the values shown in this view. Example for a data field:



Further information on the different types of properties is provided below.

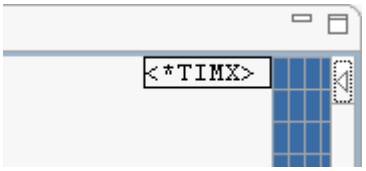
Adding Controls to the Map

You can add controls using the palette, by cloning existing controls, or by importing data definitions.

To add a control to the map

1 Make sure that the **Layout** page is shown.

2 If the palette is currently hidden, display it using its arrow button.



Note: By default, the palette is shown on the right side of the map editor. You can also move it to the other side of the map editor.

3 Click the graphical representation of the required control in the palette.

4 Click the position in the map where you want to place the control. Do not release the mouse button immediately. Keep it pressed and draw the outline of the control until it has the required size. Then release the mouse button.

The new control is now shown in the **Outline** view and in the **Properties** view.

To clone a control

1 Select one or more controls in the **Layout** page. These can be text constants or data fields.

2 Press CTRL and drag the selected control(s) to a different position in the map.

Or:

Use copy and paste. This also works between two different maps. When controls are pasted into the same map, they overlay the positions of the copied controls and you have to move them to the desired positions.

To create a control based on an external data definition (method 1)

1 Make sure that the **Layout** page is shown.

2 Invoke the context menu at the position where you want to place the imported data field(s) and choose **Import Data Fields**.

Or:

Press ALT+SHIFT+I.

The **Import Data Fields** dialog appears.

3 From the dialog, select the data definition(s) to be imported as described in *Importing Data Fields*.

Notes:

1. When the selected field already exists on the map as a modifiable field and can be adapted, a message confirmation dialog appears. This behavior can be influenced by changing the Natural preferences. See *Map Editor* in *Setting the Preferences*.
2. The warning does not appear when the field exists as an **Output Only** field on the map. In this case, the field is imported as a new field at the selected position. To adapt the data definition, perform the following steps before importing the data field:
   a. Select the I/O characteristics in the **Properties** View.
   b. Change the setting from **Output Only** to another attribute value.
   c. Switch the I/O characteristics back to the old setting.

To create a control based on an external data definition (method 2)

1 Switch to the **Data** page.

2 Invoke the context menu at the position where you want to place the imported data field(s) and choose **Advanced Edit Features > Import Data Fields**.

   Or:

   Press ALT+SHIFT+I.

   The **Import Data Fields** dialog appears.

3 From the dialog, select the data definition(s) to be imported as described in *Importing Data Fields*.

   For each imported data definition, a new field node is created for the current data section in the **Outline** view.

4 Move the data definitions to data fields as described in *Moving Data Definitions to Data Fields*.

Selecting Controls in the Map

You can either select controls in the **Layout** page or in the **Outline** view.

The palette that is provided on the **Layout** page offers different tools for selection purposes: the select tool (default) and the marquee tool.

To select controls using the select tool

1 Make sure that **Select** is selected in the palette.

2 In the **Layout** page, click the control that you want to select.

   Or:

   To select more than one control, hold down CTRL and then click each required control in the **Layout** page.

   The selection is automatically reflected in the **Outline** view.

To select controls using the marquee tool

1 Select **Marquee** in the palette.

2 In the **Layout** page, drag the mouse around all controls that you want to include in your selection.

The selection is automatically reflected in the **Outline** view.

To select several controls in the Outline view

- Hold down CTRL and then click each required control in the **Outline** view.

The selection is automatically reflected in the **Layout** page.

To select a whole array

1 Press ALT.

2 In the **Layout** page, click any one of the array's field elements.

Note: To perform a clipboard operation (**Cut**, **Copy**, **Delete**) on an array, it is sufficient to select one or more of its elements. By default, a message dialog appears, prompting you to confirm the operation. You can suppress this message dialog for the future, either within the message dialog itself or in the Natural preferences (see ***Map Editor*** in *Setting the Preferences*). This feature allows clipboard operations to work in conjunction with marquee selection (see above).

Changing the Reference Control for Selection

Although multiple controls can be selected, only one of these controls at any time is the reference control, and is distinguishable from the other selected controls by having solid drag handles instead of hollow ones. For some operations such as control deletion, the reference control is irrelevant. However, for other operations such as field alignment, one of the selected controls must be taken as the basis for the operation, and this is invariably the reference control.

To change the reference control

1 In the **Layout** page, select all required controls.

2 Press and hold down SHIFT.

3 Click the new reference control.

4 Perform the desired action.

Managing the Controls in the Map

To resize or move a control

1 Select the control in the **Layout** page of the map.

2 Use the mouse to resize or move the control.

Or:

Modify the corresponding values in the **Properties** view.

Note: Resizing an array causes the number of array elements to be changed in preference to the width or height of each element. Since only whole rows and columns can be displayed, this may result in nothing appearing to happen if the bounding rectangle for the array was not expanded enough in either direction to accommodate an additional row or column. See also *Selecting Controls in the Map* for information on how to select a whole array prior to resizing it.

To rename a text constant

1 Select the text constant in the **Layout** page of the map or in the **Outline** view.

2 Modify the **Label** value in the **Properties** view.

Or:

In the **Layout** page, click the text constant once more and then type the new label.

To delete a control

1 Select the control in the **Layout** page of the map or in the **Outline** view.

2 Press DEL.

Or:

In the **Outline** view, invoke the context menu and choose **Delete**.

To align several controls

1 Select the controls in the **Layout** page of the map.

2 Invoke the context menu and choose **Align >** *`alignment-option`*.

The controls are aligned relative to the reference control. See also *Changing the Reference Control for Selection*

Defining Rules

You can define processing rules and Predict free rules for data fields. See also *Changing the Properties for a Rule*.

To create a processing rule

1 Select a data field in the **Outline** view.

Or:

Select the top-level node in the **Outline** view if you want to create a processing rule for the entire map.

2 Invoke the context menu and choose **New Rule**.

Or:

Press CTRL+ALT+R.

A new rule is shown in the **Outline** view and the **Source** page of the map editor is automatically opened.

3 Enter the processing rule on the **Source** page. For example:

```
IF & = ' ' THEN
  REINPUT 'Please enter a starting name.'
  MARK *&
END-IF
```

An ampersand (&) in the processing rule will dynamically be replaced with the name of the field.

4 Define the rank for the rule in the **Properties** view.

The rank defines the sequence in which the rules for the different fields are to be processed.

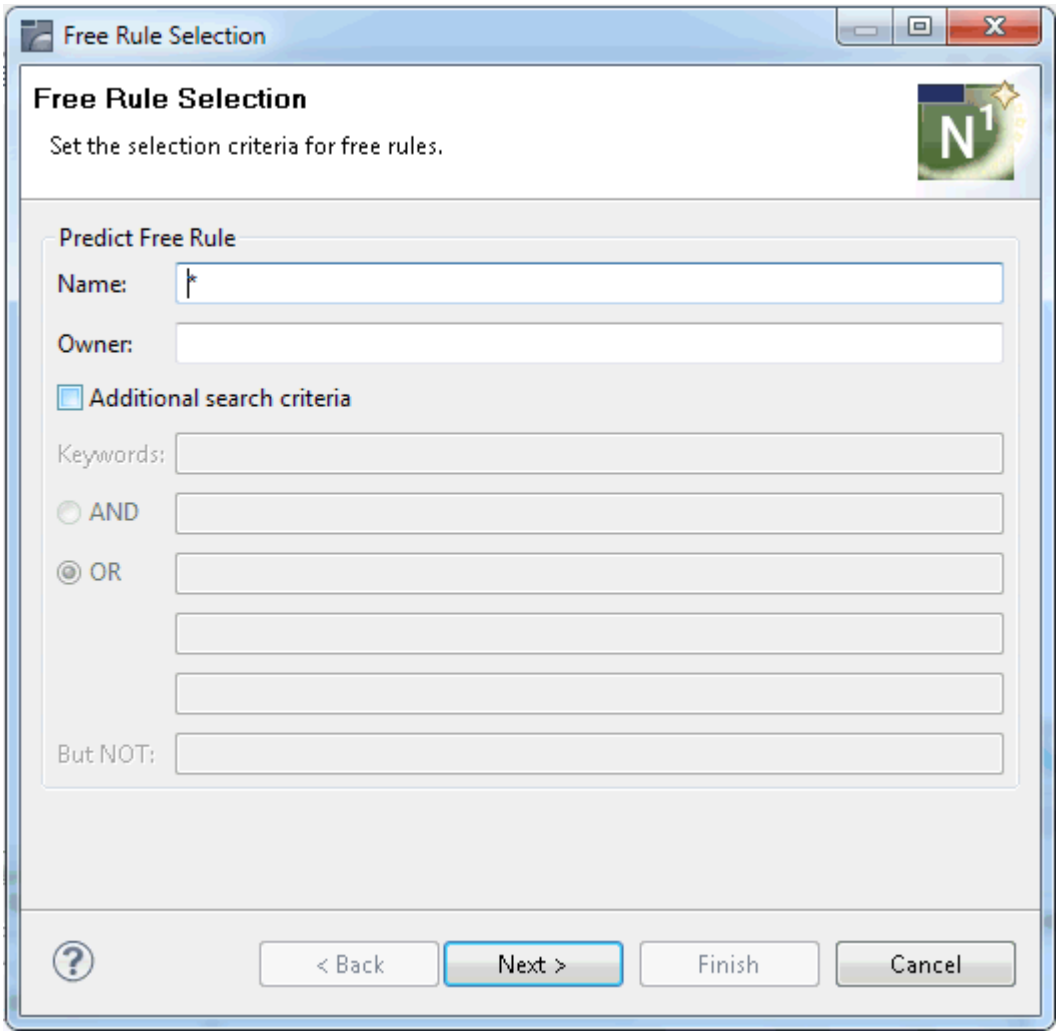
To link Predict free rules

Only available when Predict is available in the corresponding Natural server environment.

1 Select a data field in the **Outline** view.

2 Invoke the context menu and choose **Free Rule Wizard**.

The following dialog box appears.



3 Enter the rule name, the Predict owner and up to five keywords under which the rule is stored in Predict.

Asterisk notation can be used for the rule name, the keywords can be combined with the Boolean operators `AND` and `OR`, and a `BUT NOT` keyword can also be specified.

4 Choose the **Next** button.

A list of free rules is shown.

5 Select the rule(s) you want to link to the field.

6 Optional. Choose the **Next** button to display a preview of the selected rule(s).

7 Choose the **Finish** button to link the rule(s) to the field.

To unlink Predict free rules

1 Select the rule in the **Outline** view.

2 Press DEL.

Or:

Invoke the context menu and choose **Delete**.

Notes:

1. Unlinking a rule does not delete the rule on the Predict server.
2. An automatic rule that has been implicitly created for a view field cannot be unlinked.

Moving Data Definitions to Data Fields

You can move a data definition from a local data area or parameter data area which is defined for the map to a data field.

To move a data definition to a data field

1 Select a data definition in the **Outline** view (that is, select an entry under **Local Data** or **Parameter Data**).

2 Invoke the context menu and choose **Create Map Field(s)**.

The data definition is removed under **Local Data** or **Parameter Data**.

A new data field is now available on the **Layout** page. It is initially positioned in the upper left corner of the map (row 1 and column 1) from where you can move it to the required position.

Editing the Code of a Map

When you invoke the context menu on the **List**, **Data** or **Source** page of the map editor, several standard source editor commands are also available. For detailed information on these commands, see the corresponding descriptions in *Using the Source Editor*.

Changing the Properties for the Map

When you display the properties for the map itself (for example, by selecting the top-level entry in the **Outline** view), you can change the following information:

Property	Description
Appearance	
Control Variable	The name of an attribute control variable, the content of which determines the attribute characteristics of fields and texts that have the attribute definition `AD=Y`. The attribute control variable referenced in the map must be defined in the program using that map. Removing an attribute control variable from the map properties implies that the attribute control variable is removed from the map, too, unless it is associated to any other map field.
Print Mode (PM)	The default print mode for variables. This value is copied into the field definition when a new field is created. You can select the required value from the drop-down list box: None Alternative character set (C) Inverse print direction (I) No hardcopy can be made (N) None means that the standard character set is used. For further information, see the description of the session parameter `PM` in the Natural documentation for the appropriate platform.
Filler Characters	
Label Filler	Trailing filler character used for padding text constant labels if the length of the label is less than the width of the text constant.
Optional, complete	Filler character used for fields that are optional but must be completely filled if used.
Optional, partial	Filler character used for fields that are optional and can be partially filled.
Required, complete	Filler character used for fields that are required and must be completely filled.
Required, partial	Filler character used for fields that are required and can be partially filled.
General	
Auto Rule Rank	The default rank for any newly created processing rule in the map. (Later on, you can alter ranks individually).
Column Shift	Column shift (0 or 1) to be applied to the map. This feature can be used to address all 80 columns on a 80-column screen (column shift = 1, line size = 80).

Property	Description
Decimal Character	The character to be used as the decimal notation character; the default character is a period (.).
Field Sensitive	When set to true, the consistency check for a map field is made as soon as the field is filled by the user. When set to false, field checking is performed when the map is filled completely.
Help Map	When set to true, the map is marked as help text. When set to false, the map is not marked as help text.
Manual Skip	When set to true, the cursor is not moved automatically to the next field in the map at execution time, even if the current field is completely filled. When set to false, the cursor is moved automatically to the next field in the map at execution time.
Output Map	When set to true, the result of the map definition process is a `WRITE` statement and the resulting (output) map can be invoked from a program using a `WRITE USING FORM` statement. Empty lines at the end of the map are automatically deleted so that the map can be output several times on one page. When set to false, the result of the map definition process is an `INPUT` statement and the resulting map can be invoked from a program using an `INPUT` statement.
Right Justified	When set to true, numeric and alphanumeric fields copied from data definitions in other Natural objects are right-justified. When set to false, numeric and alphanumeric fields copied from data definitions in other Natural objects are not right justified.
Standard Keys	When set to true, the last two lines of the map remain empty so that function key specifications can be entered at execution time. When set to false, all lines are used for the map.
Upper Case	When set to true, input is converted to upper case during map execution. When set to false, input is not converted to upper case during map execution.
Zero Printing	When set to true, numeric fields that contain all zeroes (print one zero, right-justified) are printed. When set to false, the printing of numeric fields that contain all zeroes is suppressed.
Help	
Helproutine Name	The name of the helproutine or help map that is called at execution time when the help function is invoked for this map (global help for the map).
Help Field Default	Only shown when the name of a helproutine or help map has been defined. When set to true, the helproutine or help map specified for the map applies as the default to each individual field on the map, that is, the name of each field is passed individually to the helproutine or help map.

Property	Description
	When set to false, the name of the map is passed to the helproutine or help map.
Help Parameters	Only shown when the name of a helproutine or help map has been defined. Enter the name of the parameter(s) that will be called at execution time when the help function is invoked.
The syntax that applies to specifying helproutine names and parameters corresponds to the syntax of the session parameter HE (see the Natural documentation for the appropriate platform).	
Layout	
Layout Name	The name of the map that serves as standard layout for the current map. You can use this property to simplify the creation of many similar maps by creating one map as the basic layout map with a set of fields to be used by the other fields. In all similar maps, you specify the name of the standard layout map and you only add the fields that are specific to the new map.
Dynamic Layout	When set to true, the fields are imported dynamically into the layout at runtime. This means that you can alter your standard layout and this change will automatically be reflected in all similar maps using this layout. When set to false, the standard layout is not used dynamically.
RTL Support	
Visual Order	When set to true, the text constants of the map are assumed to be in visual order. When set to false (default), the text constants are assumed to be in logical order. See also *Bidirectional Language Support*.
Size	
Columns	The number of columns which make up the width of the map.
Rows	The number of rows which make up the height of the map.

Changing the Properties for a Text Constant

When you select a text constant, you can change the following properties:

Property	Description
Appearance	
Color Definition (CD)	Defines the color attribute for the text constant. You can select the required value from the drop-down list box: Default (None) Blue (BL) Green (GR) Neutral (NE) Pink (PI)

Property	Description
	Red (RE) Turquoise (TU) Yellow (YE) For further information, see the description of the session parameter `CD` in the Natural documentation for the appropriate platform.
Attribute Definition (AD)	
Representation	Defines how the value is displayed. You can select the required value from the drop-down list box: Default (None) Blinking (B) Italic (C) Intensified (I) Non-display (N) Underlined (U) Reverse Video (V) Dynamic (Y) For further information, see the description of the session parameter `AD` in the Natural documentation for the appropriate platform.
General	
Label	The label that is to appear on the map.
Position	
Column	The column in the map in which the text constant starts. When you move the text constant using the mouse, this value is automatically adjusted.
Row	The row in the map which contains the text constant. When you move the text constant using the mouse, this value is automatically adjusted.
Size	
Width	The length of the text constant. When you resize the text constant using the mouse, this value is automatically adjusted.

Changing the Properties for a Data Field

When you select a data field, you can change the following properties:

Property	Description
Appearance	
Color Definition (CD)	Defines the color attribute for the data field. You can select the required value from the drop-down list box: Default (None) Blue (BL) Green (GR) Neutral (NE) Pink (PI) Red (RE) Turquoise (TU) Yellow (YE) For further information, see the description of the session parameter `CD` in the Natural documentation for the appropriate platform.
Print Mode (PM)	Defines the print mode for the data field. You can select the required value from the drop-down list box: None Alternative character set (C) Inverse print direction (I) No hardcopy can be made (N) None means that the standard character set is used. For further information, see the description of the session parameter `PM` in the Natural documentation for the appropriate platform.
Array Info	
Dimensions	You can define an array of up to three dimensions. For further information, see *Defining Arrays*.
Attribute Definition (AD)	
Alignment	Defines how the content of the field is displayed. You can select the required value from the drop-down list box: None Left-justified (L) Right-justified (R) Leading zeros (Z) None means that the default alignment is used which depends on the format of a field. Left-justified is the default for alphanumeric fields. Right-justified is the default for numeric fields. Leading zeros can be defined for numeric values which are then displayed right-justified.

Property	Description
Casing	Defines whether values are to be translated to upper case or whether lower case values are to be accepted. You can select the required value from the drop-down list box: Upper case (T) Mixed case (W)
Filler Character	You can specify a character that is to be used to fill empty input fields or modifiable output fields. When you edit an input field, which is padded with blanks to its maximum length, it may appear as if the text cannot be edited. In such cases, the blanks must be explicitly deleted from the end of the field.
I/O Characteristics	Defines whether the field is an input field or output field. You can select the required value from the drop-down list box: Input Only (A) Output and Input (M) Output Only (O)
Input Length	Defines whether the value entered in the field must have same length as the field (fixed length - only relevant for input-only fields) or whether the value entered in the field can be shorter than the field (variable length). You can select the required value from the drop-down list box: Fixed (G) Variable (H)
Mandatory Input	Defines whether a value must be entered in the field (only relevant for input-only fields (`AD=A`)) or whether a value can, but need not, be entered in the field. You can select the required value from the drop-down list box: Input Mandatory (E) Input Optional (F)
Representation	Defines how the value is displayed. You can select the required value from the drop-down list box: Default (None) Blinking (B) Italic (C) Intensified (I) Non-display (N) Underlined (U) Reverse Video (V) Dynamic (Y)
For further information, see the description of the session parameter `AD` in the Natural documentation for the appropriate platform.	
General	
Alphanumeric Length (AL)	Only shown for Natural data formats A (alphanumeric) and U (Unicode).

Property	Description
	You can specify the default output length for an alphanumeric field; that is, when it is specified shorter than the field length, the field will be right-truncated. For further information, see the description of the session parameter `AL` in the Natural documentation for the appropriate platform.
Control Variable (CV)	You can enter a dynamic field attribute control variable. This is the control variable that will contain the attributes to be used for the data field. The variable must be defined with Natural data format C (for attribute control) in the program that references the map. The control variable also contains a `MODIFIED` data tag, which indicates if the field has been modified following map execution. A single control variable can be applied to several map fields, in which case the `MODIFIED` data tag will be set if any of the fields referencing the control variable have been modified. For further information, see the description of the session parameter `CV` in the Natural documentation for the appropriate platform.
Display Length (DL)	Only shown for Natural data formats A (alphanumeric) and U (Unicode). This property can be used to specify the output length for Unicode strings which can require extra space. For further information, see the description of the session parameter `DL` in the Natural documentation for the appropriate platform.
Date Format (DF)	Only shown for Natural data format D (date). This property determines the length of a date when converted to alphanumeric representation without an edit mask being specified. You can select the required option from the drop-down list box: Default (none) Short (S) Internal (I) Long (L) For further information, see the description of the session parameter `DF` in the Natural documentation for the appropriate platform.
Dynamic Attributes (DY)	You can assign attributes for dynamic attribute field display. For further information, see the description of the session parameter `DY` in the Natural documentation for the appropriate platform.
Edit Mask (EM)	You can enter the edit mask to be used for the data field. For further information, see the description of the session parameters `EM` and `EMU` in the Natural documentation for the appropriate platform. **Note:** The edit mask overrides the display length.

Property	Description
Float Length (FL)	Only shown for Natural data format F (floating point). You can specify the mantissa length of a floating point variable during input or output. For further information, see the description of the session parameter `FL` in the Natural documentation for the appropriate platform.
Format	You can select the required Natural data format from the drop-down list box: Alphanumeric (A) Binary (B) Date (D) Floating point (F) Integer (I) Logical (L) Numeric (N) Packed numeric (P) Time (T) Unicode (U) The default format is A. **Note:** Because the information required to define a field depends on the Natural data format, other properties may appear/disappear when the format changes.
Length	Only shown for Natural data formats A (alphanumeric), B (binary), F (floating point), I (integer), N (numeric), P (packed numeric) and U (Unicode). You can specify the internal length of the field which is used by a program that references this field. For valid input values, see *Format and Length of User-Defined Variables* in the *Programming Guide* in the Natural documentation for the appropriate platform.
Name	The name of the field. **Note:** The default name generated for the field when it is first created depends on the allowed identifier characters defined for the Natural server environment.
Numeric Length (NL)	Only shown for Natural data formats B (binary), I (integer), N (numeric) and P (packed numeric). You can specify the default input/output length for a numeric field used in a `DISPLAY`, `INPUT`, `PRINT` or `WRITE` statement. For further information, see the description of the session parameter `NL` in the Natural documentation for the appropriate platform.
Sign Position (SG)	Only shown for Natural data formats F (floating point), I (integer), N (numeric) and P (packed numeric).

Property	Description
	Using the drop-down list box, you can specify whether a sign position is to be allowed for the field. For further information, see the description of the session parameter `SG` in the Natural documentation for the appropriate platform.
Zero Printing (ZP)	Only shown for Natural data formats F (floating point), I (integer), N (numeric) and P (packed numeric). If set to on, zero values are printed as one zero. If set to off, zero values are suppressed. For further information, see the description of the session parameter `ZP` in the Natural documentation for the appropriate platform.
Help	
Helproutine Name	You can enter the name of a helproutine or help map to be assigned to the map field. The helproutine or help map is then invoked for the map field at execution time when a help request is made for this map field.
Help Parameters	Only shown when the name of a helproutine or help map has been defined. Enter the name of the parameter(s) that are to be passed to the helproutine or help map Removing a parameter from this text box implies that the parameter is also removed from the map, unless the parameter is a map field or is associated with any other map field as a help parameter or is a start index for an array.
The syntax that applies to specifying helproutine names and parameters corresponds to the syntax of the session parameter `HE` (see the Natural documentation for the appropriate platform). In addition to the syntax explanations provided for `HE`, the following applies when using the map editor. *`operand1`*: ■ If a variable name is specified which corresponds to the name of a map field, this field must be in the Natural data format A8. ■ If a variable name is specified for which no map field yet exists, a map parameter with that name is automatically defined in the Natural data format/length field A8. *`operand2`*: ■ If a variable name is specified for which no map field yet exists, a map parameter with that name is automatically defined in the Natural data format/length N7.	
Position	
Column	The column in the map in which the field starts. When you move the field using the mouse, this value is automatically adjusted.
Row	The row in the map which contains the field. When you move the field using the mouse, this value is automatically adjusted.

Property	Description
Size	
Width	The length of the field. When you resize the field using the mouse, this value is automatically adjusted.
The following attributes determine the width of a field in the following priority sequence: `DL > EM > { AL, NL, FL } > [SG] Length` When the width property is changed, the highest priority attribute that is used (i.e., which is non-zero or non-blank) is adjusted. Exception: The edit mask (EM) and length attributes are not automatically adjusted. **Examples:** Given field length = 30 1. Width is changed to 25 and `DL=0` => `AL` is set to 25 2. Width is changed to 25, `AL=18` and `DL` is not zero => `DL` is set to 25 3. `EM=X(25)`, `DL=0` and `AL=18` => Width property cannot be changed	

Changing the Properties for a Rule

When you select a rule, you can change the following properties:

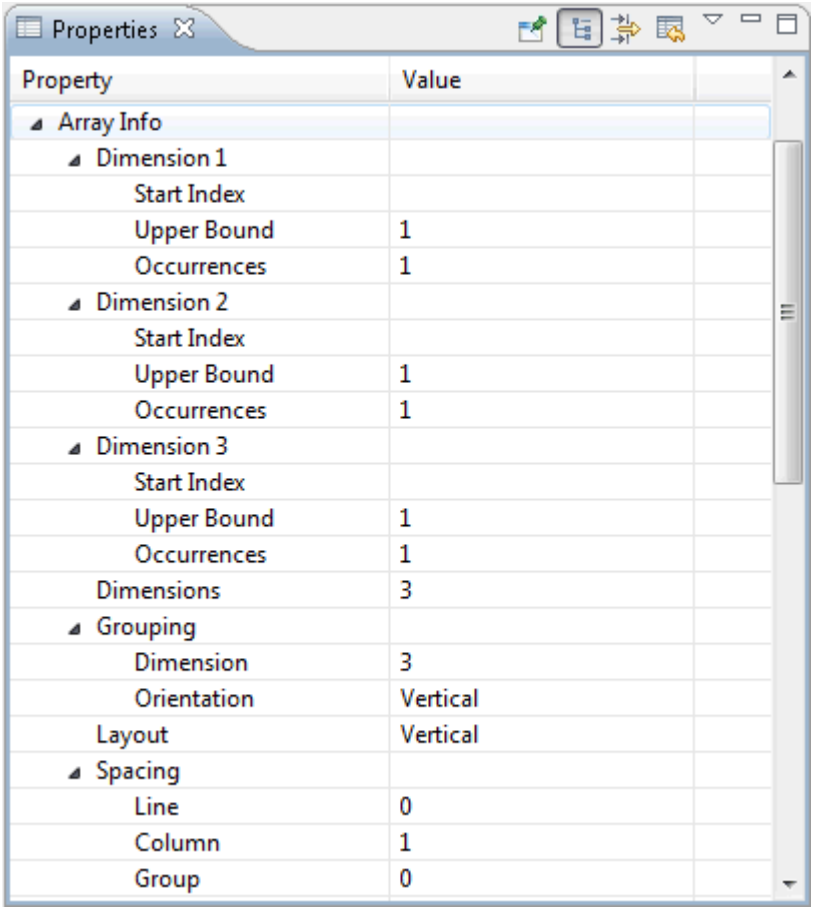
Property	Description
Type	Rule type. Can be one of the following values: ■ **Inline Rule** A rule whose code is stored within the map source. ■ **Free Rule** A named Predict rule that may be explicitly associated with a data field, or with the map itself. ■ **Automatic Rule** Predict automatic rules apply to database fields and are defined by the Predict administrator. When a data field is created from a view data definition (see *Moving Data Definitions to Data Fields*), all automatic rules for that field are implicitly linked to the map definition. All automatic rules are concatenated and treated as a single map rule. A free rule can be converted into an inline rule by selecting the type **Inline Rule** from the drop-down list box. For all other rule types, this drop-down list box is not available. **Note:** The code for Predict rules is stored and maintained on the Predict server and is not modifiable from within the map editor.

Property	Description
Rank	A numeric value (0 - 99) that determines the order in which the rule is executed relative to the others (if any). A field can have up to 100 processing rules, each of which must have a unique rank. At map execution time, the processing rules are executed in ascending order, first by rank, then by screen position of the field. For this purpose, processing rules associated with the map itself are always assumed to have the first screen position. The default rank for automatic rules is defined by the property **Auto Rule Rank** in the map properties. See also *Changing the Properties for the Map*.

Defining Arrays

Arrays are defined in the properties of a data field. You can define an array of up to three dimensions. The order in which the dimensions of the array are mapped to the map layout is determined by the values you enter.

The number of the shown array-specific properties depends on the dimension that you specify. When you specify, for example, the value 3 in the **Dimensions** property, the following properties are shown:



You can change the following array properties:

Property	Description
Dimensions	When set to 0, dimensions are not used (default) and this is the only property that is shown under **Array Info**. You can set the value to 1, 2 or 3 to use a one-, two- or three-dimensional array. In this case, additional properties are shown (see below).
Layout	Determines how the array is represented in the map layout (vertically or horizontally).
Dimension *n*	
Start Index	The starting index value for each dimension. You can enter a number or the name of a variable; the actual value is supplied in the program that invokes the map definition. Unless defined otherwise as a field in the map, the variable is assumed to be defined as Natural data format/length N7. Removing a starting index value from an array implies that the variable is removed from the map, too, unless it is a map field or it is associated to any other map field as a starting index value or help parameter.

Property	Description
Upper Bound	The upper bounds of the field. This number is the highest occurrence of the first, second and third dimension.
Occurrences	The number of occurrences that are to be displayed on the map. You can also use the mouse to resize the field vertically; the appropriate value is automatically considered in the properties. This does not apply to the third dimension of a three-dimensional array because only two ranges of occurrences can be displayed on the screen. One-dimensional arrays can be displayed as multi-line/multi-column fields. For these arrays, the second number of occurrences and the layout for the second dimension can be defined.
Spacing	
Line	Only shown for arrays that have a dimension with a vertical orientation. This is the number of blank lines to be inserted horizontally between each dimension occurrence in the array.
Column	Only shown for arrays that have a dimension with a horizontal orientation. The number of blank columns to be inserted vertically between each dimension occurrence in the array.
Group	Only shown for arrays with a grouped dimension. The number of blank lines to be inserted vertically or columns to be inserted horizontally between each occurrence of the grouped dimension.
Grouping	
Dimension	Only shown for a two- or three-dimensional array. The index of the dimension for which the array elements are grouped within the other dimension with the same orientation.
Orientation	Only shown for a three-dimensional array. The orientation in which the elements corresponding to the grouped dimension are shown.

For further information on arrays, see the *Programming Guide* in the Natural documentation for the appropriate platform.

Viewing the Status Properties

The following properties are available for the map itself and for all controls in the map:

Property	Description
Modified	Modification status (read-only). When set to true, the selected item has been modified since the last time the map was saved.
Error Status	Error and warning status (read-only). Can be one of the following values: ■ **Critical Error** The map contains a critical error. ■ **Error** The selected item or one of its children contains an error. ■ **Warning** The selected item or one of its children contains a warning. ■ **None** Neither the selected item nor any of its children contain errors or warnings. Errors and warnings are displayed in the form of label decorations in the **Outline** view. For information on the severity of an error or warning, see *Error Handling in the Map Editor*.

Saving Maps

Maps are saved using the standard Eclipse functionality.

However, when a dynamic layout is present, the following applies. When you save a map, only the so-called host map is saved. The layout map is only saved when you close the editor; in this case, a dialog appears asking whether you want to save the layout map.

If you want to save the layout map at an earlier point in time, proceed as described below.

To save the layout map (dynamic layout only)

1 In the **Outline View**, select the layout map.

2 Invoke the context menu for the map and choose **Save**.

Or:

Press CTRL+S.

Or:

From the **File** menu, choose **Save**.

Stowing Maps

When you have opened a map from the **Natural Server** view, you can stow the map directly from within the map editor.

To stow the map (Natural Server view only)

1 Make sure that the **Layout** page is shown.

2 Invoke the context menu for the map and choose **Stow**.

Or:

Press CTRL+T.

Validating Maps

When you save a map, the following consistency checks are automatically performed in order to determine the validity of the map:

Consistency Check	Description
Critical Errors	Detects whether the map contains one or more critical errors, indicating that the map could not be loaded successfully. If so, an error message is displayed. This condition is usually caused by syntax errors within the `DEFINE DATA` or `INPUT`/`WRITE` statements, which can be inspected by opening the map with the source editor.
Invalid Data Definitions	Detects whether one or more of the data sections in the map contain errors. If so, an error message is displayed. This condition is usually caused by one or more of the entered data definitions having a syntax error.
Empty Map	Checks whether the map contains no fields. If so, a warning message is displayed. Note that empty maps are not syntactically valid in Natural, because the underlying `INPUT` or `WRITE` statements require at least one operand to be present. In addition, maps containing one or more processing rules but no fields cannot be opened correctly due to this restriction. To minimize the risk of this situation arising, the saving of such maps is not allowed.
Overlapping Fields	Detects whether the map contains one or more overlapping fields (thus making it syntactically invalid). If so, an error message is displayed. In addition, the overlapping fields are automatically selected in order to assist you in locating the source of the error.

If you want to validate the map at an earlier point in time, proceed as described below.

To validate the map

1 Make sure that the **Layout** page is shown.

2 Invoke the context menu for the map and choose **Validate Map**.

Or:

Press CTRL+ALT+V.

In the case of an error, a message dialog appears providing information about the error.

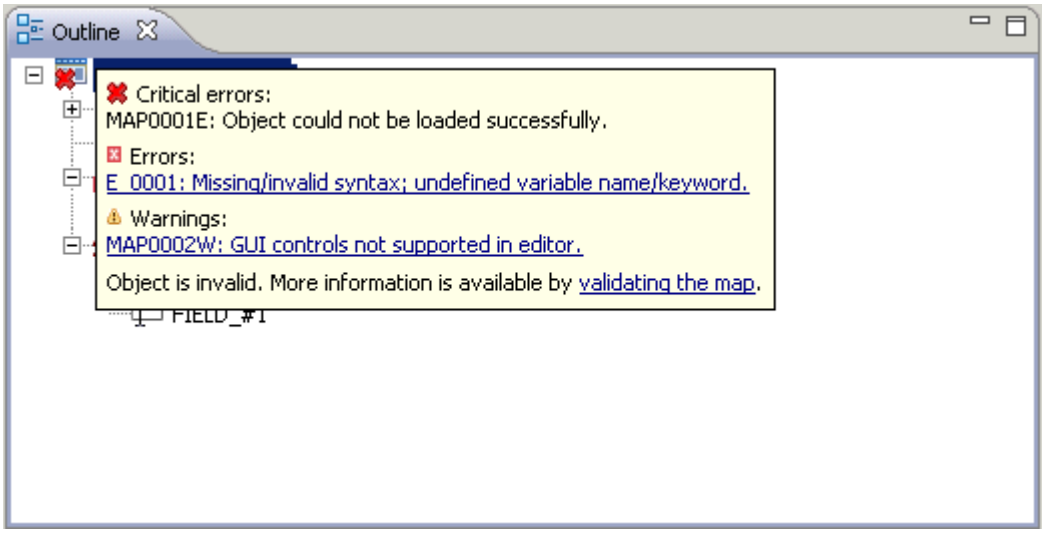
By default, a message dialog is shown even if the validation was successful. This behavior can be changed in the Natural preferences. See *Map Editor* in *Setting the Preferences*.

Error Handling in the Map Editor

In the **Outline** view, errors and warnings are displayed in the form of label decorations (see also *Viewing the Status Properties*). The image used depends on the severity as follows:

Image	Description
	Critical error. This decoration is displayed for the top-level node if one or more components of the map (such as field or data definitions) could not be loaded correctly, typically due to a syntax error. Critical errors cannot be corrected within the current editor session. The exact location of the error(s) can be determined by opening the map using the source editor.
	Error. This decoration is displayed to represent errors that are fixable in the current editor session.
	Warning. This decoration alerts the user to conflicts that are not clearly serious enough to be regarded as errors.

When you move the mouse over an item in the **Outline** view which causes an error or warning, a tooltip appears. This tooltip may contain one or more links. You can navigate to an error in a map by clicking the corresponding link. In specific cases, a link is also available which allows you to validate the map and thus display information about the error in a message box (see also *Validating Maps*).



You can also navigate to an error in a map using the **Problems** view (see the description under *Problems in Your Natural Sources*). The preferences for the **source editor**, especially the **Show error list** option, also apply to the **List**, **Data** and **Source** pages of the map editor.

When you navigate to an error, the map editor responds to such a request by trying to find the first map component (data definition, text constant, data field, array element or processing rule) that contains (or partially contains) the line represented by the chosen error marker, and (if such a component is found) selects it on the **List** page of the map editor, or in the case of a processing rule, selects it on the **Source** page. If no component is found, or the navigation fails for any other reason, a message dialog is displayed.

It is important to recognize that the errors shown in the **Outline** view tooltip relate to the current state of the map within the editor, whereas the resource markers shown in the **Problems** view relate to the map as stored in the workspace.

Information relating to the error or warning may also be written to the **Error Log** view.

Notes:

1. The information provided in the section *Error Handling in the Source Editor* also applies to the **List**, **Data** and **Source** pages of the map editor.
2. The label decorations for errors and warning in the map editor are controlled by the preferences under **General > Appearance > Label Decorations**. If you do not want to have these label decorations, just go to the above mentioned preference page and deselect **Natural Map Editor Errors and Warnings**.

18 Using the DDM Editor

- Creating a DDM 262
- About the DDM Editor 265
- Associated Views 266
- Managing the Fields of the DDM 268
- Field Attributes 269

Creating a DDM

For Natural to be able to access a database file, a logical definition of the physical database file is required. Such a logical file definition is called a data definition module (DDM).

For more information on DDMs, see *Accessing Data in a Database* in the *Programming Guide* in the Natural documentation for the appropriate platform.

You can create DDMs directly from the field definitions in a database. This can be an Adabas, SQL or XML database, or any other database (such as VSAM).

Note: DDMs from XML databases cannot be created for mainframe servers.

To create a DDM

1 In the **Navigator** view or in the **Natural Navigator** view, select the library in which you want to store the new DDM.

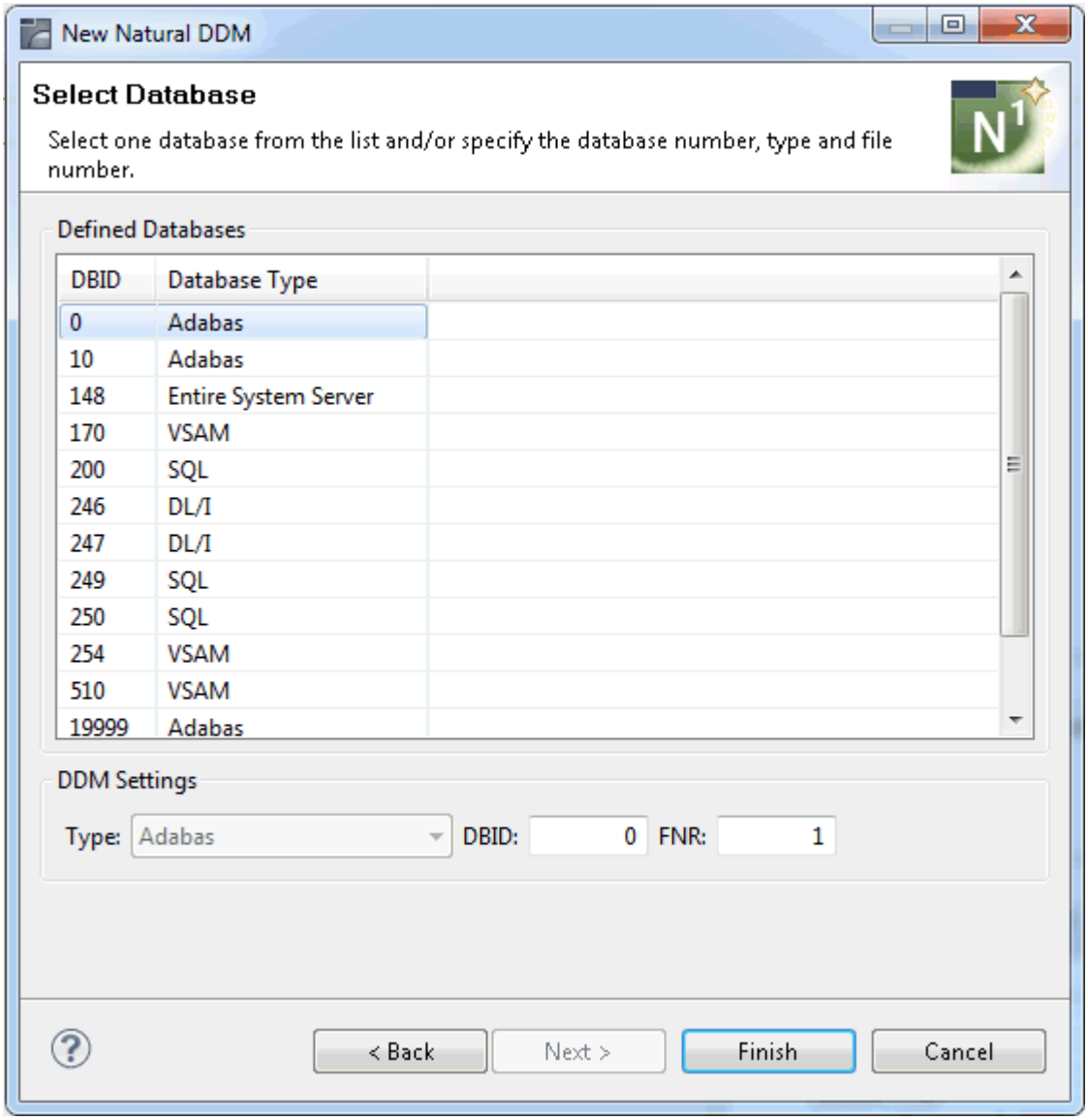
2 From the **File** menu, choose **New > DDM**.

The wizard for creating a new DDM appears (see also ***Creating Natural Objects***). Several pages are provided in the wizard, in addition to the page on which you enter the DDM name.

3 Enter a name for the new DDM.

4 Choose the **Next** button.

NaturalONE now checks the available databases on the Natural server which is defined for the current project. When the corresponding server is currently mapped in the **Natural Server** view, all defined databases on this server are shown on the next page. When the corresponding server is not mapped or is currently not active, the table of defined databases remains empty.



5 Select the database from which you want to create the DDM from the table of defined databases.

Or:

Enter the corresponding DDM settings at the bottom of the page (this is helpful when the table of databases is empty):

1. Select the required database type from the drop-down list box.
2. As the database ID (DBID), enter a numeric value in the range from 0 to 65535 (except 255).

   If you enter a 0 (zero) as the database ID, the database ID specified with the Natural profile parameter `UDB` (see the Natural documentation for the appropriate platform) of the Natural parameter file `NATPARM` is used.

3. A file number (FNR) is only required for Adabas databases and any other database types except SQL and XML. Enter a numeric value in the range from 1 to 5000.

6 For SQL and XML databases, choose the **Next** button.

The resulting page depends on the type of database that you have selected:

- **SQL**
  A page is shown on which you have to specify the selection criteria for the SQL table retrieval:
  - To list all SQL tables for selection, use the asterisks (*) in the text boxes **Owner** and **Name**. The asterisks (*) are entered by default.
  - To list a particular range of SQL tables, use asterisk (*) notation (for example, AB* selects all SQL tables with names that start with AB).

    Note: When an SQL database is accessed via the ODBC interface, a table catalog is not provided. This means that you can only specify a table name but not a table owner.

  Choose the **Next** button.

  Depending on your SQL database settings, a database logon dialog appears if you access this SQL database for the first time in this session. Enter the user ID and the password for the database and choose the **OK** button.

  You can now select the required SQL table from the next page.
- **XML**
  A page is shown on which you have to select a Tamino doctype.

  A database logon dialog appears, if you are accessing this Tamino database for the first time in this session. Enter the user ID and password for the database and choose the **OK** button.

Note: For Adabas and other database types, additional pages are not shown.

7 Choose the **Finish** button.

If the specified database and the file are available, the DDM editor is invoked and the fields contained in that database file are read into the DDM editor.

If the specified database is not active or cannot be accessed or if the file does not exist, a corresponding error message is issued. Nevertheless, an empty DDM editor window appears (containing a single dummy field) in which you can enter new field attribute definitions and

save the DDM source. However, in this case you cannot check any definitions against the database file description.

About the DDM Editor

When you create or open a DDM, the DDM editor is invoked. Example:

EMPLOYEE.NSD

Type	Level	Short...	Name	Format	Length	Sup...	Descriptor	Header
	1	AA	PERSONNEL-ID	A	8		D	PERSONNEL/ID
*			CNNNNNNN					
*			C=COUNTRY					
G	1	AB	FULL-NAME					
	2	AC	FIRST-NAME	A	20	N		
	2	AD	MIDDLE-I	A	1	N		
	2	AE	NAME	A	20		D	
	1	AD	MIDDLE-NAME	A	20	N		
	1	AF	MAR-STAT	A	1	F		MARITAL/ST...
*			M=MARRIED					
*			S=SINGLE					
*			D=DIVORCED					
*			W=WIDOWED					
	1	AG	SEX	A	1	F		S/E/X
	1	AH	BIRTH	N	6.0		D	DATE/OF/BIR...
	1	AH	N@BIRTH	I	2		D	
G	1	A1	FULL-ADDRESS					
M	2	AI	ADDRESS-LINE	A	20	N		ADDRESS
	2	AJ	CITY	A	20	N	D	

When the DDM editor is active, a corresponding toolbar is available. See *Managing the Fields of the DDM*.

The behavior of the DDM editor can be influenced by changing the Natural preferences. See *DDM Editor* in *Setting the Preferences*.

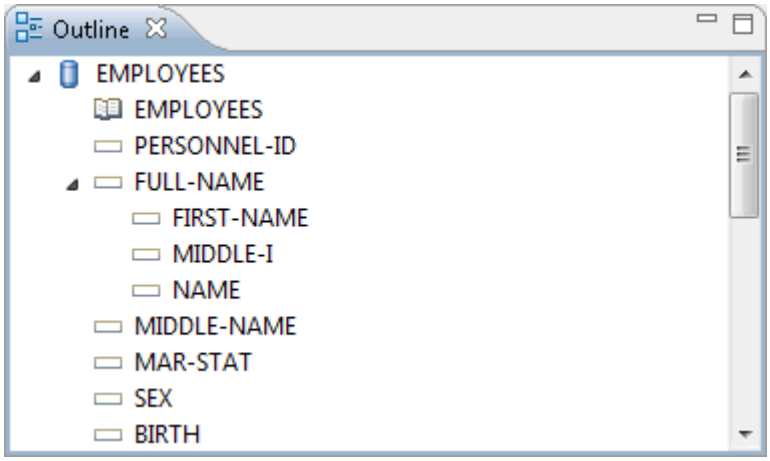
Associated Views

The DDM editor uses the following views of the NaturalONE perspective:

- Outline View
- Dependencies View
- Properties View

Outline View

This view shows the hierarchical structure of the DDM and the DDM fields in a tree. The top-level node contains the long name of the DDM, the first child node represents the DDM header and the other child nodes represent the DDM fields.

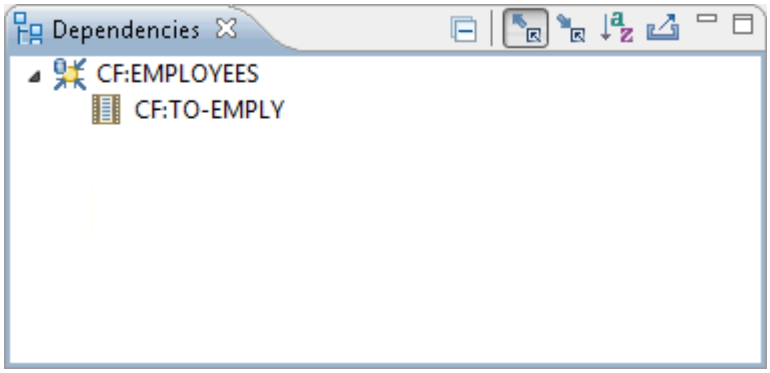


You can invoke a context menu to insert, cut, copy, paste or delete fields. See also *Managing the Fields of the DDM*.

You can navigate to a particular item in the DDM editor by selecting it in the **Outline** view, and vice versa. In the **Properties** view (see below), the selection is changed accordingly.

Dependencies View

This view shows the dependencies between the DDM which is currently shown in the active editor window and other objects. For example, when the passive cross-references are currently displayed, you can see all Natural objects that reference the DDM being edited.

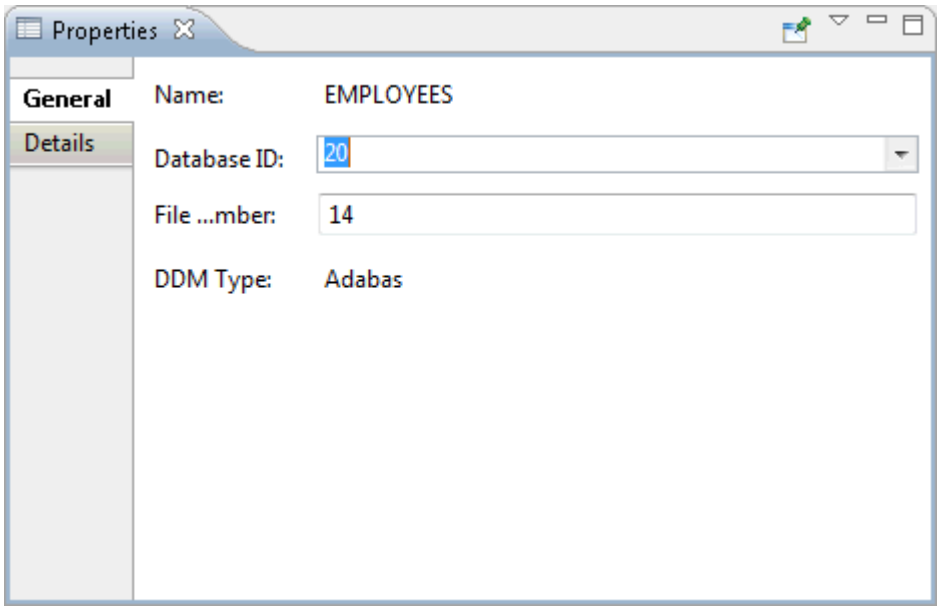


For further information on this view, see *Dependencies View* in the description of the source editor.

Properties View

This view shows information on the DDM, the DDM header or the field that is currently selected in the DDM editor or in the **Outline** view.

In general, the attribute values shown in the **Properties** view are not modifiable. As an exception, information that cannot be edited in the table view can be supplied here. This includes for all database types the database ID, the file number, the default sequence and, in particular, definitions required for VSAM databases.



The **General** page shows common information. When a DDM or DDM field is selected, it shows the name of the DDM or DDM field. When the DDM header is selected, it shows the name of the DDM header, database ID, file number and database type. Database ID and file number can be changed here, the name of the DDM and the database type are not modifiable.

If available, the **Details** page shows additional, more specific information that depends on the currently selected object:

- For Adabas DDMs, it shows information on coupled files. For Adabas DDM headers, it shows the default sequence. For Adabas DDM fields, it shows the parent fields of the selected super-descriptor.
- For SQL DDM headers, it shows the default sequence.
- For Tamino DDM headers, it shows information on the XML schema definition, doctype, collection and namespaces. For Tamino DDM fields, it shows field-specific information taken from the XML schema such as XML tag and XML path.
- For VSAM DDMs, it shows extended information on the VSAM header and the VSAM field definitions.

Managing the Fields of the DDM

Using the commands from the **Edit** menu, you can cut, copy, paste and delete fields in the DDM editor, and you can undo and redo your last change. Some of these commands are also available when you invoke the context menu for a row in the DDM editor, and they are also available in a toolbar. The DDM editor also offers a toolbar of its own which you can use to insert fields in the DDM editor.

In addition, it is possible to invoke a context menu in the **Outline** view. Using the commands in this context menu, you can also insert, cut, copy, paste and delete fields.

The commands **Copy**/**Cut** and **Paste** are used to copy or move one or more fields within the current DDM source or between different DDM sources. The copied or cut fields are placed on the clipboard and can be pasted into the current or another DDM source; the fields are inserted either before or after a selected field, depending on the insert location that has been defined in the **preferences** for the DDM editor.

In the DDM editor, you can move from one field to another using the TAB key.

To select a row in the DDM editor (for example, if you want to copy or delete a row), you can either double-click the row or press ESC so that the entire row is selected.

To add or modify a specific value, you simply click the corresponding cell in the DDM editor. Or, when a row is currently selected, you can also press F2 to edit the cell which was active before the row was selected (when no cell was previously active, the first cell is activated for editing). For some columns, simple text boxes are provided in which you have to enter information. For other columns, drop-down list boxes are provided from which you can select a value.

If you want to rearrange the columns in the DDM editor, you simply drag the column header to the desired position.

Field Attributes

Each row in the DDM editor represents a DDM field. The following field attributes can be viewed or defined in the different columns of the DDM editor.

All types of DDMs have the following columns in common:

Column	Description
Type	Type of field. *blank*: Elementary field. G: Group. M: Multiple-value field. P: Periodic group. *: Comment line.
Level	Level number assigned to the field.
Name	Name of the field.
Format	Natural data format of an elementary field, such as A (alphanumeric).
Length	Standard length of an elementary field.
Descriptor	Descriptor type. *blank*: No descriptor. D: Elementary descriptor. H: Hyperdescriptor. N: Non-Descriptor. P: Phonetic descriptor. S: Subdescriptor or superdescriptor.
Header	Header to be produced for each field specified in a `DISPLAY` statement.
Edit Mask	Edit mask to be used.
Remarks	Comment which applies to a field and/or the DDM.

Adabas DDMs and SQL DDMs additionally show the following columns:

Column	Description
Short Name	Two-character short name of the corresponding field in the database file.
Suppression	Null-value suppression option. *blank*: Standard Adabas suppression. F: Adabas fixed storage option. N: Adabas null-value suppression option. M: SQL null-value option.
Format Option	Only shown for database type ADA2. Adabas format option for alphanumeric fields.

Column	Description
SQL Type	Only shown for SQL DDMs. Information generated from the data types BLOB (binary large object) or CLOB (character large object) if contained in an Oracle database.

For detailed information on the field attributes that can be defined in the different columns, see the description of the DDM editor in the Natural documentation for the appropriate platform.

V Using the Debugger

This part describes how to debug the different types of applications that can be created with NaturalONE. It covers the following topics

Debugging Natural Applications

Using a Debug Attach Server

If you want to debug traditional Natural applications, you proceed as described in *Debugging Natural Applications*.

If you want to debug Natural RPC applications or external Natural applications, a debug attach server is required. In this case, you have to proceed as described in *Using a Debug Attach Server*.

Note: If you want to debug workplace applications (that is, Natural for Ajax pages of type MFPAGE), a debug attach server is also required. For further information, see *Executing and Debugging Workplace Applications* in the Natural for Ajax documentation.

19 Debugging Natural Applications

- General Information 274
- Using Symbol Tables 274
- Starting the Debugger 275
- Commands in the Debug Perspective 276
- Using the Debug Perspective 276
- Specifying the Breakpoint and Watchpoint Properties 283
- Going to the Next Statement 287

General Information

NaturalONE saves the Natural sources in the Eclipse workspace. The compiled objects remain in the Natural environment (local Natural runtime or Natural server). Therefore, debugging is initiated in local mode, that is, from a project which is stored in the Eclipse workspace. This means that the source is available in the workspace and the corresponding generated program is available in the Natural environment. Since there is no virtual machine for Natural in the Eclipse environment, the Natural environment is required for execution. This approach has the advantage that you will be able to test your application in the appropriate environment; that is, in an environment in which your application will be executed in production.

The library in which a generated program is debugged is determined by the mode you have defined for the project (either shared mode or private mode). See ***Steplibs*** in *Changing the Project Properties* for further information.

Important: Web I/O must be enabled on the Natural server. Otherwise, the output of a debugged program cannot be displayed.

The standard Eclipse functionality for debugging is used; see the Eclipse online help. The information which is specific to NaturalONE is provided in the topics below.

Using Symbol Tables

This information applies only when you are using a Natural server in a UNIX, OpenVMS or Windows environment.

The debugger needs information from a symbol table in order to display the names of variables. So that symbol tables can be generated in Natural, the Natural parameter `GPGEN` must be set to "DEBUGGER=ON". This parameter can be set in one of the following ways:

- in the default parameter file `NATPARM` on the Natural server (using the Configuration Utility), or
- as a session parameter when **mapping** a Natural server with NaturalONE.

When you catalog or stow an object and `GPGEN` is set to "DEBUGGER=ON", a symbol table is generated as part of the generated program. This table contains the information relevant to the variables active for this object.

Variables cannot be accessed when `GPGEN` is not set to "DEBUGGER=ON", although it is still possible to debug the object.

Note: When you are using a Natural server in a mainframe environment, a symbol table is always generated as part of the generated program. It is not required to set a parameter for this purpose.

Starting the Debugger

For details concerning the **Default Launch** settings, see *Launching Natural Applications* in section *Working with Natural Projects in Local Mode*.

Tip: Since NaturalONE always debugs the generated program in the Natural environment (and not the source in the local workspace), it is recommended that you enable the **Check time stamp on server** option in the Natural preferences (see *Natural > Builder* in *Setting the Preferences*). Thus, you can avoid a situation where the source in your local workspace differs from the corresponding source on the server, which may lead to unpredictable results. When a time stamp conflict is found while debugging (that is, when a source in the local workspace has a time stamp which differs from the time stamp of the corresponding source on the server), a dialog box appears, asking whether you want to update the source in the local workspace with the source from the server. See also *Checking the Time Stamps in the Natural Environment*.

To start the debugger

1 Select the program that you want to debug.

2 Invoke the context menu and choose **Debug As > Natural Application**.

Or:

Press ALT+SHIFT+D, N.

Or:

After the Launch was activated, the Debug perspective is opened. In the **Console** view, you can see that the connection to the Natural environment is established and that the debugger is started.

In the editor window, the debugger waits at the first executable source code line. You can now use the standard Eclipse functionality to debug the Natural application. If a source cannot be found in the workspace, a dialog box appears asking whether you want to download the object from the Natural environment. If you agree, NaturalONE tries to download the object.

The output of the debugged object is either shown in the internal browser or in an external browser, depending on the settings in the Natural **preferences** or in the launch configuration that you have created (see below).

Note: If you receive an error message after choosing the **Debug** command which tells you that the connection to the debug attach server has failed, this indicates that the use of the debug attach server has been enabled in the Natural preferences (see *Debug Attach Settings* in *Setting the Preferences*). For debugging traditional Natural applications, however, a debug attach server is not required. Therefore, you can disable the debug attach server in the Natural preferences. Then, the error message will no longer appear and you will be able to debug your program.

Commands in the Debug Perspective

When you are working in the Debug perspective, you can use several commands from the **Run** menu, for example:

- **Resume**
- **Terminate**
- **Step Into**
- **Step Over**
- **Step Return**
- **Watch**
- **Toggle Breakpoint** (automatically uses line breakpoints)
- **Toggle Line Breakpoint**
- **Toggle Watchpoint**
- **Skip All Breakpoints**
- **Remove All Breakpoints**

See the Eclipse online help for detailed information on these commands.

Using the Debug Perspective

NaturalONE uses the following elements in the Debug perspective:

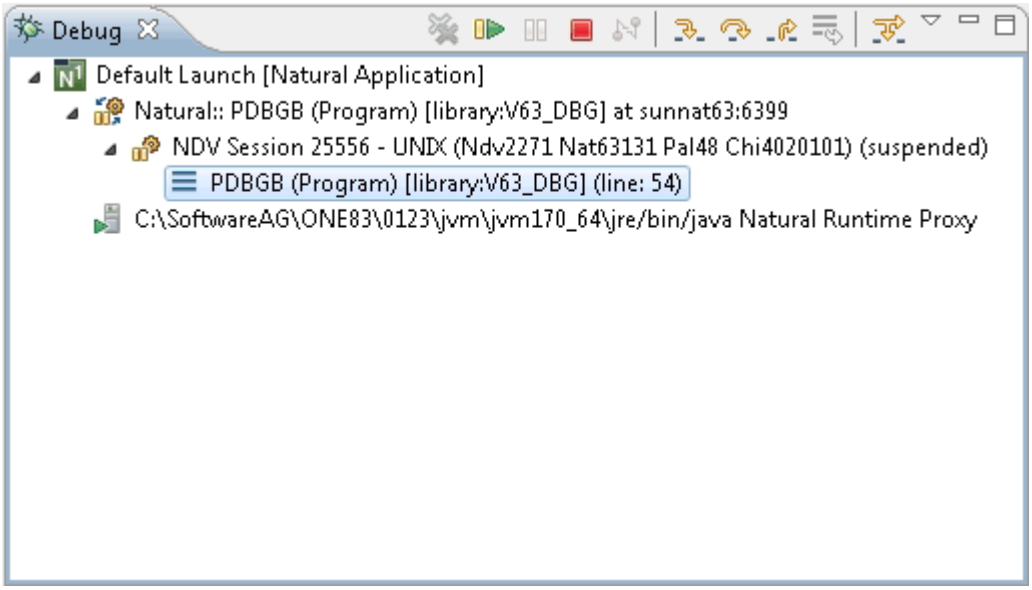
- Debug View
- Variables View
- Breakpoints View
- Natural Stack View
- Expressions View
- Editor Area

- Outline View

Note: The Debug perspective can also be opened by choosing **Open Perspective > Other** from the **Window** menu.

Debug View

The **Debug** view displays the stack frame.



In Natural terminology, this is the “call stack” which lists the objects which have been called during the current debugging session in hierarchical order.

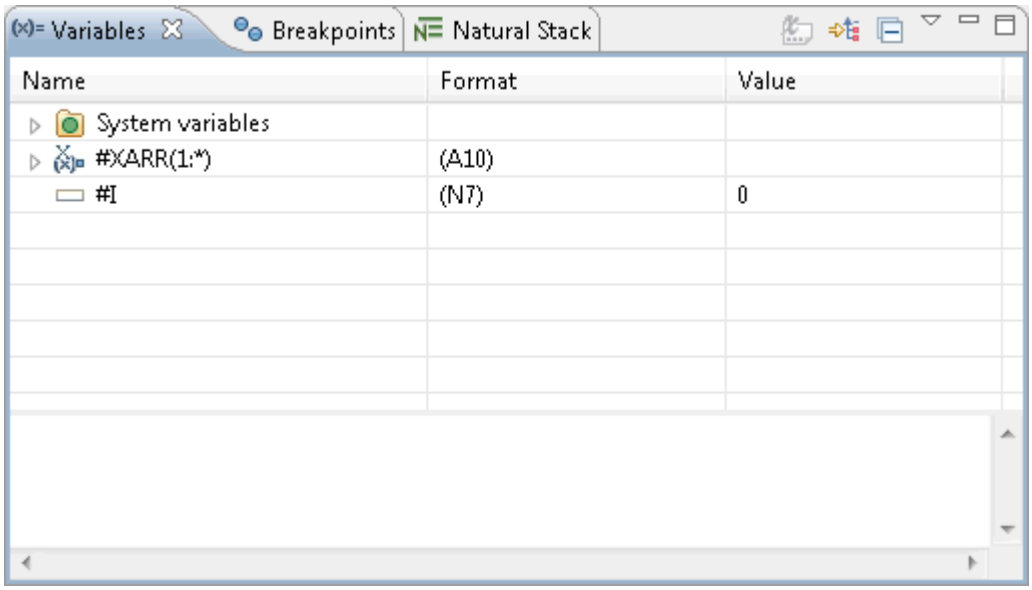
Exception: when debugging on a mainframe, only the top-level element is shown in the stack frame.

You can debug several Natural applications or even Java applications in parallel. You determine the application which gets the debugger attention by selecting the appropriate launch configuration in the **Debug** view.

See the Eclipse online help for further information on this view.

Variables View

The **Variables** view shows information about the variables in the currently selected stack frame.



Separate nodes are provided for the different types of variables (global variables, system variables, application-independent variables (AIVs) and context variables). The exceptions are the local variables; they are shown at the top level and are not grouped into a node.

You can change the value of a variable directly in the **Value** column. You can also do this using the context menu.

Using the **Watch** command in the context menu, you can add a watch expression for the selected variable which is then added to the **Expressions** view.

Enhancement of Variable Display

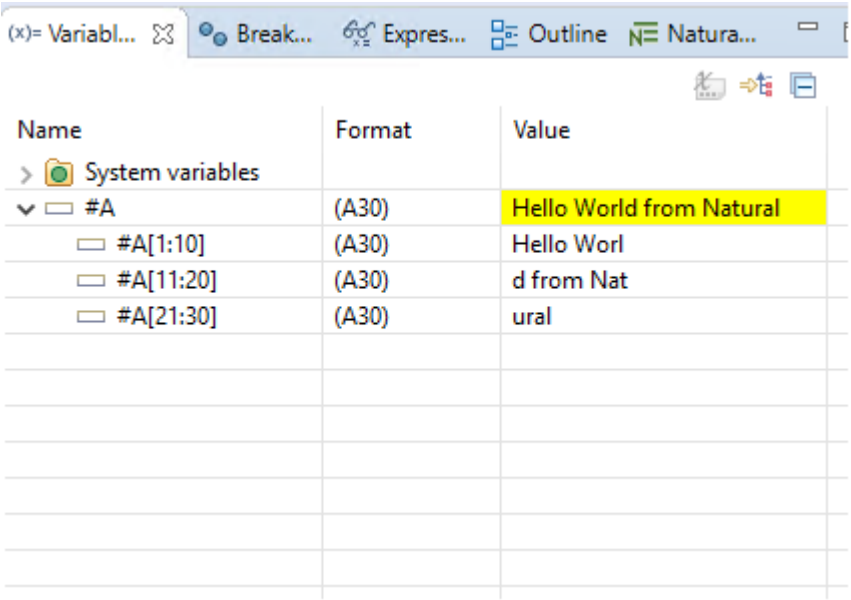
The variable value display within the debugger for large and dynamic variables of type A, U or B in some cases is limited and a Natural developer may not see the value in its complete length.

In order to provide the display of the value in its complete length, you can use the variable value chunks. With this enhancement, variables of type A, U or B are expandable in the **Variables** and **Expressions** view if their size is greater than the chunk size.

Value chunks are displayed as children of the variable node. The length of a single chunk is configurable in the Preferences as described under ***Display Options*** in *Setting the Preferences*.

You can handle the variable chunk nodes in the same way as other variable nodes. Its value can be modified and the variable chunk can be added to the **Expressions** view with the context menu command **Watch**.

There will be as many variable chunk nodes as are needed to display the whole variable value as displayed in the following screen.

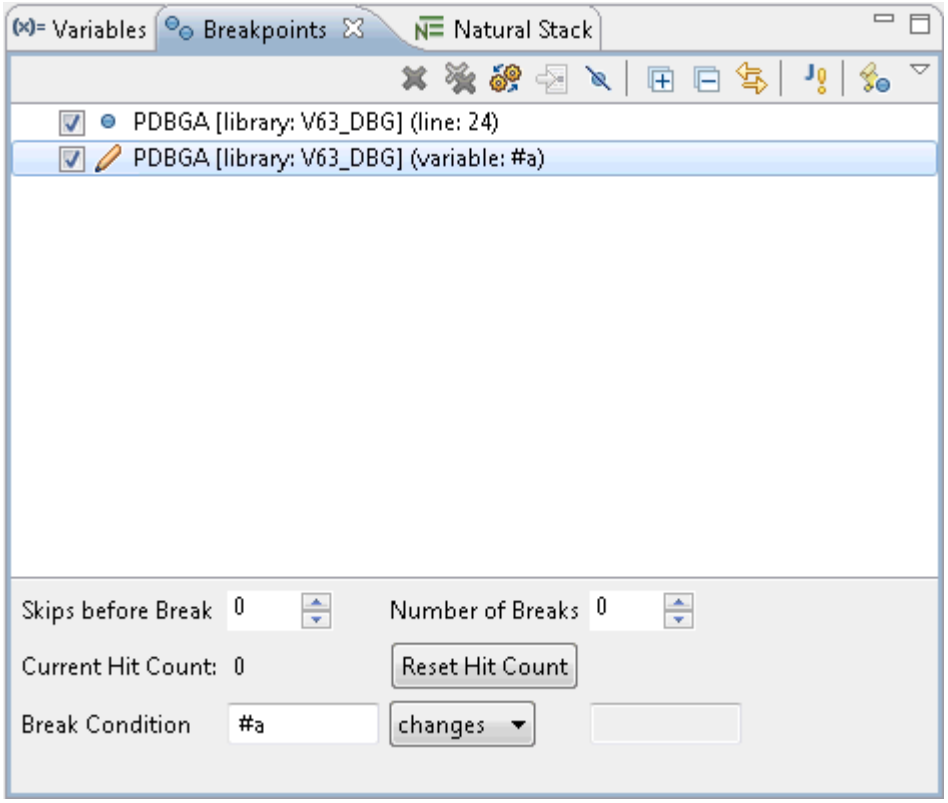


This means that with this enhancement you can watch the complete variable value and modify any part of the value.

See the Eclipse online help for further information on this view.

Breakpoints View

The **Breakpoints** view lists all line breakpoints and watchpoints you have set in your workbench projects.

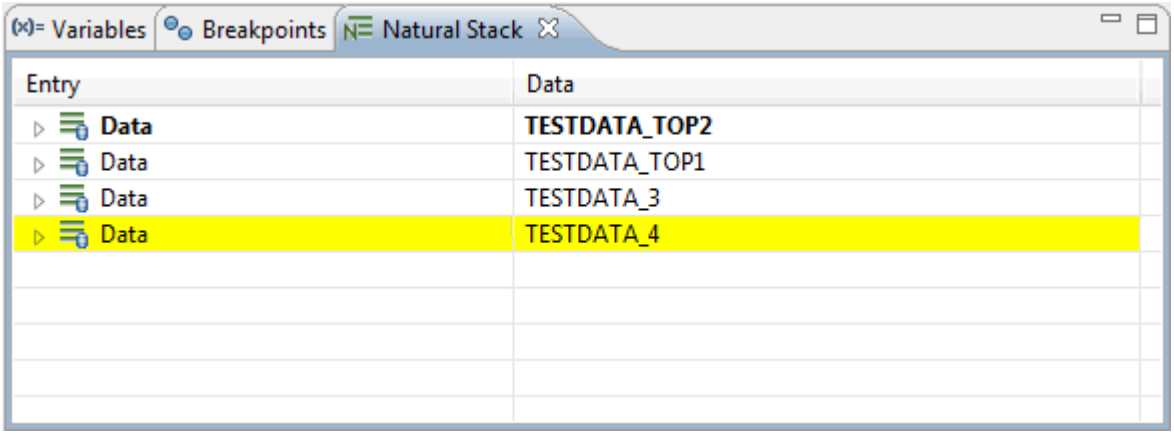


You can use the context menu, for example, to disable a breakpoint or to specify the properties for your breakpoints and watchpoints. You can also specify the properties for your breakpoints and watchpoints using the options at the bottom of the view (for detailed information on these options, see *Specifying the Breakpoint and Watchpoint Properties*).

See the Eclipse online help for further information on this view.

Natural Stack View

The **Natural Stack** view is a Natural-specific view which is shown by default. It shows the current contents of the Natural stack. The top entry in the stack is shown in bold. Modifications between two debugger suspends are marked with a yellow background color.

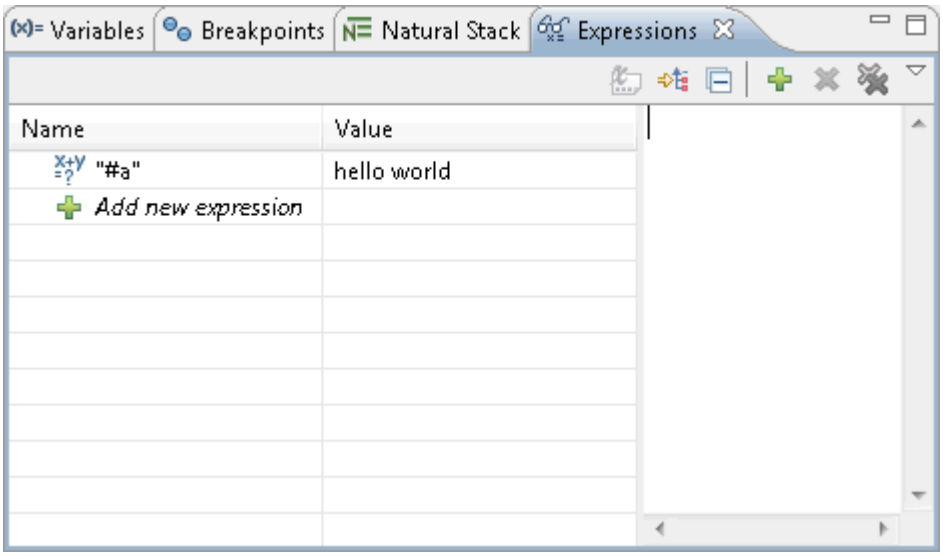


Note: The **Natural Stack** view is part of the Debug perspective. If it is currently not shown, you can redisplay it with **Window > Show View > Other > Debug > Natural Stack**.

Expressions View

The **Expressions** view is not shown by default. However, is automatically opened when a watch expression is added to this view.

The **Expressions** view shows the current values for all watch expressions that you have defined. The values change while you step through the code. In Natural terminology, these are the "watchvariables". The benefit of this view is that only the contents of those variables are shown that you want to observe. This is different from the **Variables** view in which you see all used variables and the Natural system variables at the same time.



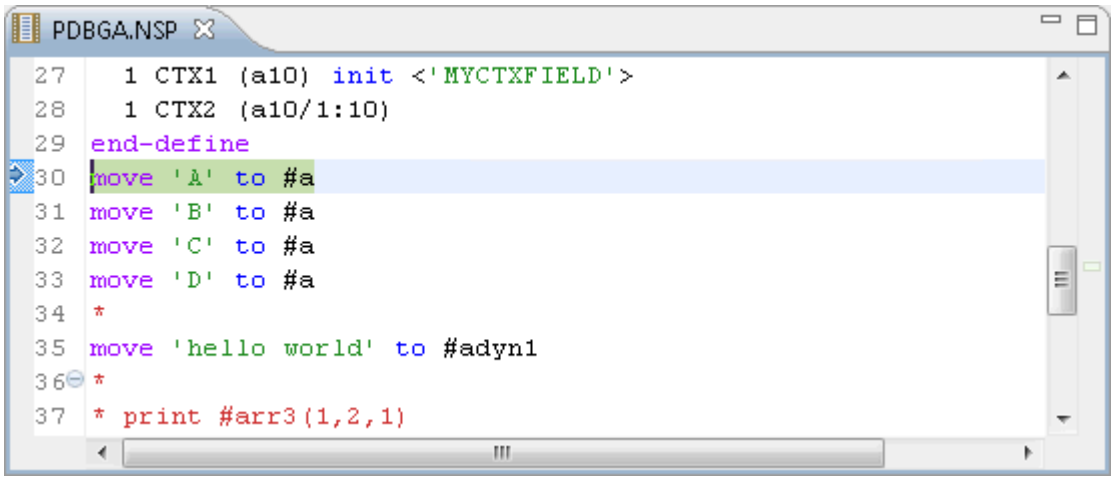
You can use the context menu, for example, to remove one or more watch expressions.

For watch expressions which can be expanded, it is possible to edit the values of the contained variables. For example, when you expand an array, you can change the values of the occurrences.

See the Eclipse online help for further information on this view.

Editor Area

The current trace position is indicated by an arrow in the marker bar of the editor window. When the debugger is started, the trace position is shown at the first executable source code line.



You can use Eclipse features such as breakpoints, stepping, or expression evaluation to debug your program. See the Eclipse online help for further information.

To add a breakpoint, you can simply double-click on the marker bar, directly next to the line for which you want to add the breakpoint. To remove this breakpoint, you simply double-click it once more in the marker bar.

When you add breakpoints while the Natural runtime on the server is suspended, the new breakpoints are considered when the Natural runtime is resumed.

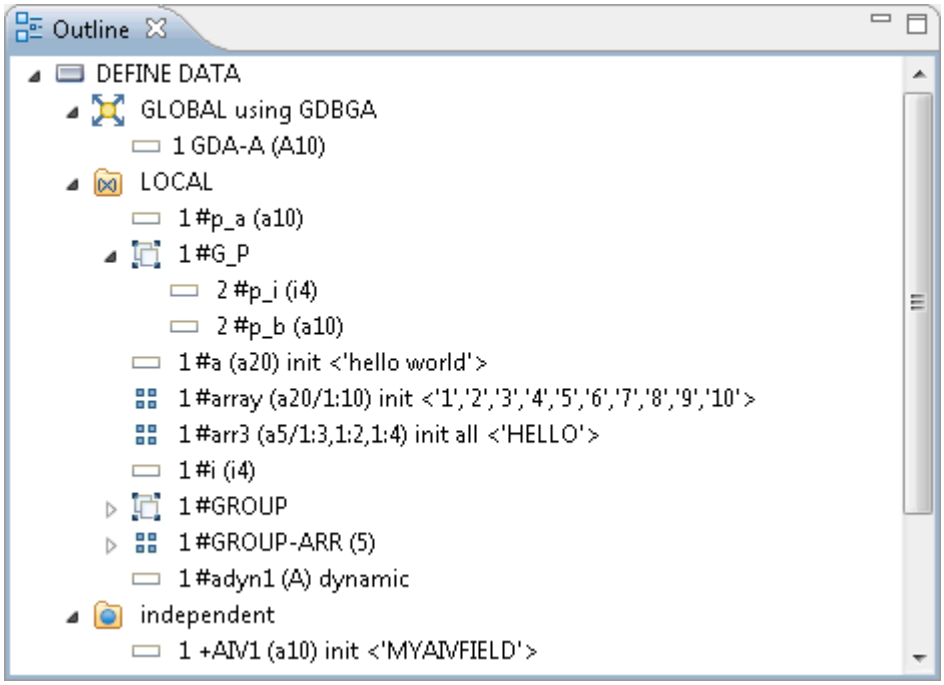
When the debugger is in suspend mode, it is possible to display the current format/length and value of a Natural variable by positioning the mouse over the variable (in Eclipse terminology, you “hover” over the variable name in the editor). The current content is then shown in a hover, for example:

```
29  end-define
30  move 'A' to #a
31  move 'B' to
32  move 'C' to
33  move 'D' to #a
34  *
```

#A(A20) = hello world
Press 'Tab' from proposal table or click for focus

Outline View

The **Outline** view provides the same information for a source as in the NaturalONE perspective. It visualizes the hierarchical structure of the program.



Specifying the Breakpoint and Watchpoint Properties

You can modify a breakpoint or watchpoint by changing its properties.

Every breakpoint or watchpoint has a hit count which increases every time the debug entry is passed. With NaturalONE, the number of executions of a debug entry can be restricted in the following ways:

- A number of skips can be specified before the breakpoint or watchpoint is executed. The debug entry is then ignored until the event count is higher than the number of skips specified.
- A maximum number of executions can be specified, so that the breakpoint or watchpoint is ignored as soon as the event count exceeds the specified number of executions.

To modify a breakpoint or watchpoint

1 In the **Breakpoints** view, select the breakpoint or watchpoint.

2 Invoke the context menu and choose **Breakpoint Properties**.

   Or:

   Press CTRL+ENTER.

   The Eclipse dialog boxes which appear for the properties of the breakpoints and watchpoints have been adapted to Natural. The content of the resulting dialog box depends on the selected item.

3 Modify the breakpoint or watchpoint as described in the topics below.

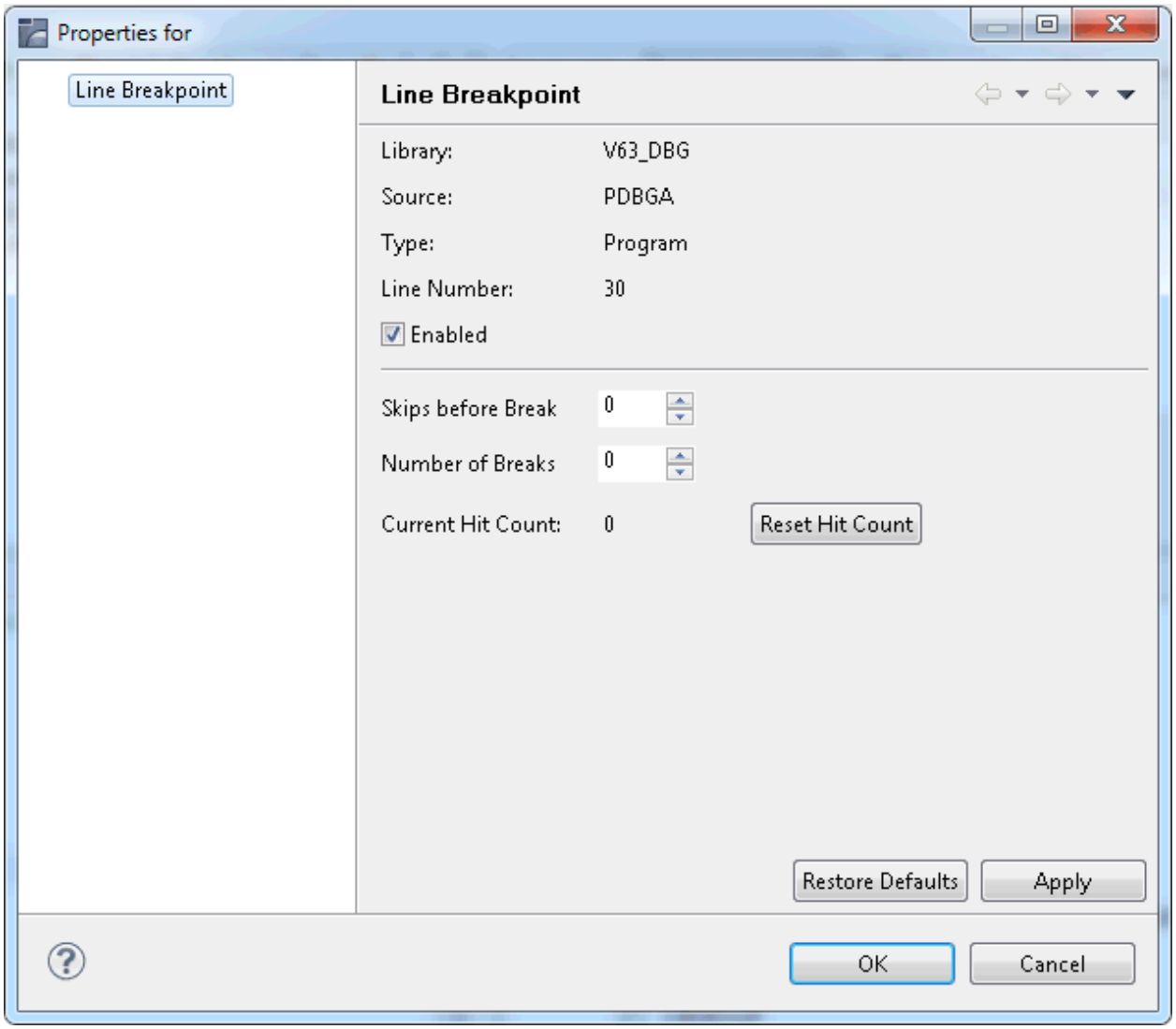
4 Choose the **OK** button.

The following topics are covered below:

- Breakpoint Properties
- Watchpoint Properties

Breakpoint Properties

The following dialog box appears for a breakpoint.



You can set the following Natural-specific options:

Skips before break

The number of skips before execution of the breakpoint if it is not to be executed until the program has run a certain number of times. The default is 0.

Number of breaks

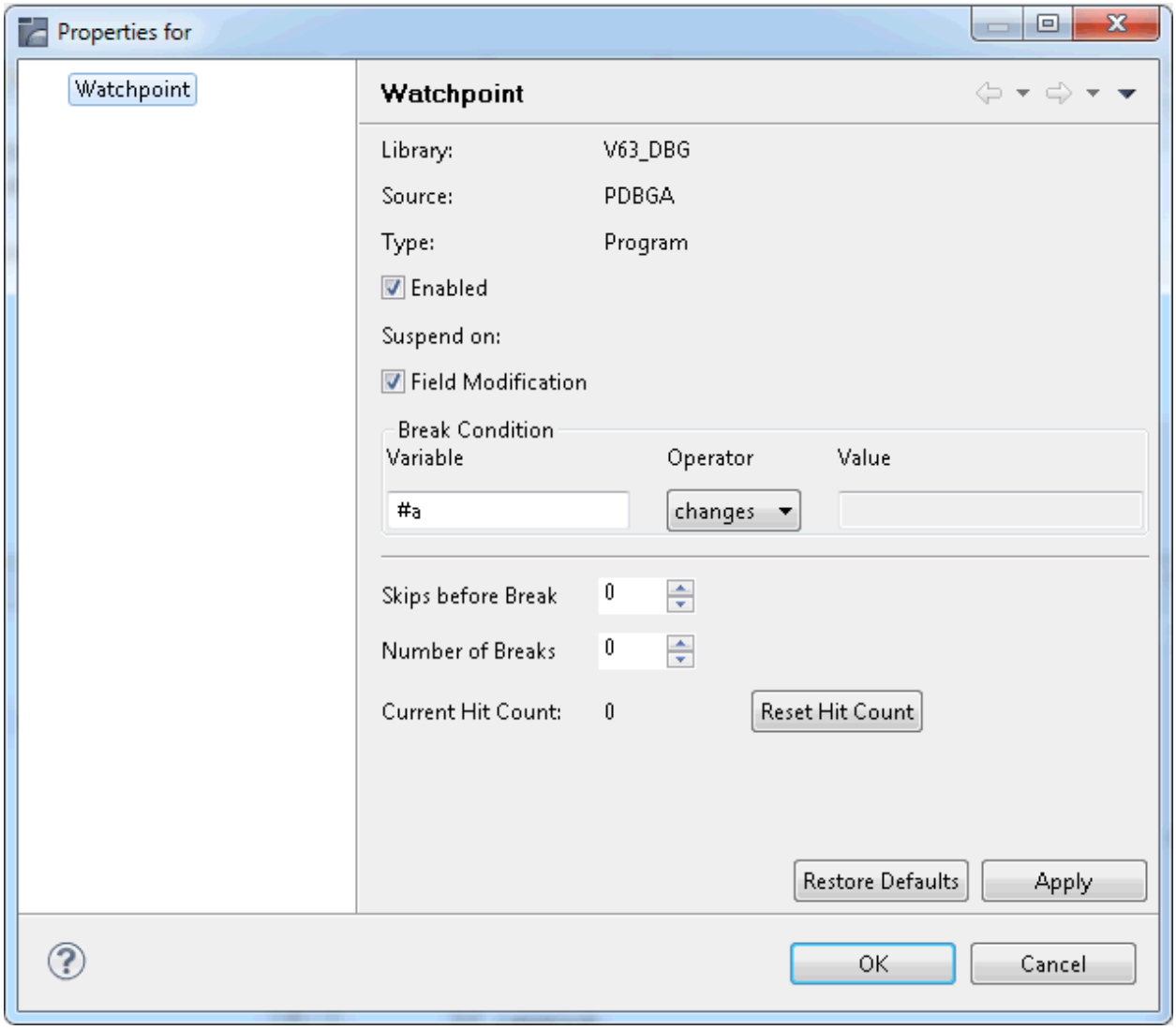
The maximum number of executions of the breakpoint. After this number has been reached, the breakpoint is ignored. The default is 0.

Reset Hit Count

When you choose this command button, the current hit count is reset to 0.

Watchpoint Properties

The following dialog box appears for a watchpoint.



You can set the following Natural-specific options:

Variable

The variable that is to be watched in the debugged program.

Operator/Value

To define a condition for the watchpoint, select an appropriate watchpoint operator and specify a value for this operator. If you do not specify a condition, the default setting (**changes**) applies.

The watchpoint operators are:

Operator	Activation of the Watchpoint
changes	Each time the variable is changed. Default.
EQ (=)	Only when the current value of the variable is equal to the specified value.
NE (!=)	Only when the current value of the variable is not equal to the specified value.
GT (>)	Only when the current value of the variable is greater than the specified value.
LT (<)	Only when the current value of the variable is less than the specified value.
GE (>=)	Only when the current value of the variable is greater than or equal to the specified value.
LE (<=)	Only when the current value of the variable is less than or equal to the specified value.

Skips before break
The number of skips before execution of the watchpoint if it is not to be executed until the program has run a certain number of times. The default is 0.

Number of breaks
The maximum number of executions of the watchpoint. After this number has been reached, the watchpoint is ignored. The default is 0.

Reset Hit Count
When you choose this command button, the current hit count is reset to 0.

Going to the Next Statement

This information applies only when you are debugging a program which is associated with a UNIX, OpenVMS or Windows environment.

You can instruct the debugger to skip code and to resume execution of the object with the source code line in which you have placed the cursor. The skipped code is not executed.

Caution: Depending on the code you want to skip, the **Set Next Statement** command may lead to unpredictable results. Use this command with care.

To go to the next statement

1. In the editor area of the **Debug** perspective, place the cursor in the source code line with which you want to resume execution. This line can be located before or after the current trace position. It must contain an executable statement such as `MOVE` or `PRINT` (it must not contain a non-executable statement such as `DEFINE DATA` or a comment).
2. Invoke the context menu and choose **Set Next Statement**.

   The new trace position is now indicated by the arrow in the marker bar of the editor window. When you resume debugging, the debugger continues the execution of the object with this source code line.

Notes:

1. The **Set Next Statement** command is only available for the object which is currently processed.
2. When debugging an object which is associated with a mainframe environment, the **Set Next Statement** is not available (it appears gray).

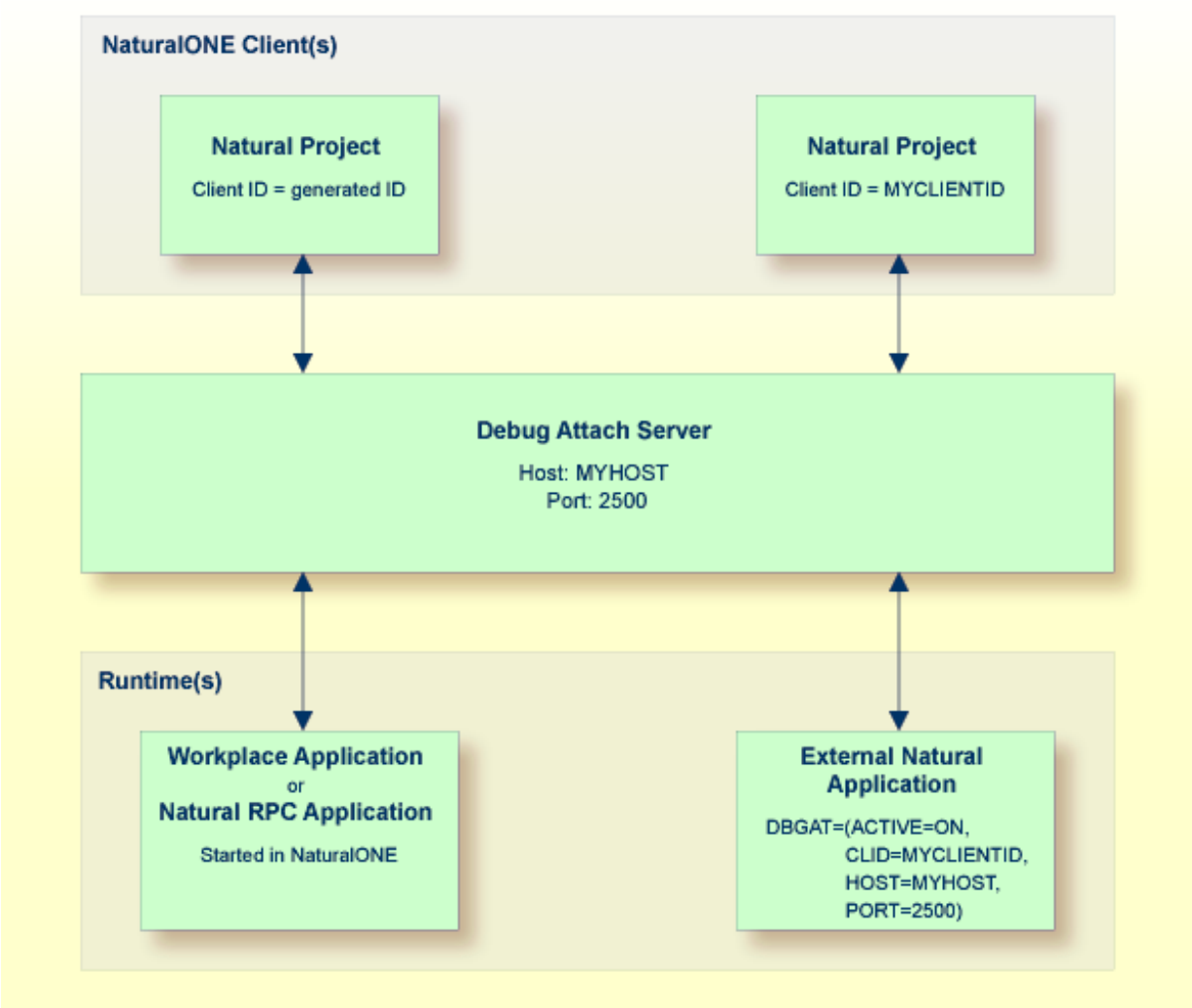
20 Using a Debug Attach Server

- General Information 290
- Starting the Debug Attach Server 292
- Debugging a Natural RPC Application 292
- Debugging an External Natural Application 293

General Information

The debug attach server is only available for Windows. It is delivered with the local Natural runtime of NaturalONE and with Natural for Windows as of version 6.3.13.

In order to debug workplace applications (that is, Natural for Ajax pages of type MFPAGE), Natural RPC applications or other external Natural applications, a debug attach server is required. The debug attach server acts as a broker between the NaturalONE clients and the attachable sessions. It enables the NaturalONE debugger to attach to a running application and links the debugger with the appropriate Natural runtime.



The debug attach server must be accessible to all NaturalONE clients and to all executing Natural sessions that want to participate in debugging.

- **Preparing the NaturalONE client**
  To enable debugging for a NaturalONE client, you have to enable the debug attach server in the Natural preferences of NaturalONE, and you have to specify the appropriate settings (host name and port number). See ***Debug Attach Settings*** in *Setting the Preferences*.

  All relevant sources for the application to be debugged must be contained in a Natural project. For example, you can download the sources from a Natural server into a Natural project, or you can check out the sources from your version control system.

  A Natural source is registered on the debug attach server if it contains a breakpoint. You have to set at least one breakpoint in the source with which you want to start your debug session.

  When you debug an application from within NaturalONE (that is, a workplace application or Natural RPC application), the sources containing the breakpoints are automatically registered.

  If you want to debug an external Natural application (that is, an application that is started, for example, with Natural for Windows), you have to register the sources containing the breakpoints explicitly using the command **Activate Debug Attach**.
- **Starting the Natural runtime in debug attach mode**
  The Natural runtime to be debugged has to be started in debug attach mode.

  When you debug an application from within NaturalONE (that is, a workplace application or Natural RPC application), this mode is set automatically.

  If you want to debug an external Natural application (that is, an application that is started, for example, with Natural for Windows), you have to set the debug attach mode explicitly using the Natural profile parameter `DBGAT`.

At runtime, when a registered breakpoint is hit during the execution of the application, a debugging session for the corresponding Natural runtime is launched and you can debug as described in ***Debugging Natural Applications***.

All Natural objects that have been registered for a specific client are automatically removed from the debug attach server when you exit NaturalONE.

Notes:

1. The debug attach server uses a client ID to manage its attach records. It is recommended that you always use the unique client ID which is automatically generated. However, if you want to debug an external Natural application or, if necessary, for testing purposes, you can manually define a custom client ID in the properties of a project. See ***Debug Attach Settings*** in *Changing the Project Properties*.
2. How to debug Natural RPC applications and external Natural applications is described below. For information on how to debug workplace applications, see *Executing and Debugging Workplace Applications* in the Natural for Ajax documentation.

Starting the Debug Attach Server

It is not required that each NaturalONE client starts its own debug attach server. It is sufficient to start one debug attach server which can be accessed by all NaturalONE clients, even if the clients run on different machines. It is only required that all NaturalONE clients define the appropriate debug attach settings in the Natural preferences.

The debug attach server is started using *natdas.exe*. You can find this program in the *\naturalone\natrun\bin* folder of your NaturalONE installation.

The syntax for starting the debug attach server is the following (the elements contained within the square brackets are optional):

```
natdas [-p port-number] [-o file-name]
```

where `port-number` is the number of the listener port, and `file-name` is the name of your trace file.

If you do not specify any parameters when starting the debug attach server using *natdas.exe*, the default port 2500 is used.

If you want to use a port other than the default port, or if you want to create a trace file, you have to specify additional parameters. For example:

```
natdas -p 9999 -o c:\temp\natdas.log
```

If you want to start the debug attach server using a Windows shortcut, you can specify the parameters as shown in the following example:

```
D:\SoftwareAG\NaturalONEnn\naturalone\natrun\bin\natdas.exe -p 9999 -o ↩
c:\temp\natdas.log
```

When you specify that a trace file is to be created and if a trace file with the same name already exists, the existing file is overwritten.

Debugging a Natural RPC Application

A Natural RPC application switches to a different Natural runtime environment.

The following steps assume that you have already created a Natural project containing the relevant sources for the Natural RPC application in NaturalONE (for example, a main program which invokes two subprograms using the `CALLNAT` statement).

To debug a Natural RPC application

1 Open the source editor for one of the subprograms which is invoked by the main program and set at least one breakpoint.

2 In the **Navigator** view or in the **Natural Navigator** view, select the main program.

3 Invoke the context menu and choose **Debug As > Natural Application**.

   Or:

   Press ALT+SHIFT+D, N.

   Or:

   The main program is now started in debug mode. You can use the **Debug** perspective as described in the section *Debugging Natural Applications*.

   When the first breakpoint which has been set in the subprogram is reached, the main debugging session turns into wait state. Since the RPC is executed on a different machine, a new debugging session is launched and debugging stops on the breakpoint.

   With **Step Return**, you return to the debugging session for the main program.

Debugging an External Natural Application

If you want to debug a Natural application that is started outside of NaturalONE (for example, a batch application which is started with Natural for Windows or UNIX), you have to start the application using the Natural profile parameter `DBGAT`. This parameter specifies the client ID to be used, and the name and port of the debug attach server. For detailed information, see the description of the `DBGAT` parameter in the Natural documentation for the appropriate platform.

The following steps assume that you have already created a Natural project containing the relevant sources for the Natural application in NaturalONE.

To debug an external Natural application

1 Open the source editor for a Natural program which belongs to the application and set at least one breakpoint.

2 In the **Navigator** view or in the **Natural Navigator** view, select the Natural project.

3 Invoke the context menu, choose **Properties** and then specify a custom client ID. See also *Debug Attach Settings* in *Changing the Project Properties*.

4 Invoke the context menu for the Natural project once more and choose **NaturalONE > Activate Debug Attach**.

This command registers the existing breakpoints in the debug attach server.

Note: This command is only visible when the debug attach server has been enabled in the Natural preferences.

5 Go to your external Natural application and start it using the `DBGAT` parameter.

Important: The parameters for the host name and port number that have been specified for the external Natural application using the `DBGAT` parameter must be the same as defined in your Natural preferences. The client ID that has been specified with the `DBGAT` parameter must be the same as in your project properties.

When the program in which the breakpoint has been set is about to be executed inside the corresponding Natural session, the debugger is launched and the application stops at the first breakpoint. You can now use the **Debug** perspective as described in the section *Debugging Natural Applications*.

6 If you do not want to continue debugging, you have to remove the application from the debug attach server. To do so, select the Natural project, invoke the context menu and choose **NaturalONE > Deactivate Debug Attach**.

Note: If you do not choose the above command, the application is automatically removed from the debug attach server when you exit NaturalONE.

VI Creating Application-Specific Messages

21 Creating Application-Specific Messages

- General Information 298
- Type, Name and Location of the Message Files 298
- Creating Message Files 299
- Opening an Existing Message File 301
- About the Error Message Editor 301
- Associated Views 303
- Translating a Message File 306
- Layout of a Message File 306

General Information

When you develop a Natural application, you may want to separate error or information messages from your Natural code and manage them separately. This makes it easy for you, for example, to standardize messages, to have predefined message ranges for different kinds of messages, to translate messages into other languages or to attach to a message a long text that explains it in more detail.

The Natural statements `INPUT` and `REINPUT` are used to issue the messages from a Natural program.

This section explains how to write your own application-specific messages. For more information on messages, see the description of the `SYSERR` utility in the Natural documentation for the appropriate platform.

Type, Name and Location of the Message Files

There are two types of messages:

- **User-defined Messages**
  User-defined messages are issued by applications written by a user. In the Eclipse workspace, they are normally stored in the *ERR* folder of a library. Each language is stored in a separate message file. A maximum of 9999 messages can be stored per library and message file.

  A user-defined message file always has the following name:

  ```
  NnnAPMSL.ERR
  ```

  where *nn* is the language code (01 through 60), for example, *N01APMSL.ERR* for English.

  For an overview of the language codes, see the description of the system variable `*LANGUAGE` in the Natural documentation for the appropriate platform.

- **Natural System Messages**
  Natural system messages are delivered by Software AG. They are not stored in a library. They are stored in internal structures of NaturalONE and cannot be modified.

In the Eclipse workspace, error message files are stored in readable format (ASCII) with the extension *ERR*. When an error message file is uploaded to a Natural server, the file is automatically converted to the appropriate format.

In the project properties, you can determine whether all error messages are to be deleted on the server before a new error message file is uploaded (see the description of the **Natural** page in the section *Changing the Project Properties*).

Creating Message Files

In the Eclipse workspace, you can create message files in different ways:

- by downloading a library containing message files or by downloading only single message files from a Natural server, see *Downloading an Existing Library or Object from a Natural Server*,
- by using a wizard, see below,
- by checking out a message file from the repository of your version control system.

To create a new message file using a wizard

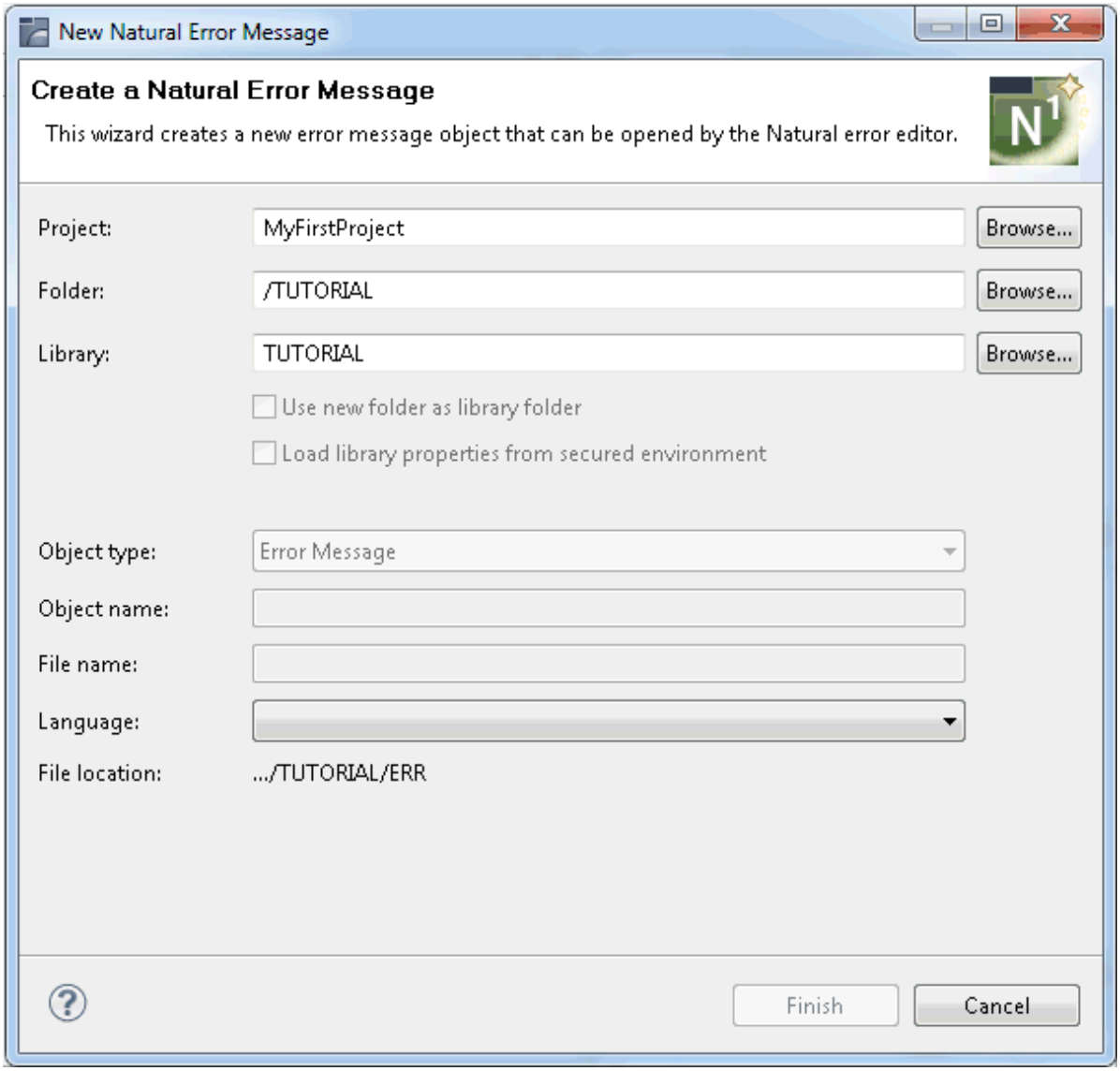
1 In the **Navigator** view or in the **Natural Navigator** view, select the library in which you want to store the new message file.

2 From the **File** menu, choose **New > Error Message**.

Or:

Invoke the context menu and choose **New > Error Message**.

The following dialog box appears.



3 From the **Language** drop-down list box, select the language for the message file.

The name for the message file (containing the language code for the selected language) is then shown in the **Object name** text box. The name is also shown in the **File name** text box. **File names**, however, are not supported for error messages.

The **File location** text box, which is read-only, shows the path where the new object will be created. The path that is shown here depends on the setting of the **Group new objects by object type** option in the **project properties**. When this option is not selected, this path may include the special folder *ERR* or a library folder. When this option is selected, the path includes the appropriate group folder. For further information, see ***Group Folders***.

For information on other options in this dialog box, see ***Creating Natural Objects***.

4 Choose the **Finish** button.

The new message file is created in the selected library and the error message editor is automatically invoked.

5 Add all required messages as described below.

6 Save the message file using the standard Eclipse functionality (for example, by pressing CTRL+S).

Opening an Existing Message File

When you open a message file, the error message editor is invoked.

To open message files

1 In the **Navigator** view or in the **Natural Navigator** view, select the message file(s) that you want to edit.

2 Invoke the context menu and choose **Open**.

Or:

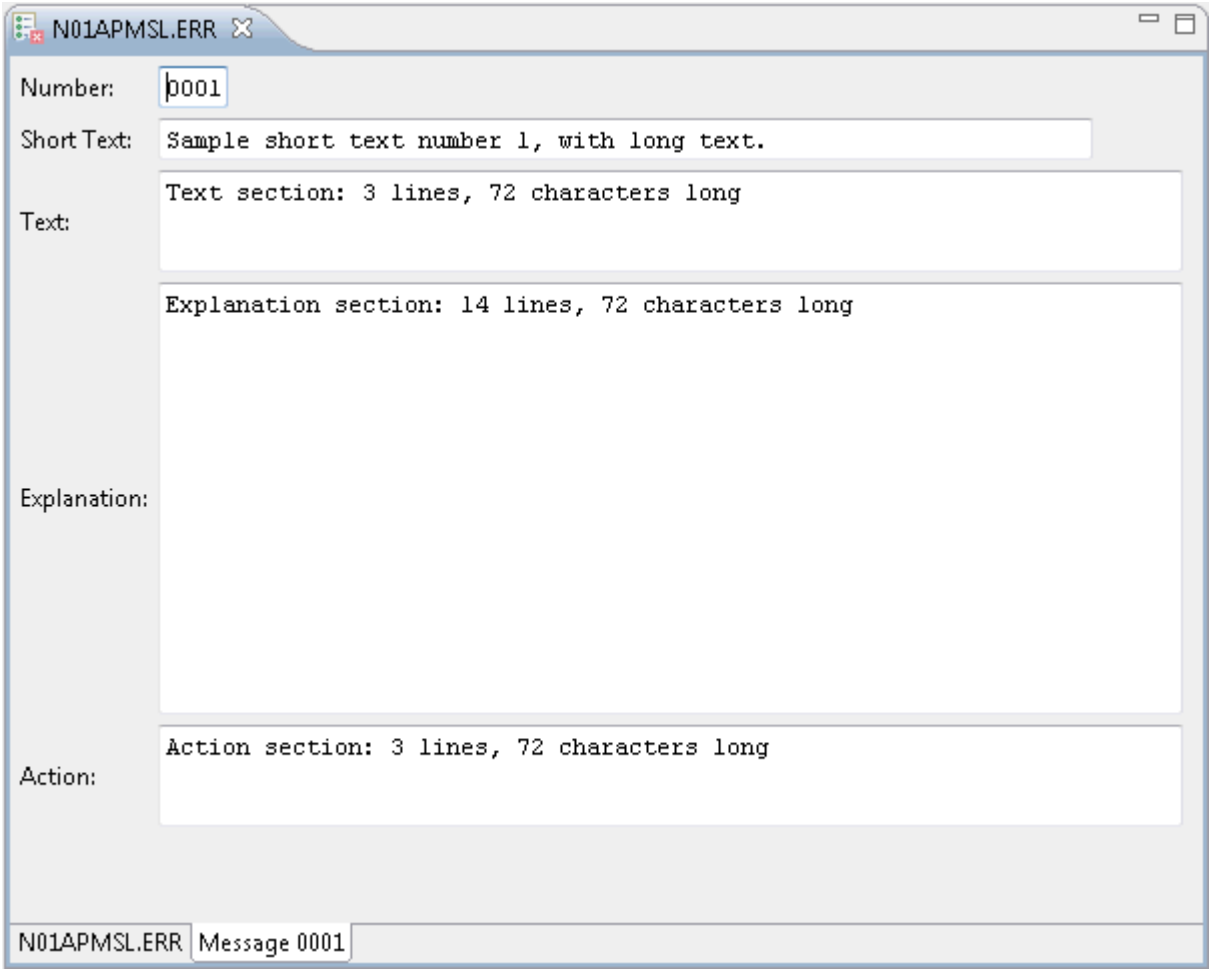
Double-click each message file that you want to open.

For each selected message file, an editor appears in the Eclipse editor area. You can now modify the messages as described below.

About the Error Message Editor

When you create or open a message file, the error message editor is invoked. The error message editor is a multi-page editor which provides the following types of pages:

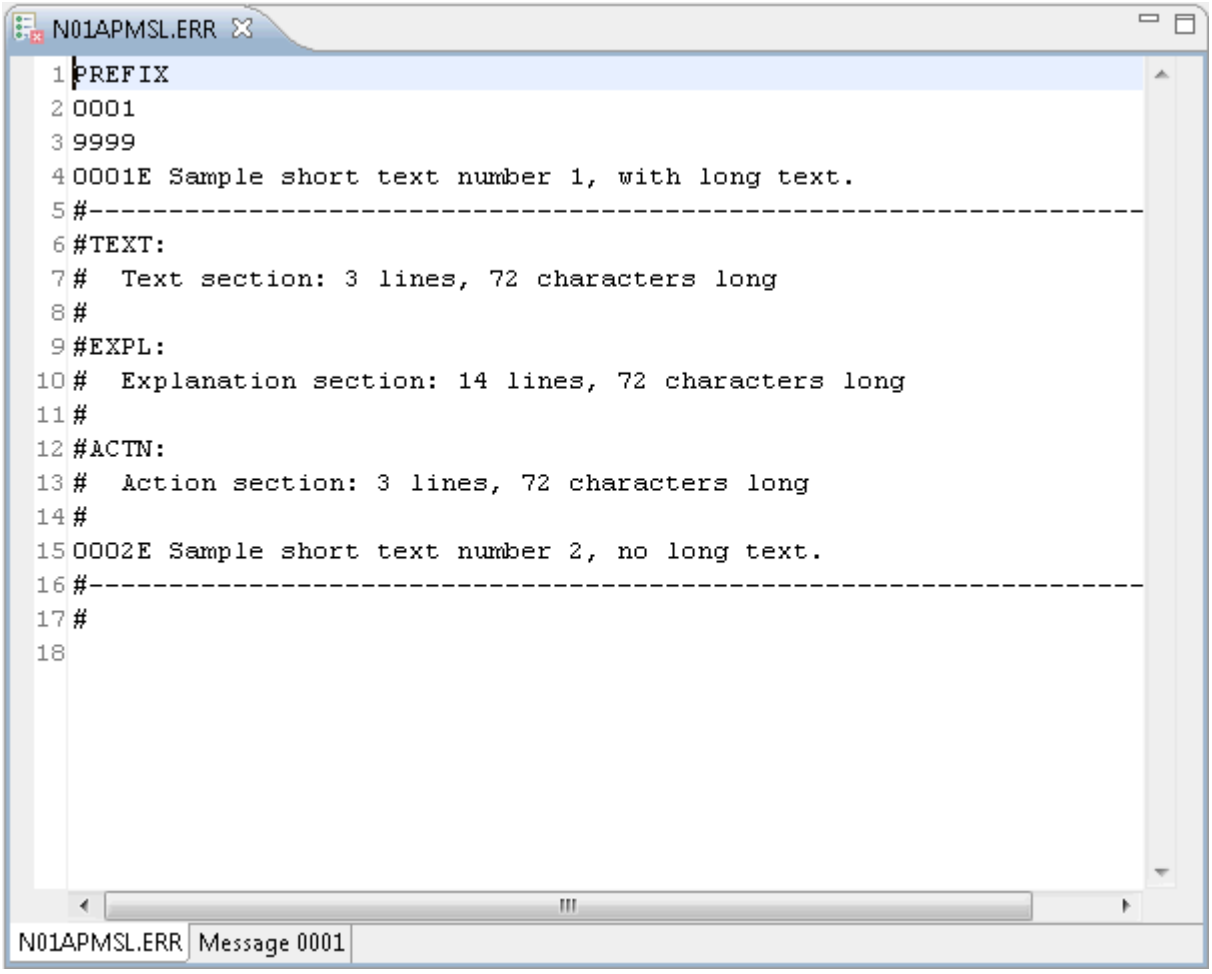
- **Form-based editor**
  The tab of this page contains the number of the message which is currently active. The page itself provides a form-based editor.



To add a new message, you simply enter the new message number in the **Number** text box. Empty input fields are then shown in which you enter all information: a short text (mandatory) and, if required, a long text for the message. The long text is entered in the text boxes labeled **Text**, **Explanation** and **Action**.

When you enter the number of an existing message in the **Number** text box, all text for this message number is shown. If required, you can then modify the existing text.

- **Text editor (read-only)**
  The tab of this page contains the name of the error message file. The page itself, which is read-only, shows the contents of the error message file in text format (see also *Layout of a Message File*). To add new messages, you have to use the form-based editor (see above).



Associated Views

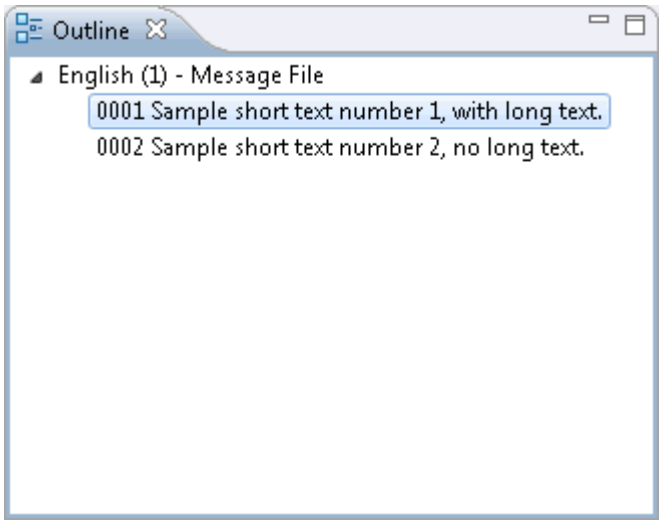
The error message editor uses the following views of the NaturalONE perspective:

- Outline View

- Properties View

Outline View

This view shows the language and all defined messages in a tree.



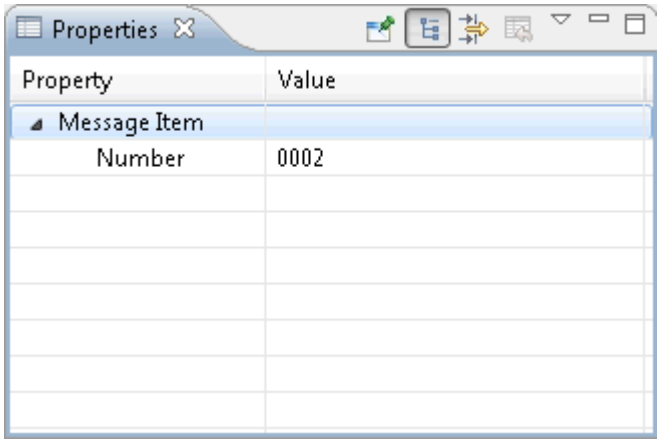
You can navigate to a particular message in the editor by selecting it in the **Outline** view, and vice versa.

If you want to delete a message, you do this in the **Outline** view. The context menu for a selected message provides the corresponding command.

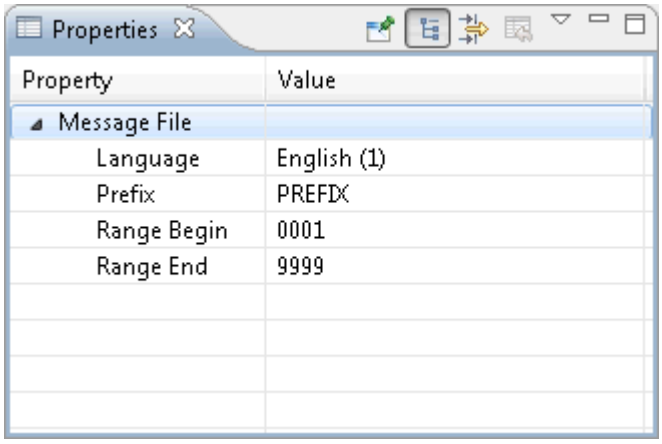
Properties View

This view shows the properties of the item that was last selected in the error message editor or in the **Outline** view.

The following properties are shown for the selected message. If you want to renumber a message, you do this in the **Properties** view; you can define any message number that is not yet used (it is not possible to define an existing number).



When you select the top-level node in the **Outline** view, the following properties are shown. You can change all values except for the language.



Property	Description
Language	The language that is defined for this message file. The language code is shown in parentheses. This value cannot be modified.
Prefix	The prefix of the message number to be displayed with the message. For example, the library ID.
Range Begin	The starting number for a range of messages.
Range End	The ending number for a range of messages. All message numbers that are defined in this message file must be within this range.

Translating a Message File

Messages can be translated into different languages. Each language is stored in a separate message file.

To translate a message file

1 Copy the message file that you want to translate.

2 Make sure that the name of the copied message file corresponds to the format described in *Type, Name and Location of the Message Files* and that it contains the correct language code.

3 Use the error message editor to overwrite the original messages with the messages in the appropriate language.

Layout of a Message File

A specific layout is required for message files, as shown in the following example. This format is automatically created when you use the form-based editor.

```
PREFIX
0001
9999
0001E Sample short text number 1, with long text.
#-------------------------------------------------------------------------------
#TEXT:
#  Text section: 3 lines, 72 characters long
#
#EXPL:
#  Explanation section: 14 lines, 72 characters long
#
#ACTN:
#  Action section: 3 lines, 72 characters long
#
0002E Sample short text number 2, no long text.
#-------------------------------------------------------------------------------
#
```

The first three lines contain the prefix and the message range (begin and end).

The messages start in line 4. Each error number is followed by an "E", which stands for "error".

Each message has at least a short text which follows directly after the message number.

There is a separator line (#---) below the short text.

The long text is optional. It follows below the corresponding short text (that is, below the separator line). Each line of the long text starts with a hash (#).

The following keywords identify the individual sections of the long text. These keywords must not be translated into other languages.

Keyword	Description
#TEXT:	Extended version of the short message text.
#EXPL:	Further explanation of the message.
#ACTN:	The action to be taken to resolve a problem, if relevant.

VII Using the Data Browser

22 Using the Data Browser

- General Information 312
- Creating a Report Template 312
- About the Data Browser Editor 314
- Selecting the Fields for the Report 314
- Displaying the Properties for a Field 316
- Setting Options for the Report 318
- Creating the Report 321
- Saving a Report Template 323
- Saving a Report 323
- Displaying the Properties for a Report 324
- Opening an Existing Report Template 325

General Information

With the data browser, you can quickly and easily issue database queries against Adabas and SQL databases. You can use the data browser during application development to find out whether the application works properly. Or you can simply use it to find out which data has been stored in a database.

So that you are able to generate data reports from Adabas or SQL databases which are available in your current Natural server environment, an appropriate DDM is required for each database file.

Creating a Report Template

To get a report from a database, a template is required which specifies the database fields that are to be retrieved from the database. You create (or edit) a so-called "report template" which defines the required fields. The report template is stored for further use in the Eclipse workspace.

To create a report template

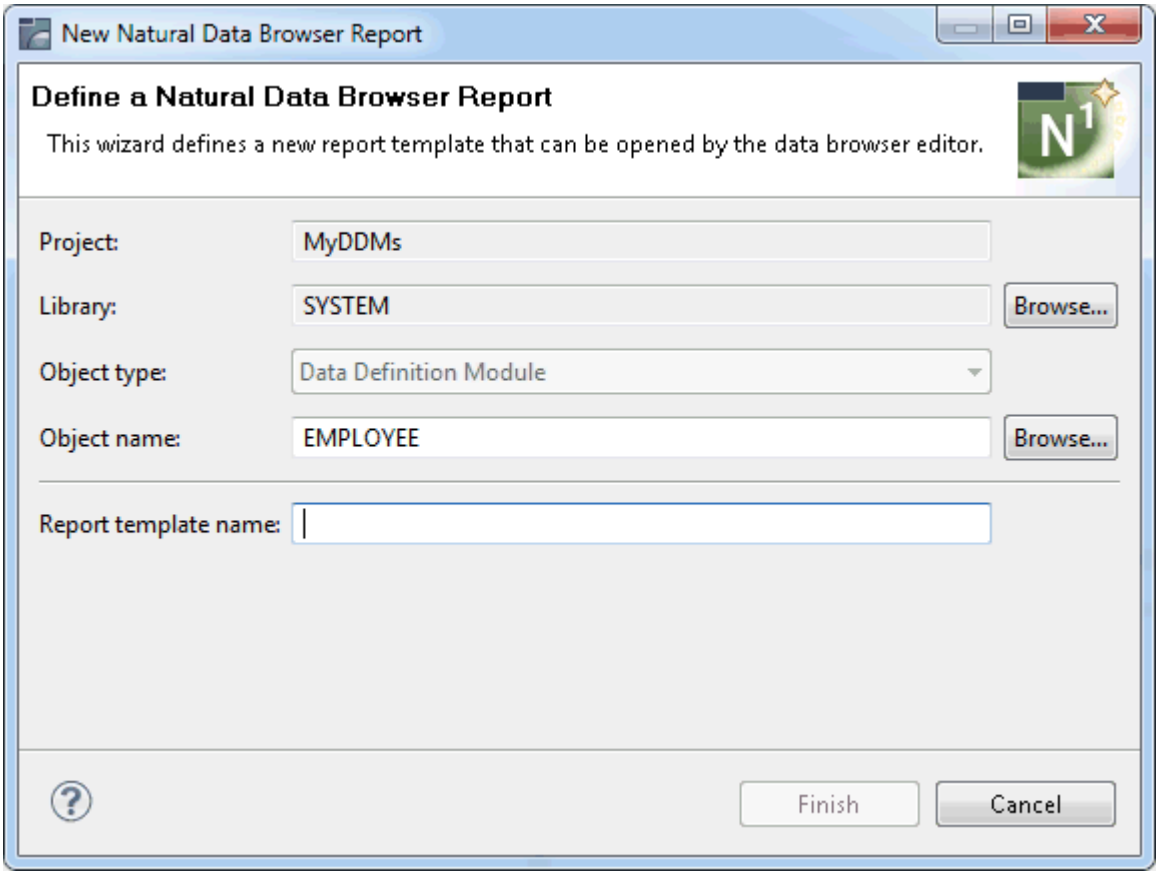
1 In the **Navigator** view or in the **Natural Navigator** view, select the DDM for which you want to define a new report.

2 From the context menu, choose **Browse Data**.

   Or:

   From the **File** menu, choose **New > Other**. In the resulting **New** dialog box, expand the **Natural** node, select **Browse Data** and then choose the **Next** button.

   The following dialog box appears.



3 Enter a name for the report template.

4 Choose the **Finish** button.

The selected DDM is now validated on the Natural server.

It may happen that there are differences between the local DDM and the DDM on the server (for example, when the DDM on the server has been changed in the meantime and therefore has a different version). It may also happen that the DDM is no longer available on the server or that it is not cataloged on the server. The corresponding information is then provided, for example, in the **Properties** view and/or in the **Problems** view.

When the DDM has been validated, a new report template for the selected DDM is opened in the editor area. For further information, see *About the Data Browser Editor*.

Note: The Adabas file which is referenced in the DDM may be protected. In this case a dialog to specify the Adabas password is displayed.

The new report template is an XML file that is stored in the *RES* folder of the library which contains the DDM that you have selected.

About the Data Browser Editor

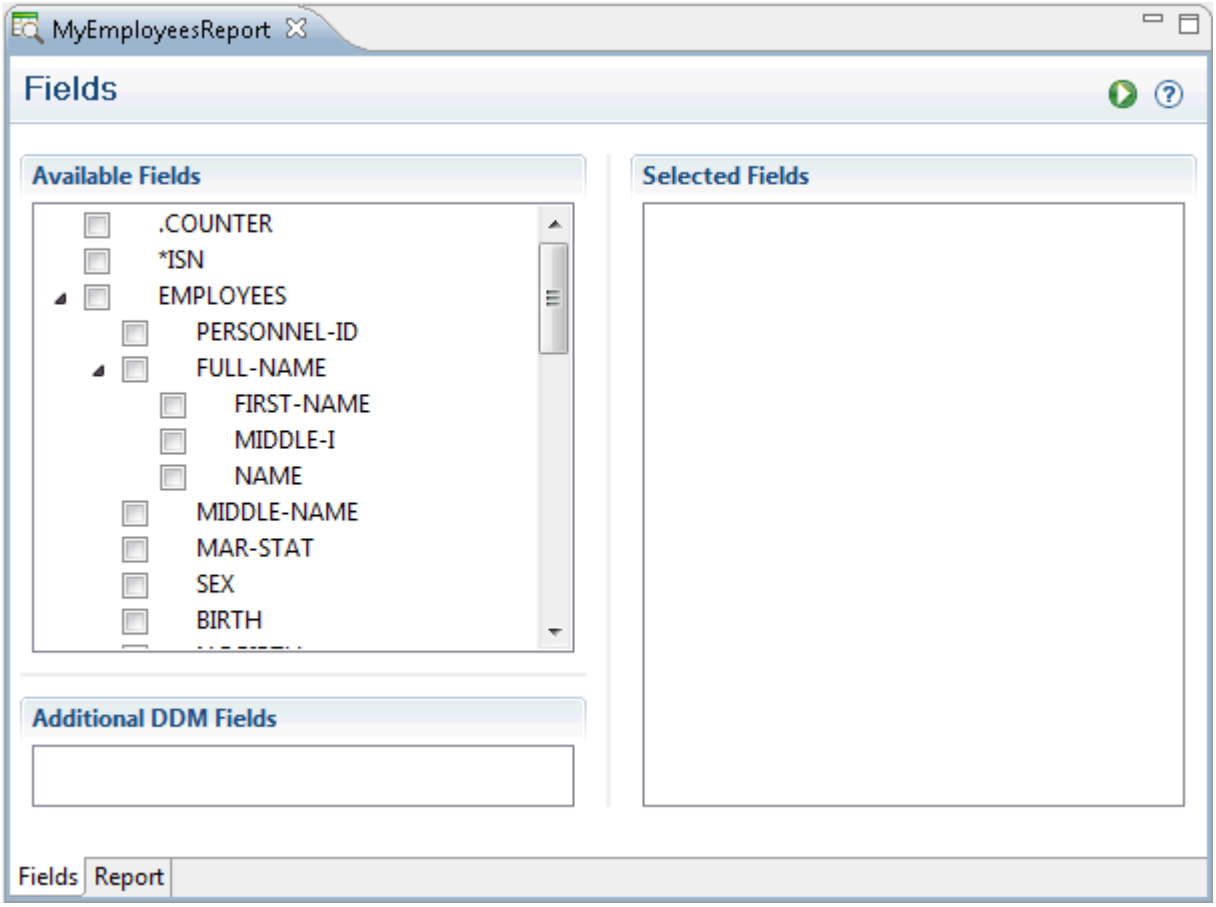
The report templates are edited using the data browser editor. This editor is invoked when you **create** a new report template or when you **open** an existing report template.

The data browser editor is a multi-page editor which provides the following pages:

- **Fields**
  On this page, you select the fields that are to be included in the report. See ***Selecting the Fields for the Report***.
- **Report**
  On this page, you can set several options (optional). See ***Setting Options for the Report***.

Selecting the Fields for the Report

The **Fields** page of the data browser editor shows a tree structure with the fields of the DDM. Example:



By default, no fields are selected.

The following general fields are shown at the top of the tree, before the DDM name:

- **.COUNTER**
  When selected, the records are numbered in the sequence in which they are read from the database. This number is then shown in the report. This feature is available as of Natural Development Server Version 2.2.6.
- ***ISN**
  This corresponds to the Natural system variable `*ISN`. When selected, the internal sequence number (ISN) of a record is shown in the report. This field is only available for Adabas and Tamino databases. This feature is available as of Natural Development Server Version 2.2.6.

Groups can be expanded and collapsed. For a multiple field, the indices of the separate dimensions are indicated in parentheses. For a group, only an asterisk is shown in parentheses to indicate that this is a multiple group.

Note: Under **Additional DDM Fields**, you can see the fields which are available in the current DDM, but which are not available in the report template. This may only happen

when you have selected a different DDM on the **Report** page. See *Setting Options for the Report*.

To select the fields for the report

1 In the tree structure of the **Fields** page, activate the check box for each field and/or group that you want to include in the report.

When you activate the top-most check box of the DDM (next to the DDM name), all DDM fields are automatically selected.

When you activate the check box for a group, all individual fields and subgroups that belong to the group are automatically selected.

Each selected field is shown on the right side of the **Fields** page.

If you want to remove a selected field, simply deactivate the corresponding check box in the tree structure.

2 Optional. In order to design the report, you can arrange the sequence of the selected fields individually. To do so, select one or more fields on the right side of the **Fields** page and drag them to the desired positions.

As long as you do not change the sequence of the selected fields, any new fields that you select are inserted in the same sequence as in the tree structure. However, if the selected fields have already been arranged individually, further fields that are selected in the tree structure are always added at the end of the list. Thus, the individual sequence is not corrupted.

In the list of selected fields, you can choose **Reset** from the context menu. This rearranges the selected fields in the same order as in the tree structure.

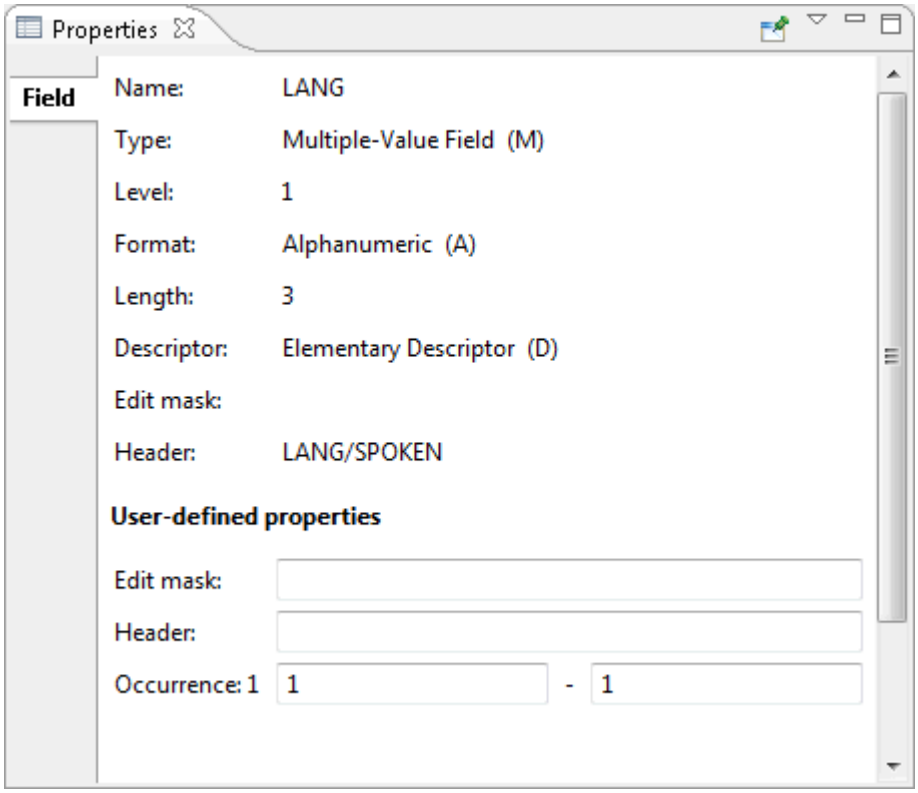
3 Optional. When you select a field (either in the list of available or selected fields), information on this field is shown in the **Properties** view. Some field properties have modifiable options. See *Displaying the Properties for a Field* for further information.

Displaying the Properties for a Field

The **Properties** view shows additional information for the entry which is currently selected on the **Fields** page of the data browser editor.

For the DDM, the following information is shown: database ID, file number, type (for example, Adabas) and DDM name (long name).

For a field, the following information is shown:

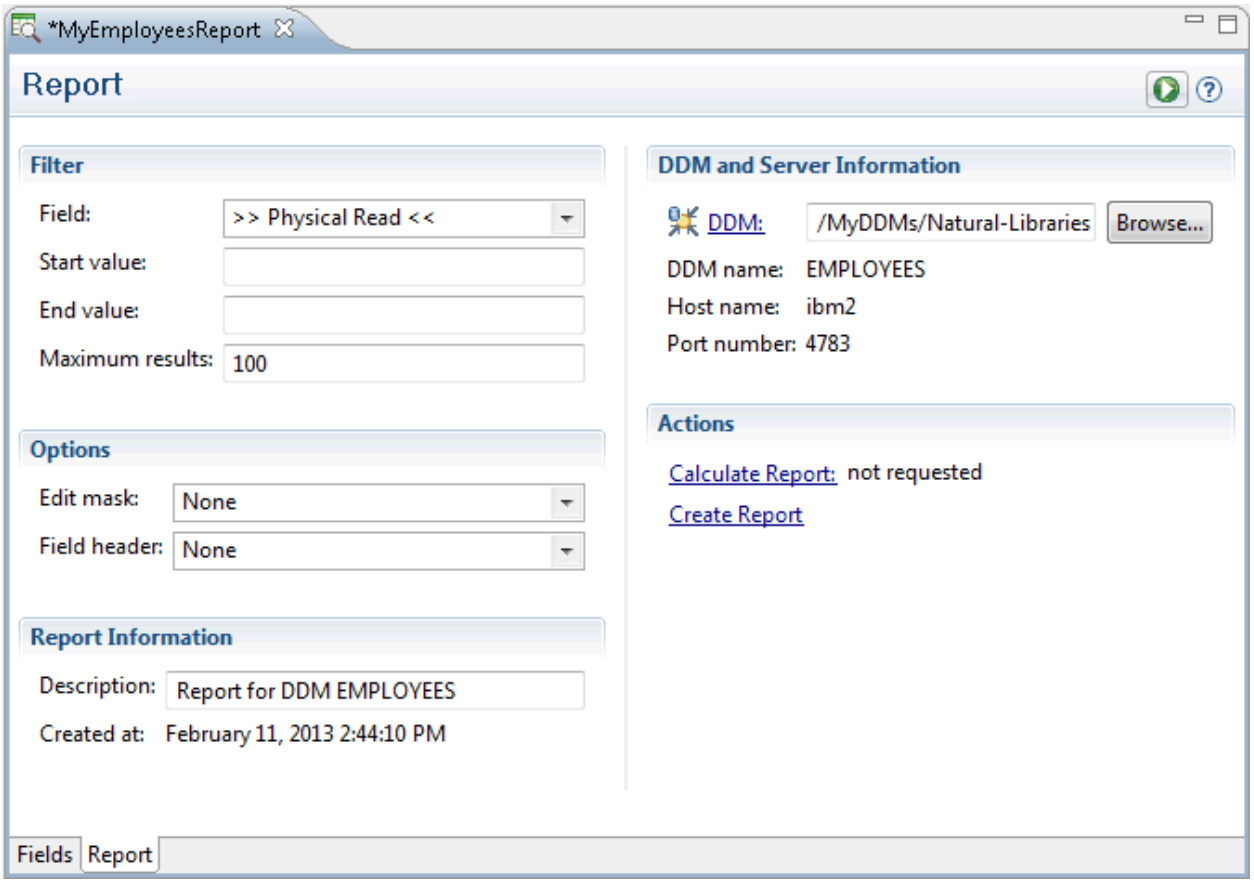


Option	Description
Name	Name of the field.
Type	Type of field: Elementary field. Group (G). Multiple-value field (M). Periodic group (P).
Level	Level number assigned to the field.
Format	Natural data format of an elementary field, such as "Alphanumeric (A)".
Length	Length of an elementary field.
Descriptor	Descriptor type: Elementary descriptor (D). Hyperdescriptor (H). Non-descriptor (N). Phonetic descriptor (P). Subdescriptor or superdescriptor (S). When a descriptor is not defined, this text box is blank.
Edit mask	Two text boxes are shown for the edit mask.

Option	Description
	The first text box shows the value that is defined in the DDM. This is the default edit mask to be used when the field is output, for example, with a `DISPLAY` statement. This value cannot be modified. The second text box (under the heading **User-defined properties**) is modifiable. This edit mask is used in the report in order to show the values of a field. When the DDM-defined or user-defined edit mask is to be used in the report, you must also set the corresponding option on the **Report** page of the data browser editor. See ***Setting Options for the Report***.
Header	Two text boxes are shown for the header. The first text box shows the value that is defined in the DDM. This is the header to be produced for each field specified in a `DISPLAY` statement. This value cannot be modified. The second text box (under the heading **User-defined properties**) is modifiable. You can specify a user-defined header for this field. When the DDM-defined or user-defined header is to be used in the report, you must also set the corresponding option on the **Report** page of the data browser editor. See ***Setting Options for the Report***.
Occurrence *n*	Only shown for a multiple field. The range that is to be used in the report can be modified. You can specify a different start value and a different end value for the range. This feature is available as of Natural Development Server Version 2.2.4. **Note:** When the preference to **Use maximum number of occurrences** is selected, the defined number of occurrences for multiple fields and periodic groups in the Predict file for the Adabas DDM are used. If no Predict file is defined or Predict is not installed on the server, the maximum number of occurrences for a multiple field (191) or periodic group (91) is used. See also ***Data Browser*** in *Setting the Preferences*. **Caution:** If you specify a large range, this may result in a long waiting time.

Setting Options for the Report

Optional. When you have selected the fields that are to be included in the report, you can set several options. This is done on the **Report** page of the data browser editor.



You can set the following options:

Option Name	Description
Filter	
Field	The report is sorted according to the value that is selected in this drop-down list box. By default, the records are read as they stand physically in the file (physical read). You can select another sorting criterion from this drop-down list box. The drop-down list box provides for selection the ISN and all descriptors that are defined in the DDM (including subdescriptor and superdescriptor).
Start value / End value	The range of values of the descriptor field can be defined by a start value and/or by an end value. If only an end value is defined, the start value is set to the format-dependent default initial value (for example, it is set to blank for format A or to 0 for format N). The start value must be less than the end value.
Maximum results	All filter criteria can be further limited by entering an absolute record limit (the default setting is 100 records). Regardless of the filter criteria, only the corresponding number of records is written to the report. By entering 0 (zero), the record limit is switched off.
Options	

Option Name	Description
Edit mask	The edit masks to be used in the report. This drop-down list box provides for selection the following options: ■ **None** Edit masks are not used for the fields in the report. ■ **DDM-defined** For all fields in the report, the edit masks as defined in the DDM are used. ■ **User-defined** For all fields in the report, the edit masks as defined in the **Properties** view are used. See *Displaying the Properties for a Field*. ■ **User- or DDM-defined** For all fields in the report, the user-defined edit masks as defined in the **Properties** view are used. When a user-defined edit mask cannot be found for a field, the edit mask as defined in the DDM is used. When a DDM-defined edit mask cannot be found, no edit mask is used for this field.
Field header	The column headers to be used in the report. This drop-down list box provides for selection the following options: ■ **None** Field headers are not used as column headers. Instead, the field names are used as the column headers. ■ **DDM-defined** The field headers as defined in the DDM are used as the column headers. ■ **User-defined** The field headers as defined in the **Properties** view are used as the column headers. See *Displaying the Properties for a Field*. ■ **User- or DDM-defined** The user-defined field headers as defined in the **Properties** view are used as the column headers. When a user-defined field header cannot be found, the field header as defined in the DDM is used as the column header. When a DDM-defined field header cannot be found, the field name is used as the column header.
Report Information	
Description	A brief description for the report. You can specify any description you like.
Created at	Shows the date and time when the report definition was created.
DDM and Server Information	
DDM	Shows the path to the DDM to which this report definition pertains. When you choose the **DDM** action, this DDM is opened in the **DDM editor**. Using the **Browse** button, you can also use another DDM with the current report definition. When you select a DDM which is not suitable for the current report definition, errors and/or warnings are shown in the **Properties** view and in the **Problems** view. On the **Fields** page of the data browser editor, the errors and warnings are indicated

Option Name	Description
	by corresponding icons, and all fields that are not available in the report definition are listed under **Additional DDM Fields**.
DDM name	Shows the name of the DDM to which this report definition pertains.
Host name	Shows the name of the host on which the DDM is located.
Port number	Shows the port number on the above host.
Actions	
Calculate Report	When you choose this action, the number of records is calculated and shown on this page.
Create Report	When you choose this action, the report is created. See also *Creating the Report*.

Creating the Report

When you have selected the fields that are to be included in the report (and when you have set any optional report options), you can create the report.

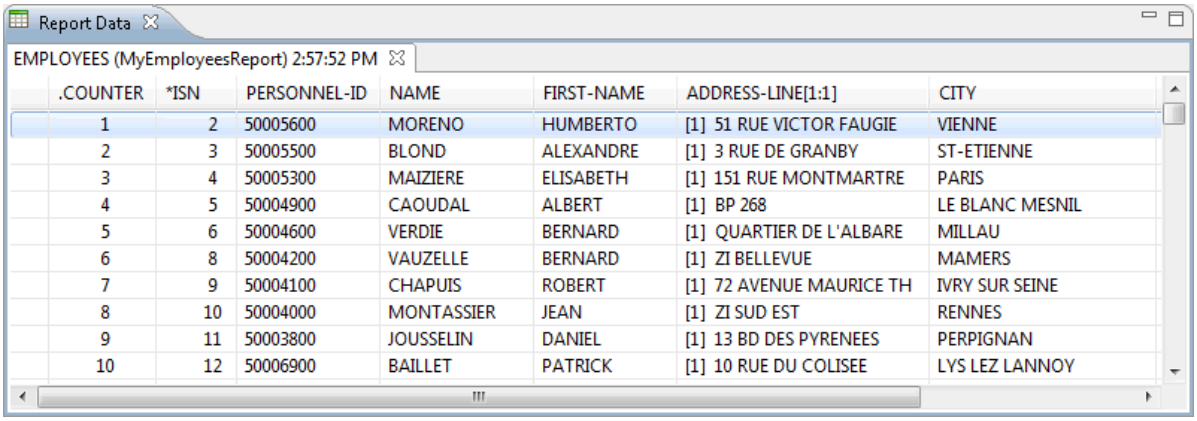
To create the report

- Choose the following button which is shown at the top of the data browser editor (it is available on the **Fields** page and on the **Report** page):

  Or:

  On the **Report** page, choose the **Create Report** action.

  The report is generated. It is shown in the **Report Data** view. Example:



.COUNTER	*ISN	PERSONNEL-ID	NAME	FIRST-NAME	ADDRESS-LINE[1:1]	CITY
1	2	50005600	MORENO	HUMBERTO	[1] 51 RUE VICTOR FAUGIE	VIENNE
2	3	50005500	BLOND	ALEXANDRE	[1] 3 RUE DE GRANBY	ST-ETIENNE
3	4	50005300	MAIZIERE	ELISABETH	[1] 151 RUE MONTMARTRE	PARIS
4	5	50004900	CAOUDAL	ALBERT	[1] BP 268	LE BLANC MESNIL
5	6	50004600	VERDIE	BERNARD	[1] QUARTIER DE L'ALBARE	MILLAU
6	8	50004200	VAUZELLE	BERNARD	[1] ZI BELLEVUE	MAMERS
7	9	50004100	CHAPUIS	ROBERT	[1] 72 AVENUE MAURICE TH	IVRY SUR SEINE
8	10	50004000	MONTASSIER	JEAN	[1] ZI SUD EST	RENNES
9	11	50003800	JOUSSELIN	DANIEL	[1] 13 BD DES PYRENEES	PERPIGNAN
10	12	50006900	BAILLET	PATRICK	[1] 10 RUE DU COLISEE	LYS LEZ LANNOY

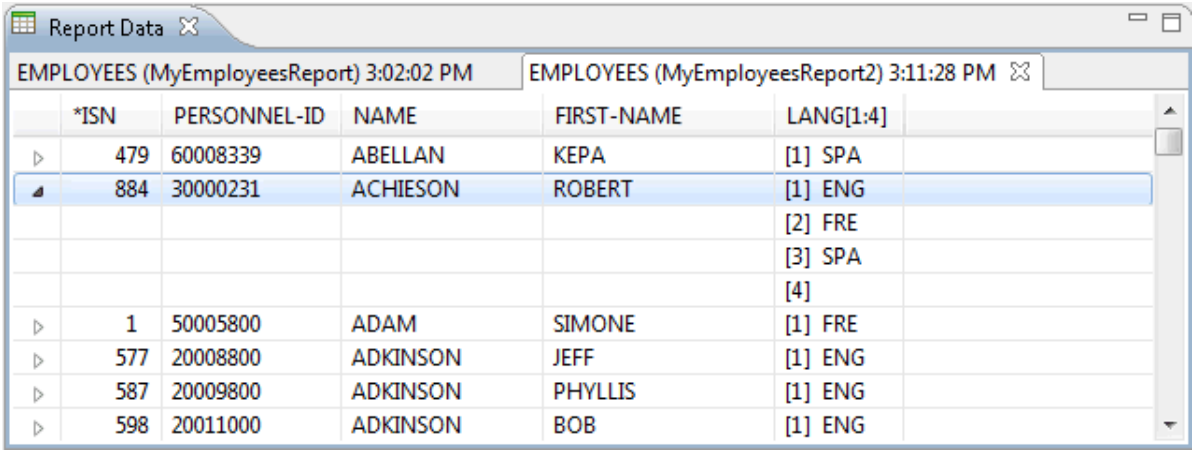
Note: This view is automatically opened when you create a report. It is not shown by default when you open the NaturalONE perspective. See also *Showing a View of the NaturalONE Perspective*.

In the **Report Data** view, you can

- sort a report according to a specific column by clicking on the column header;
- drag a column header to a different position.

When you have created more than one report, all of these reports are shown in the **Report Data** view (unless you have closed them). The tab of each report page contains the name of the DDM, the name of the report template that has been used to create the report, and a time stamp which indicates when this report has been created.

When a range is defined for a field, this is indicated in the column header (for example, "1:4"). In this case, only the first element of the range is initially shown and expand/collapse icons are provided. You can also double-click an entry in order to expand or collapse it. When you expand an entry, all elements of the range are shown (see the example below). When you move the mouse pointer over a column which shows ranges (either in expanded or collapsed state), a tooltip appears, listing all elements of the range. This feature is available as of Natural Development Server Version 2.2.4.



When expand/collapse icons are shown in the report, the context menu contains the commands **Expand All** and **Collapse All**. Using these commands you can expand or collapse all entries in the report at once.

Saving a Report Template

You save a report template using the standard Eclipse functionality (for example, by pressing CTRL+S).

The following applies when your report template has been changed in such a way so that it contains the former template structure, but pertains to a different DDM. When you save the report template (with **Save** or **Save As**), a dialog box appears asking whether you want to use the new DDM with the report template. You have two options for saving:

- To use the new DDM, you choose the **Yes** button.

  The fields of the new DDM are shown as the available fields on the **Fields** page.

  The fields that are also available in the new DDM remain selected on the **Fields** page. All other selected fields are cleaned. For example, when the selected field `PERSONNEL-ID` is used in both DDMs, this field is kept.

  When a descriptor which has been defined as a filter on the **Report** page is no longer available in the new DDM, the filter is reset to its default value.

- If you do not want to use the new DDM, you choose the **No** button.

  The definition of the new DDM is disregarded and the report template is saved with the original DDM. The next time you open the report template, you can see that the original DDM is used again.

Saving a Report

You can save each report which is currently shown in the **Report Data** view. Different commands are available for this purpose. You can either save the whole report or just selected lines of the report. The result is a text file containing, by default, a comma-separated list of fields. The character that is used to separate the field values is determined by the Natural input delimiter character (`ID` parameter) as defined in the **character assignments** of the current project properties.

To save the report

1. In the **Report Data** view, go to the page which contains the report that you want to save.
2. In the list of fields, open the context menu and choose **Save Report**.

   A dialog box appears.
3. Select a folder and specify a file name.

4 Choose the **Save** button.

To save selected lines of the report

1 In the **Report Data** view, go to the page which contains the report that you want to save.

2 In the list of fields, select each line that is to be included in the report.

   You can use the standard key combinations for selecting a range of fields (using the SHIFT key) or individual fields (using the CTRL key).

3 Open the context menu and choose **Save Selection**.

   A dialog box appears.

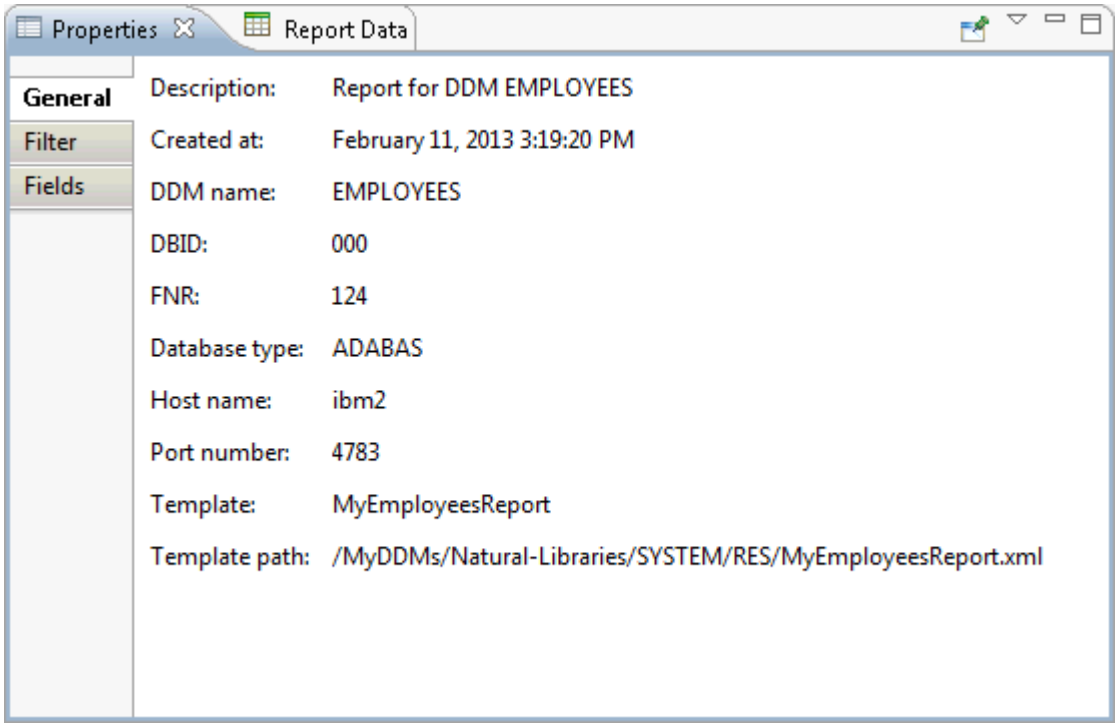
4 Select a folder and specify a file name.

5 Choose the **Save** button.

Note: A file extension is not automatically added to the file name. You have to specify it yourself.

Displaying the Properties for a Report

When you select a report in the **Report Data** view, the **Properties** view shows metadata for this report.



The following pages are provided:

- **General**
  Shows general information about the report. This includes the name and location of the DDM and the report template that has been used to create the report.
- **Filter**
  Shows the filter criteria for the report. These are the definitions which have been made on the **Report** page of the data browser editor (see *Setting Options for the Report*). In addition, the number of processed records is shown.
- **Fields**
  Shows the fields that are used in the report. These are the fields that have been selected on the **Fields** page of the data browser editor (see *Selecting the Fields for the Report*).

Opening an Existing Report Template

A report template is an XML file that is stored in the *RES* folder of the library which contains the DDM to which it pertains. When you open a report template, you can change all required information as described previously and you can create a new report.

To open a report template

- In the **Navigator** view or in the **Natural Navigator** view, select the report template, invoke the context menu and choose **Open**.

  Or:

  Double-click the report template.

  The report template is opened in the data browser editor.

Note: The report template may be copied from the *RES* folder of the original into the `RES` folder of a new Natural project. To find the DDM which pertains to the report template, the new Natural project must have a reference to the original project. For further information about referenced projects, see *Types of Natural Projects*.

VIII Generating API Documentation with NATdoc

23 Generating API Documentation with NATdoc

- Quick Start 330
- Generating NATdoc 331
- Location of the NATdoc Files 334
- Previewing the API Documentation in the NATdoc View 335
- Documentation Comments in the Source Code 336
- Special Comment Files 338
- Overview of NATdoc Tags 339
- Where Can Tags be Used? 342
- Using Custom Templates 343
- Detailed Template Descriptions 345

Quick Start

NATdoc works very similar to Javadoc. As Javadoc, NATdoc is a tool for generating API documentation in HTML format from doc comments in source code.

NATdoc processes the source code from the following Natural object types:

Program
Subprogram
Subroutine
Function
Copycode
Helproutine
Global data area (GDA)
Local data area (LDA)
Parameter data area (PDA)

The processed files must be valid, error-free Natural source files that can be processed by the Natural parser. All sources must be contained in the same project. Sources such as parameter data areas or copycodes must be part of the same Natural library.

NATdoc requires that a specific type of comment is used (`/**`) and that the comments appear in certain positions (before the first statement and within a `DEFINE DATA` block). For example:

```
* >Natural Source Header 000000
* :Mode S
* :CP windows-1252
* <Natural Source Header
/** Sample Program.
/**
/** :author John Doe
/**
DEFINE DATA PARAMETER
1 #X (I2) /** :in   The 1st operand.
1 #Y (I2) /** :in   The 2nd operand.
1 #R (I2) /** :out  The result.
...
END-DEFINE
```

To find out how the comments in your applications will turn out, you generate NATdoc as described below. For this purpose, you should prepare an HTML page with a bit of contents such as the following (this is the overview page that you have to specify in the wizard):

```
<html>
<body>
<p>This is the text for the overview.</p>
</body>
</html>
```

If you have not yet inserted any NATdoc-specific comments, you can nevertheless use the generator to find out how the documentation for your existing application will appear, even if you do not yet have prepared an overview page.

Generating NATdoc

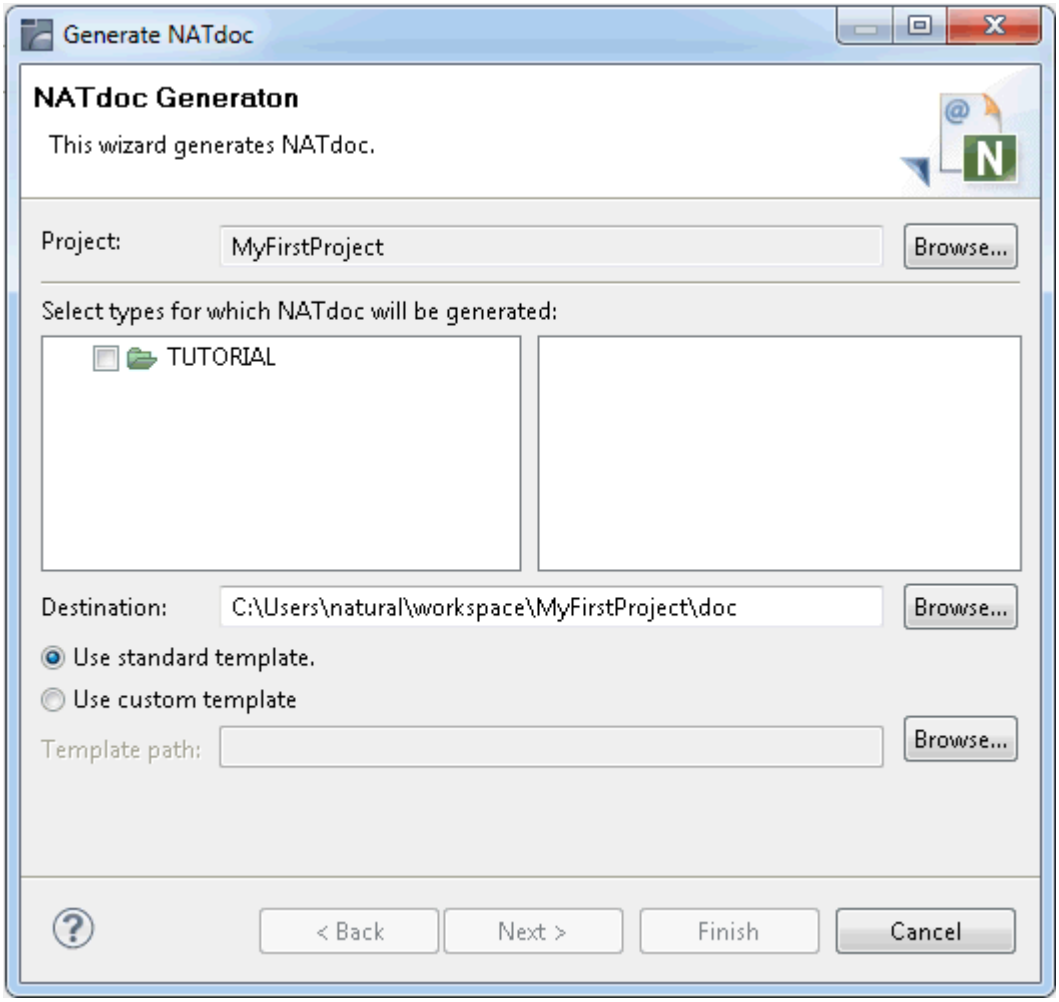
NATdoc is generated using a wizard.

To generate NATdoc

1 In the **Navigator** view or in the **Natural Navigator** view, select the project, library or source file for which you want to generate NATdoc.

2 From the **Project** menu, choose **Generate NATdoc**.

The following dialog box appears.



3 Select all types for which NATdoc is to be generated.

4 Specify the following information:

Destination

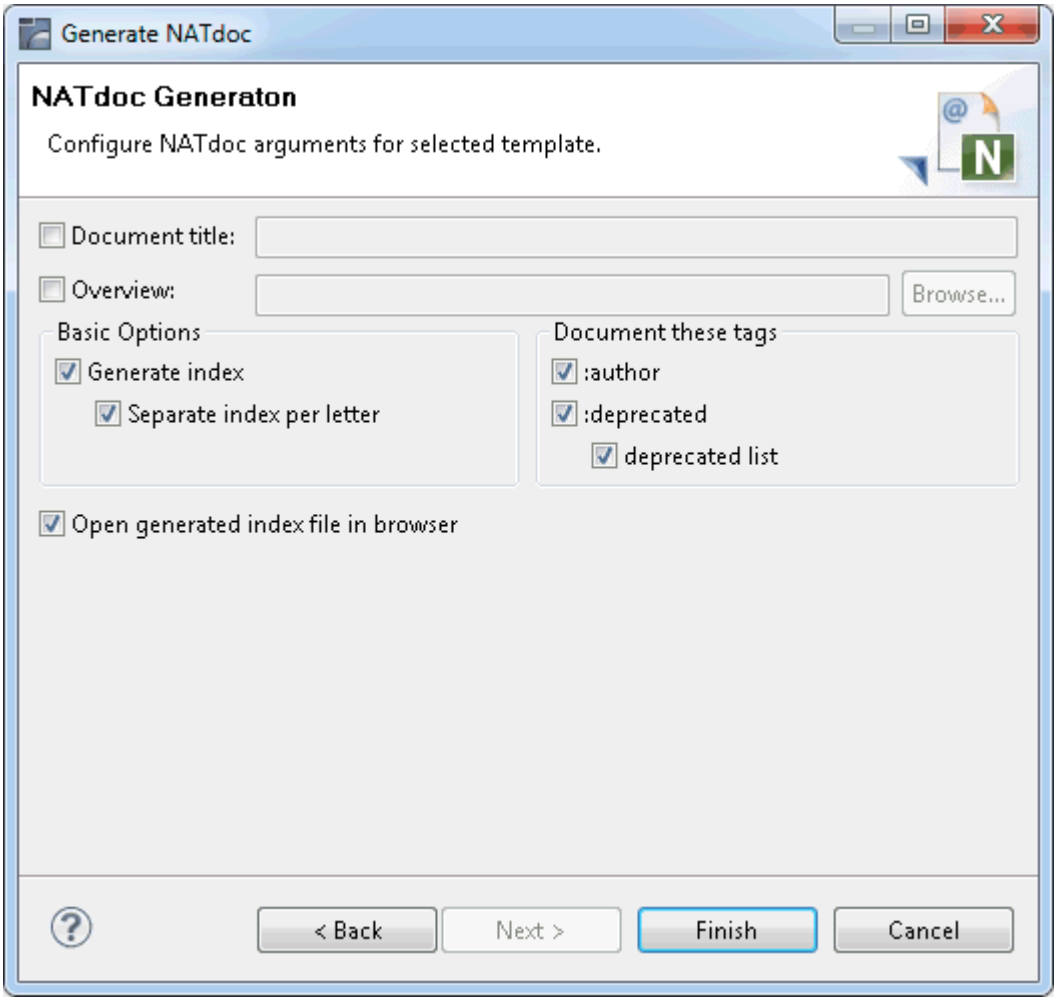
If required, specify a different destination folder. This is the folder into which NATdoc is to be generated. The destination must be within the same project.

Use standard template / Use custom template

Specify whether you want to use the standard template or a custom template.

The standard template is provided with NaturalONE. You can also create your own custom templates, on the basis of the standard template; in this case, you have to specify the path to your custom template. See also *Using Custom Templates*.

5 Choose the **Next** button to proceed to the next page.



6 Specify all required information as described below.

Document title

When enabled, you can specify the text that is to appear in the title bar of the browser and in any bookmarks/favorites that a user creates for this page.

Overview

When enabled, an overview page must exist. It can have any file name. You have to specify the path to this page.

The overview page has to be provided by the developer as an HTML page. This page contains a brief description of the project. The text from the HTML page will be included near the top of the generated *index.html* page.

See also ***Special Comment Files***.

Generate index

When enabled, an index is generated.

Separate index per letter
When enabled, a separate page is generated for each letter (for example, all entries starting with the letter A are written to a separate page, and all entries starting with the letter B are written to another separate page).

:author
When enabled, the text from the `:author` tag is available in the generated documentation.

:deprecated
When enabled, the text from the `:deprecated` tag is available in the generated documentation.

deprecated list
When enabled, the list of programs, subroutines etc. that are not to be used is generated into the documentation.

Open generated index file in browser
When enabled, the generated index file is automatically opened in the browser when the generation is completed.

7 Choose the **Finish** button to generate NATdoc.

When your project contains many files, the generation may take quite some time. Information about the generation process is provided in the **Console** view.

When the generation is complete and when **Open generated index file in browser** was enabled, the documentation is automatically opened in a browser.

Location of the NATdoc Files

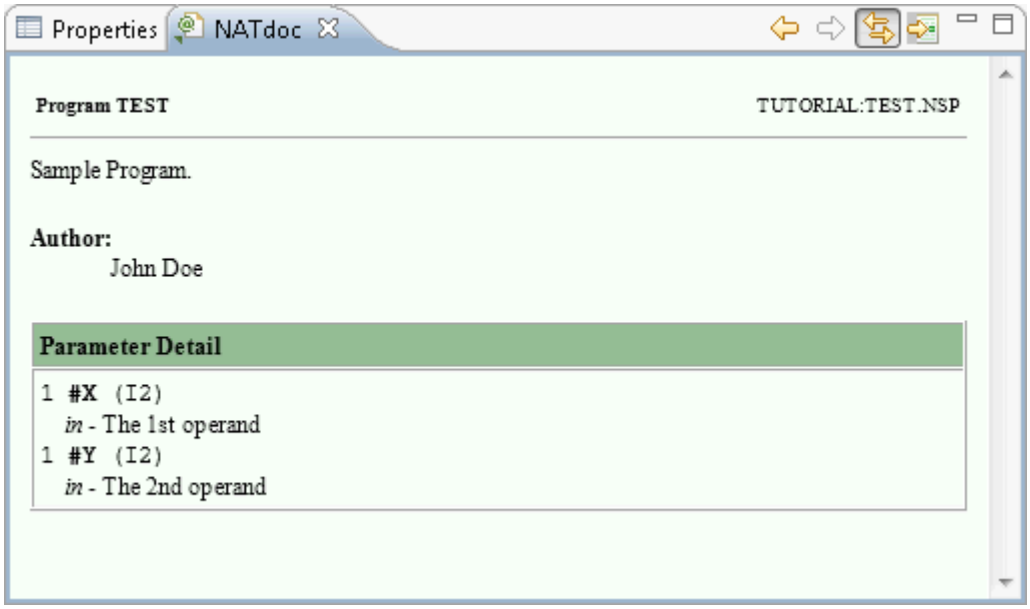
It is recommended that you switch to the Resource perspective, since the NaturalONE perspective does not show all generated files.

When NATdoc has been generated, the folder that you have defined as the destination folder (by default, a folder called *doc* is created) contains all generated files. The top-level node of this folder contains the file *index.html*. This is the starting page for the generated documentation. If you want, you can open it with the web browser.

Natural sources can have names which violate the naming conventions for normal URLs. In the generated files, special characters such as the hash (#) or plus (+) sign are therefore replaced with the name of the character (for example, "hash" or "plus").

Previewing the API Documentation in the NATdoc View

The **NATdoc** view assists you while entering the NATdoc-specific comments in the source code. The output in this view is similar to the generated NATdoc. However, since this output is generated on the fly, links to other documents may differ from those in the wizard-generated NATdoc.



Note: This view is not shown by default when you open the NaturalONE perspective. For information on how to display it, see *Showing a View of the NaturalONE Perspective*.

The local toolbar of the **NATdoc** view provides the following icons:

Icon	Description
	Go back to the previous object.
	Go forward to the next object.
	When selected, each object that you select in the **Navigator** view or **Natural Navigator** view or that you open in the source editor is automatically used in the **NATdoc** view.
	Open the corresponding Natural object in the source editor.

Documentation Comments in the Source Code

The characters `/**` begin a documentation comment. All text until the end of the line belongs to the comment. The text in a comment can continue onto multiple lines.

In your source code, the documentation comments may appear in the following places:

- Directly after the source header and before the first statement. Example:

```
* >Natural Source Header 000000
* :Mode S
* :CP windows-1252
* <Natural Source Header
/** Subroutine 01 from LIB04.
/** <h2>A Subroutine</h2>
/** :author John Doe
DEFINE SUBROUTINE Subroutine_01
/* TODO: Enter your code here
IGNORE
END-SUBROUTINE
```

  Note: The first sentence must always end with a period. It is used in an overview table.

- After the following statements:

```
DEFINE DATA GLOBAL
DEFINE DATA PARAMETER
```

  Example:

```
DEFINE DATA GLOBAL
/** Global Data Area 01 from LIB04.
/** <h2>A Local Data Global</h2>
/** :author John Doe
1 GLOBAL1(A10)
END-DEFINE
```

- After the declaration of `PARAMETER`, `GLOBAL` or `INDEPENDENT` variables in a `DEFINE DATA` block. Example:

```
* >Natural Source Header 000000
* :Mode S
* :CP windows-1252
* <Natural Source Header
/** Subprogram 01 from LIB07.
/** <h2>A Subprogram</h2>
/** :author John Doe
DEFINE DATA
GLOBAL USING MYGDA
PARAMETER
1 ACTION (A1)       /** :in what to do
1 PARMS (A36)       /** :in action-dependent
1 REDEFINE PARMS    /** :redef if action is 'a'
  2 ERROR (A4)      /** :out error number
1 REDEFINE PARMS    /** :redef if action is 'b'
  2 ERROR2 (A4)     /** :in error number
  2 ERRORTEXT (A32) /** :out error number
INDEPENDENT
1 +TRACE (L)        /** :in trace flag
END-DEFINE
WRITE ACTION
END
```

In addition, NATdoc searches each regular comment line which starts with `/*` and checks whether it contains a documentation comment (`/**`). This allows you, for example, to use different comments for IDL generation and for NATdoc. Examples:

- The following comments will be used for NATdoc generation:

```
...
1 MYPARM (A10) /* inout /** :out delivers the new parameter
...
/* a comment /** :since 02/25
FOR #I = 1 TO 10 /* a comment /** :since 02/25
```

- The following comments will *not* be used for NATdoc generation:

```
...
* 1 MYPARM (A10) /* inout /** :out delivers the new parameter
...
** a comment /** :since 02/25
* a comment /** :since 02/25
```

Documentation comments within the body of a program are ignored.

You can use HTML tags in your comments (for example, `<b>` which is used for bold text). The standard template of NATdoc generates HTML 4.0-compliant code.

Leading spaces in a documentation comment (that is blanks and tabs preceding the first character) are not removed.

Even though Natural allows writing programs in reporting mode, NATdoc provides no specific support for this programming mode. It is recommended that you write your programs in structured mode.

Special Comment Files

NATdoc makes use of the following special comment files:

- Overview Comment File
- Library Comment File

Overview Comment File

Any documentation that applies to the entire application or a set of libraries can be written to the overview comment file, which is a normal HTML file.

The overview comment file can have any name. You specify this name in the wizard when you are about the generate NATdoc. The overview comment file must be part of the project to which it pertains. It is recommended that you put it into the root of the project.

When you generate NATdoc and the corresponding option has been enabled in the wizard, the information from the `<body>` of the overview comment file is merged into the overview summary page.

Example:

```
<html>
<body>
<p>This is the text for the overview.</p>
</body>
</html>
```

Library Comment File

Any documentation that applies to the entire library can be written to a library comment file, which is a normal HTML file.

A library comment file must always have the name *library.html*, and it must be placed into the root of the library to which it pertains.

When you generate NATdoc and a file with the above name can be found in the library root, the information from the `<body>` of the library comment file is merged into the library summary page.

Example:

```
<html>
<body>
This is the 4th test library.
<p><i>Note:</i><br>
It's only a test.</p>
</body>
</html>
```

Note: The first sentence must always end with a period. It is used in an overview table. Do not put a title or any other text between `<body>` and this first sentence.

Overview of NATdoc Tags

Documentation comments may contain tags. They are case-sensitive, that is, a tag must be written as indicated below.

NATdoc distinguishes the following types of tags:

- Field Tags
- Inline Tags

Field Tags

A field tag starts with a colon (:). It is only possible to define a single field tag in each comment line.

`:author` *`name`*

Adds the string "Author" with the specified name to the generated document.

Example:

```
/** :author John Doe
```

`:deprecated` *`text`*

Indicates that the API should no longer be used. Your text appears in italics, in front of the main description. It is preceded by the word "Deprecated" in bold. The first sentence of your text also appears in the summary and index.

Example:

```
/** :deprecated This function has been replaced by <:link SYSEXT/USR2036P>.
```

`:in` *`text`*, `:out` *`text`*, `:inout` *`text`*

`:in` pertains to a parameter that is used as input only.

`:out` pertains to a parameter that is used as output only.

`:inout` pertains to a parameter that is used as both input and output.

These tags can be used in two different places.

- After the declaration of a parameter in a `DEFINE DATA PARAMETER` block. In this case, the syntax is as follows:

  ```
  tag description
  ```

  Example:

  ```
  1 ch    (A1)        /** :in the character to be tested
  1 field (A) dynamic /** :in the string to be converted
  ```

- In front of a `DEFINE SUBROUTINE` statement. In this case, the syntax is as follows:

  ```
  tag parameter-name description
  ```

  Example:

  ```
  /** :in ch the character to be tested
  /** :in field the string to be converted
  DEFINE SUBROUTINE TRANSLATE
  ```

`:redef` *`text`*

Pertains to a parameter that is redefined and where the calling program has to fill the data according to the structure.

This tag can be used in the same places as described above for the `:in`, `:out` and `:inout` tags.

`:return` *`text`*

Adds a section which contains the heading "Returns" and your text. This can only be used with a function.

Example:

```
/** :return all in one.
DEFINE FUNCTION Function_01 RETURNS (A) DYNAMIC
```

`:see` *`reference`*

Adds a section which contains the heading "See Also" and your reference. All `:see` tags in a documentation comment are listed under the same heading.

This tag can be used in different forms:

- `:see "`*`string`*`"`

  A link is not generated. Example:

  ```
  /** :see "The NaturalONE Documentation"
  ```

- `:see <a href="`*`url#value`*`">`*`label`*`</a>`

  A link is generated. You specify either a relative or absolute URL. Example:

  ```
  /** :see <a href="natlang.html#about">Natural Language</a>
  ```

`:since` *`text`*

Adds a section which contains the heading "Since" and your text. This tag indicates that a feature exists since the release specified in your text. Example:

```
/** :since 4.2
```

`:version` *`text`*

Adds a section which contains the heading "Version" and your text. This tag writes the version number you specify to the generated document. Example:

```
/** :version 8.1
```

Inline Tags

Inline tags are written in angle brackets. The tag itself also starts with a colon (:). It is possible to define more than one inline tag within the same comment line.

`<:docRoot>`

Contains the relative path to the root directory. This is useful if you want to reference a file (such as the copyright page) from all generated pages.

Example:

```
/** See the <a href="<:docRoot>/copyright.html">Software AG copyright page</a>.
```

`<:link library-name/object-name label>`

This is similar to the `:see` tag. Other than `:see`, `<:link>` inserts an inline link in the generated document. This inline link refers to the library and object that you specify. Only your label is visible in the generated document.

Example:

```
/** Use the <:link SYSEXT/USR2036P USR2036P> user exit.
```

The generated page contains the following HTML code:

```
Use the <a href="SYSEXT/USR2036P.html"><code>USR2036P</code></a> user exit.
```

`<:linkplain library-name/object-name label>`

This is the same as the `:link` tag. The only difference is in the generated HTML: the text in the link is not put in `<code>` tags; plain text is used instead.

Example:

```
/** Use the <:linkplain SYSEXT/USR2036P USR2036P> user exit.
```

The generated page contains the following HTML code:

```
Use the <a href="SYSEXT/USR2036P.html">USR2036P</a> user exit.
```

Where Can Tags be Used?

The following table shows where the NATdoc tags can be used:

Tag Name	Natural Object Description	Interface Description	Inline
`:author`	X		
`:deprecated`	X		
`:in`		X	
`:inout`		X	
`:out`		X	
`:redef`		X	
`:return`		X	
`:see`	X		
`:since`	X		
`:version`	X		

Tag Name	Natural Object Description	Interface Description	Inline
`<:docRoot>`			X
`<:link>`			X
`<:linkplain>`			X

Using Custom Templates

Using a custom template, you can adapt the generated NATdoc files to your requirements. The generated pages may be shown, for example, with different colors or in another language. You specify the path to your custom template in the wizard when you are about the generate NATdoc.

The following topics are covered below:

- Location of the Custom Templates
- Directory Structure
- Quick Customizations

Location of the Custom Templates

The following files are located in the *naturalone/samples* directory of your NaturalONE installation:

- ***NATdoc_en.zip***
  Contains all required custom templates in English. You can use these templates for your own modifications or for translations to other languages.
- ***NATdoc_de.zip***
  Contains a German translation of the English custom templates.

If you want to use the custom templates from one of the above Zip files, you have to unzip the file. You can unzip it into any directory which can be accessed by the wizard.

Directory Structure

When you unzip, for example, the file *NATdoc_de.zip*, you will find the following directory structure in your file system:

Important: Do not delete or rename any file or subdirectory. Do not change the encoding of the files, which is "UTF-8 without BOM".

Quick Customizations

You can quickly change the look-and-feel of the generated pages as described in the following topics:

- Translation into Another Language
- Color Scheme
- Header Comments
- Body Footers

Translation into Another Language

Unzip the file *NATdoc_en.zip*. Rename the resulting directory *NATdoc_en*, for example, to *NATdoc_mylanguage*.

Open the file *readme_language.txt*. This file is located in the top-level directory. It contains a list of all strings that need to be replaced in all NATdoc template files. For example, search for the following string

```
>API Help<
```

and replace it with the appropriate term in the chosen language. Make sure to keep the angle brackets which are part of the enclosing HTML tag.

Color Scheme

To change the colors that are used in the generated pages, modify the corresponding entries in the file *stylesheet.css*. This file is located in the *DOCtemplate* subdirectory.

For example, if you want to change the standard green background colors for the navigation bar which is shown at the top and bottom of the generated pages, modify the entries for `NavBarCell1` and `NavBarCell1Rev`:

```
.NavBarCell1    { background-color:#F88017 } /* dark orange */
.NavBarCell1Rev { background-color:#7E3117 } /* dark orange3 */
```

Header Comments

If you want to include additional comments in the headers of the generated pages, modify the file *natdoc($GENERATEDCOMMENT$).html*. This file is located in the *DOCtemplate* subdirectory.

The entire content of this file is included in the `<head>` section of the generated pages.

```
<html>
<head>
$GENERATEDCOMMENT$
...
</head>
```

Body Footers

If you want to include an additional footer in the generated pages, modify the file *natdoc($FOOTER$).html*. This file is located in the *DOCtemplate* subdirectory.

The entire content of this file is included at the bottom of the `<body>` section of the generated pages.

```
...
$FOOTER$
</body>
</html>
```

Detailed Template Descriptions

All templates are based on HTML. Each template contains HTML tags, data and placeholder variables.

The following topics are covered below:

- Placeholder Variables
- Template Files for NATdoc Tags
- Template Files for Generated HTML Pages
- Call Hierarchies and Placeholders in the Template Files

- Special Generation Options

Placeholder Variables

During the generation of NATdoc, a placeholder variable can be replaced either with a single value or with the content of a template which may contain further placeholder variables. Placeholder variables are always surrounded by dollar ($) signs and are case-sensitive. Examples:

```
$DO_VALUE$
$NEXTPAGE$
```

The naming conventions for the different types of placeholder variables are described below.

- **Single Value**
  Placeholder variables starting with `$DO_` or `$IN_` followed by an uppercase name are used to insert a single value. Examples:

Placeholder Variable	Description
`$DO_LEVEL$`	Level of a `DEFINE DATA` parameter.
`$DO_PARAMETER$`	Name of a `DEFINE DATA` parameter.
`$DO_FIRSTSENTENCE$`	First sentence in a documentation comment.
`$IN_OBJECTTYPE$`	Name of the object type for processing.
`$IN_OBJECTTYPENAME$`	Name of the object type for output.

- **Content of a Template**
  Placeholder variables with other uppercase names are replaced with the content of a template. Examples:

Placeholder Variable	Description
`$SUMMARY(HELPROUTINE)$`	Add summary data for help routines. The corresponding template *natdoc-natlibrary-summary($SUMMARY($OBJECTTYPE$)$).html* is used.
`$PREVPAGE$`	Add template for linking to the previous page, if existing.
`$SUMMARYLINE$`	Add summary lines. The corresponding template *natdoc-natlibrary-frame($SUMMARYLINE$).html* is used.

- **NATdoc Tag Template**
  Placeholder variables with lowercase names are used to insert NATdoc tag templates. Examples:

Placeholder Variable	Description
`$in$`	Name of a field tag in a template.
`$detail$`	Include template for `DEFINE DATA` field tags.

Template Files for NATdoc Tags

NATdoc tags are used in the documentation comments which are contained in the Natural source files. The *DOCtags* subdirectory contains a template file for each NATdoc tag.

The following topics are covered below:

- Template Files for NATdoc Field Tags
- Template Files for NATdoc Inline Tags
- Template Files for Summaries

Template Files for NATdoc Field Tags

NATdoc defines a set of field tags, but can also handle unknown field tags that obey the field tag syntax. There are different categories of field tags:

- **Field tags that generate output for each occurrence**
  This applies to the following tags:

  ```
  :in
  :out
  :inout
  :redef
  :return
  ```

  A single template is defined for each of these tags. The naming convention for such a template is *NATtag-<tagname>.html* where *tagname* is the lowercase name of the tag. For example, *NATtag-inout.html*.

- **Field tags that generate combined output over all occurrences**
  This applies to the following tags:

  ```
  :author
  :deprecated
  :see
  :since
  :version
  ```

  Two templates are defined for each of these tags.

The naming convention for the first template is *NATtag-<tagname>.html* where *tagname* is the lowercase name of the tag. For example, *NATtag-author.html*. The first template must contain the `$NEXT$` placeholder variable.

The naming convention for the second template is *NATtag-<tagname>_next.html* where *tagname* is the lowercase name of the tag. The string "_next" must also be written in lowercase. For example, *NATtag-author_next.html*

- **Unknown field tags that generate combined output over all occurrences**
  For any unknown field tags which are not listed above, it is either possible not to generate anything, or to generate generic output. Two templates are available for the latter case: *NATtag-notag.html* and *NATtag-notag_next.html*.

Template Files for NATdoc Inline Tags

This applies to the following tags:

```
<:link>
<:linkplain>
```

A single template is defined for each of these tags. The naming convention for such a template is *NATtag-<tagname>.html* where *tagname* is the lowercase name of the tag. For example, *NATtag-linkplain.html*.

Note: `<:docroot>` is a special inline tag for which a template is not required.

Template Files for Summaries

In addition to the above mentioned template files for the different NATdoc tags, the *DOCtags* subdirectory also contains the following special template files which do not apply to a special tag:

- *NATtag_summary-docu.html* for the description of the source.
- *NATtag_summary-detail.html* for the description of a single parameter in the parameter area.

These special templates are used by the *natdoc-natsource.html* template. They are included in each page that is generated for a Natural source.

Template Files for Generated HTML Pages

The template files for the generated HTML pages are located in the *DOCtemplate* subdirectory. During the generation of NATdoc, one or more template files are used to generate an HTML page.

There are two types of templates with different naming conventions:

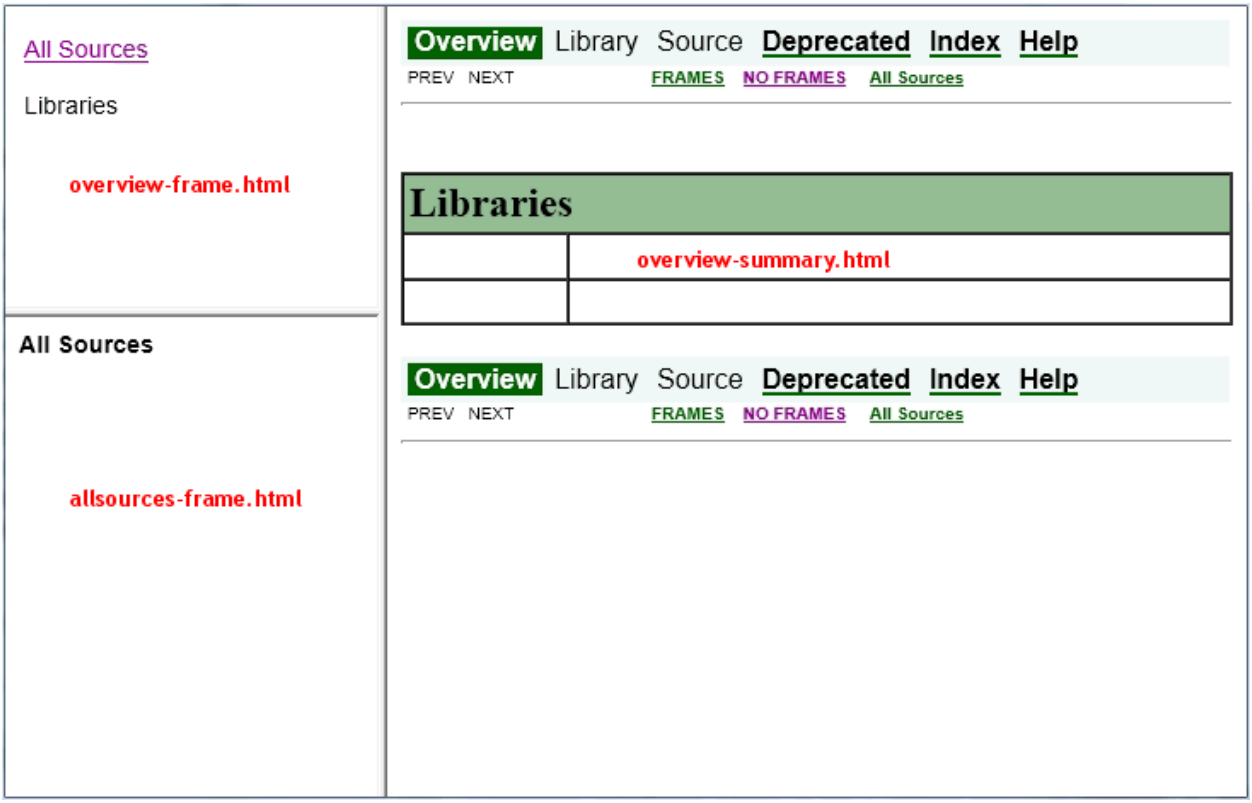
- **Main Templates**
  The name of a main template does not contain any placeholder variables. For example:

  natdoc-index-all.html

- **Subtemplates**
  The name of a subtemplate starts with the name of the main template, followed by a placeholder variable that is enclosed in parentheses and dollar ($) signs. For example, these are the names of the subtemplates that are used by the *natdoc-index-all.html* main template:

  natdoc-index-all($SOURCEINDEXALL$).html
  natdoc-index-all($LETTERBAR$).html
  natdoc-index-all($INDEXLINE$).html

The following graphic shows the names of the frames that are used in the generated *index.html* file. The name of the template for this file is *natdoc-index.html*.



The following topics are covered below:

- Contents of the DOCtemplate Subdirectory
- Contents of the DOCtemplate/index-files Subdirectory
- Contents of the DOCtemplate/library Subdirectory

Contents of the DOCtemplate Subdirectory

The following table lists the template files that are located in the root of the *DOCtemplate* subdirectory and the names of the generated HTML files. When subtemplates are used, the main template (which does not contain a placeholder variable) is always listed at the top.

Template File	Generated HTML File	Description
natdoc-index.html	*index.html*	Main page containing the frames mentioned below.
natdoc-overview-summary.html *natdoc-overview-summary($SUMMARYLINE$).html*	*overview-summary.html*	Overview page for all libraries. Right frame of the *index.html* page. Or main file if frames are not used.
natdoc-overview-frame.html	*overview-frame.html*	Overview page for all libraries with the link "All Sources". Left upper frame of the *index.html* page.
natdoc-allsources-frame.html *natdoc-allsources-frame($SUMMARYLINE$).html*	*allsources-frame.html*	Overview page for all source files if frames are used. Left lower frame of the *index.html* page.
natdoc-allsources-noframe.html *natdoc-allsources-noframe($SUMMARYLINE$).html*	*allsources-noframe.html*	Overview page for all source files if frames are not used.
natdoc-index-all.html *natdoc-index-all($SOURCEINDEXALL$).html* *natdoc-index-all($LETTERBAR$).html* *natdoc-index-all($INDEXLINE$).html*	*index-all.html*	Only used if separate index pages are not generated for each letter. In this case, this file contains the entire index.
natdoc-help.html	*help-doc.html*	Help page.
natdoc-deprecated-list.html	*deprecated-list.html*	List of deprecated APIs.

In addition to the files listed in the table above, the *DOCtemplate* subdirectory also contains the following files:

File Name	Description
stylesheet.css	During the generation, this file is simply copied. It is not modified.
natdoc($GENERATEDCOMMENT$).html	These files contain general replacements which are applied to all generated HTML pages.
natdoc($FOOTER$).html	

Contents of the DOCtemplate/index-files Subdirectory

When separate index pages are generated for each letter, a new subdirectory with the name *index-files* is generated. Each HTML file in this subdirectory is generated with the template files from the *DOCtemplate/index-files* subdirectory. These template files are listed in the following table. The main template (which does not contain a placeholder variable) is always listed at the top.

Template File	Generated HTML File	Description
natdoc-index.html *natdoc-index($INDEXLINE$).html* *natdoc-index($LETTERBAR$).html* *natdoc-index($NEXTPAGE$).html* *natdoc-index($NONEXTPAGE$).html* *natdoc-index($NOPREVPAGE$).html* *natdoc-index($PREVPAGE$).html*	*index-<n>.html*	Only used if separate index pages are generated for each letter. *n* stands for a sequential number within the generated file name. For example, *index-0.html*, *index-1.html* and so on.

Contents of the DOCtemplate/library Subdirectory

For each library, a subdirectory is generated which is named after the library.

The following table lists the template files that are located in the *DOCtemplate/library* subdirectory and the names of the generated HTML files. When subtemplates are used, the main template (which does not contain a placeholder variable) is always listed at the top.

Template File	Generated HTML File	Description
natdoc-natlibrary-frame.html *natdoc-natlibrary-frame($SUMMARY($IN_OBJECTTYPE$)$).html* *natdoc-natlibrary-frame($SUMMARYLINE$).html*	*library-frame.html*	Summary for this library. Left lower frame.
natdoc-natlibrary-summary.html *natdoc-natlibrary-summary($NEXTPAGE$).html* *natdoc-natlibrary-summary($NONEXTPAGE$).html* *natdoc-natlibrary-summary($NOPREVPAGE$).html* *natdoc-natlibrary-summary($PREVPAGE$).html* *natdoc-natlibrary-summary($SUMMARY($IN_OBJECTTYPE$)$).html* *natdoc-natlibrary-summary($SUMMARYLINE$).html*	*library-summary.html*	Summary of all source files belonging to this library. Right frame. Or main file if frames are not used.
natdoc-natsource.html *natdoc-natsource($DETAIL($TYPE$)$).html* *natdoc-natsource($DETAILLINE$).html*	*<sourcename>.html*	API description for a single Natural source

Template File	Generated HTML File	Description
natdoc-natsource($NEXTPAGE$).html *natdoc-natsource($NONEXTPAGE$).html* *natdoc-natsource($NOPREVPAGE$).html* *natdoc-natsource($PREVPAGE$).html*		file. `sourcename` stands for the name of the related Natural source file. For example, *MYMAP.html*.

Call Hierarchies and Placeholders in the Template Files

The template files use placeholder variables for single values which are replaced during NATdoc generation.

The template files can also use subtemplates, just like a Natural program can include copycodes. However, other than with Natural programs, subtemplates can be included recursively. For example, the main template *natdoc-overview-summary.html* uses the placeholder variable `$SUMMARYLINE$` which is defined by the subtemplate *natdoc-overview-summary($SUMMARYLINE$).html*. This subtemplate includes itself by also using `$SUMMARYLINE$`.

This section describes the call hierarchies of the templates and the placeholder variables for single values. The content in the topics below is arranged by the names of the generated HTML pages. The following topics are covered below:

- index.html
- overview-summary.html
- overview-frame.html
- allsources-noframe.html
- allsources-frame.html
- index-all.html
- help-doc.html
- deprecated-list.html
- <libraryname>/library-frame.html
- <libraryname>/library-summary.html
- <libraryname>/<sourcename>.html

- index-files/index-<n>.html

index.html

The following template is used for generating this file:

```
natdoc-index.html
```

This template file contains the following placeholder variables:

Placeholder Variable	Description
$DO_NATDOCVERSION$	The NATdoc version that is used for generation.
$DO_TIMENOW$	Time stamp that is used during generation.

overview-summary.html

The following template hierarchy is used for generating this file:

```
natdoc-overview-summary.html
     natdoc-overview-summary($SUMMARYLINE$).html
```

These template files contain the following placeholder variables:

Placeholder Variable	Description
$DO_DESCRIPTIONFIRSTLINE$	First sentence in a documentation comment.
$DO_LIBRARYNAME$	Name of a Natural library.
$DO_LIBRARYNAMEHTML$	Link to a Natural library.
$DO_MENUDEPRECATED$	Link to deprecated page, if existing.
$DO_MENUINDEX$	Add index menu.
$DO_PROJECTDESCRIPTIONTAGS$	Content of overview file.
$DO_PROJECTNAME$	Project name as specified by the *.project* file.

overview-frame.html

The following template hierarchy is used for generating this file:

```
natdoc-overview-frame.html
     natdoc-overview-frame($SUMMARYLINE$).html
```

These template files contain the following placeholder variables:

Placeholder Variable	Description
$DO_LIBRARYNAME$	Name of a Natural library.
$DO_LIBRARYNAMEHTML$	Link to a Natural library.

allsources-noframe.html

The following template hierarchy is used for generating this file:

```
natdoc-allsources-noframe.html
     natdoc-allsources-noframe($SUMMARYLINE$).html
```

These template files contain the following placeholder variables:

Placeholder Variable	Description
$DO_INLINESUBROUTINELINK$	Link to a Natural subroutine.
$DO_INLINESUBROUTINENAME$	Name of a Natural subroutine.
$DO_LIBRARYNAME$	Name of a Natural library.
$DO_LIBRARYNAMEHTML$	Link to a Natural library.
$DO_SOURCENAME$	Name of a Natural source.
$DO_SOURCENAMEHTML$	Link to a Natural source.

allsources-frame.html

The following template hierarchy is used for generating this file:

```
natdoc-allsources-frame.html
     natdoc-allsources-frame($SUMMARYLINE$).html
```

These template files contain the following placeholder variables:

Placeholder Variable	Description
$DO_INLINESUBROUTINELINK$	Link to a Natural subroutine.
$DO_INLINESUBROUTINENAME$	Name of a Natural subroutine.
$DO_LIBRARYNAME$	Name of a Natural library.
$DO_LIBRARYNAMEHTML$	Link to a Natural library.
$DO_SOURCENAME$	Name of a Natural source.
$DO_SOURCENAMEHTML$	Link to a Natural source.

index-all.html

The following template hierarchy is used for generating this file:

```
natdoc-index-all.html
     natdoc-index-all($LETTERBAR$).html
     natdoc-index-all($SOURCEINDEXALL$).html
          natdoc-index-all($INDEXLINE$).html
```

These template files contain the following placeholder variables:

Placeholder Variable	Description
$DO_DESCRIPTIONFIRSTLINE$	First sentence in a documentation comment.
$DO_INDEXCHAR$	Current index letter.
$DO_LIBRARYNAMEHTML$	Link to a Natural library.
$DO_LOCATION$	Link to the current object.
$DO_MENUDEPRECATED$	Link to deprecated page, if existing.
$DO_MENUINDEX$	Link to index page.
DO_NAME	Name of the current object.
$DO_SOURCENAMEHTML$	Link to a Natural source.
$DO_INLINESUBROUTINELINK$	Link to a Natural subroutine.
$DO_INLINESUBROUTINENAME$	Name of a Natural subroutine.

help-doc.html

The following template is used for generating this file:

```
natdoc-help.html
```

This template file contains the following placeholder variables:

Placeholder Variable	Description
$DO_MENUDEPRECATED$	Link to deprecated page, if existing.
$DO_MENUINDEX$	Link to index page.
DO_INDEX	For a single index, *index-all.html* is generated. When separate index pages are generated for each letter, the *index-files* subdirectory is generated containing the files *index-0.html*, *index-1.html* and so on.

deprecated-list.html

The following template hierarchy is used for generating this file:

```
natdoc-deprecated-list.html
     natdoc-deprecated-list($SUMMARY($OBJECTTYPE$)$).html
          natdoc-deprecated-list($SUMMARYLINE$).html
```

These template files contain the following placeholder variables:

Placeholder Variable	Description
$DO_DESCRIPTIONFIRSTLINE$	First sentence in a documentation comment.
$DO_INLINESUBROUTINELINK$	Link to a Natural subroutine.
$DO_INLINESUBROUTINENAME$	Name of a Natural subroutine.
$DO_LIBRARYNAME$	Name of a Natural library.
$DO_LIBRARYNAMEHTML$	Link to a Natural library.
$DO_SOURCENAME$	Name of a Natural source.
$DO_SOURCENAMEHTML$	Link to a Natural source.
$IN_OBJECTTYPE$	Name of the object type for processing.
$IN_OBJECTTYPENAME$	Name of the object type for output.

<libraryname>/library-frame.html

The following template hierarchy is used for generating this file:

```
library/natdoc-natlibrary-frame.html
     library/natdoc-natlibrary-frame($SUMMARY($OBJECTTYPE$)$).html
           library/natdoc-natlibrary-frame($SUMMARYLINE$).html
```

These template files contain the following placeholder variables:

Placeholder Variable	Description
$DO_INLINESUBROUTINELINK$	Link to a Natural subroutine.
$DO_INLINESUBROUTINENAME$	Name of a Natural subroutine.
$DO_LIBRARYNAME$	Name of a Natural library.
$DO_LIBRARYNAMEHTML$	Link to a Natural library.
$DO_SOURCENAME$	Name of a Natural source.
$DO_SOURCENAMEHTML$	Link to a Natural source.
$IN_OBJECTTYPE$	Name of the object type for processing.
$IN_OBJECTTYPENAME$	Name of the object type for output.

<libraryname>/library-summary.html

The following template hierarchy is used for generating this file:

```
library/natdoc-natlibrary-summary.html
     library/natdoc-natlibrary-summary($PREVPAGE$).html
     library/natdoc-natlibrary-summary($NOPREVPAGE$).html
     library/natdoc-natlibrary-summary($NONEXTPAGE$).html
     library/natdoc-natlibrary-summary($NEXTPAGE$).html
     library/natdoc-natlibrary-summary($SUMMARY($OBJECTTYPE$)$).html
           library/natdoc-natlibrary-summary($SUMMARYLINE$).html
```

These template files contain the following placeholder variables:

Placeholder Variable	Description
$DO_DESCRIPTIONFIRSTLINE$	First sentence in a documentation comment.
$DO_INLINESUBROUTINELINK$	Link to a Natural subroutine.
$DO_INLINESUBROUTINENAME$	Name of a Natural subroutine.
$DO_LIBRARYDESCRIPTIONTAGS$	Content of the library comment file *library.html*. See also ***Special Comment Files***.
$DO_LIBRARYNAME$	Name of a Natural library.
$DO_LIBRARYNAMEHTML$	Link to a Natural library.
$DO_MENUDEPRECATED$	Link to deprecated page, if existing.
$DO_MENUINDEX$	Link to index page.
$DO_NEXTHTML$	Link to the next library.
$DO_PREVHTML$	Link to the previous library.
$DO_SOURCENAME$	Name of a Natural source.
$DO_SOURCENAMEHTML$	Link to a Natural source.
$IN_OBJECTTYPE$	Name of the object type for processing.
$IN_OBJECTTYPENAME$	Name of the object type for output.

<libraryname>/<sourcename>.html

The following template hierarchy is used for generating this file:

```
library/natdoc-natsource.html
     library/natdoc-natsource($PREVPAGE$).html
     library/natdoc-natsource($NEXTPAGE$).html
     library/natdoc-natsource($NOPREVPAGE$).html
     library/natdoc-natsource($NONEXTPAGE$).html
     library/natdoc-natsource($DETAIL($TYPE$)$).html
           library/natdoc-natsource($DETAILLINE$).html
```

These template files contain the following placeholder variables:

Placeholder Variable	Description
$DO_DETAILS$	Detailed parameter description.
$DO_LIBRARYNAME$	Name of a Natural library.
$DO_LIBRARYNAMEHTML$	Link to a Natural library.
$DO_MENUDEPRECATED$	Link to deprecated page, if existing.
$DO_MENUINDEX$	Link to index page.
$DO_NEXTHTML$	Link to the next source page.
$DO_PREVHTML$	Link to the previous source page.
$DO_SOURCEDESCRIPTIONTAGS$	Add NATdoc tags.
$DO_SOURCEFILENAME$	The long name of the source file.
$DO_SOURCENAME$	Name of a Natural source.
$DO_SOURCENAMEHTML$	Link to a Natural source.
$DO_SOURCETYPE$	Natural object type.
IN_TYPE	Type of data area (for example, "Local Data Area").

index-files/index-<n>.html

The following template hierarchy is used for generating this file:

```
index-files/natdoc-index.html
     index-files/natdoc-index($PREVPAGE$).html
     index-files/natdoc-index($NEXTPAGE$).html
     index-files/natdoc-index($NOPREVPAGE$).html
     index-files/natdoc-index($NONEXTPAGE$).html
     index-files/natdoc-index($LETTERBAR$).html
     index-files/natdoc-index($INDEXLINE$).html
```

These template files contain the following placeholder variables:

Placeholder Variable	Description
DO_INDEX	For a single index, *index-all.html* is generated. When separate index pages are generated for each letter, the *index-files* subdirectory is generated containing the files *index-0.html*, *index-1.html* and so on.
$DO_INDEXCHAR$	Current index letter.
$DO_INDEXNUMBER$	Total number of index files.
$DO_MENUDEPRECATED$	Link to deprecated page, if existing.
$DO_NEXTINDEX$	Link to the next index page, if existing.
$DO_PREVINDEX$	Link to the previous index page, if existing.
$DO_LIBRARYNAMEHTML$	Link to a Natural library.
$DO_SOURCENAMEHTML$	Link to a Natural source.
$DO_INLINESUBROUTINELINK$	Link to a Natural subroutine.

Placeholder Variable	Description
`$DO_NAME$`	Name of the current object.
`$DO_LOCATION$`	Link to the current object.
`$DO_DESCRIPTIONFIRSTLINE$`	First sentence in a documentation comment.

Special Generation Options

The following topics are covered below:

- Index Link in the Navigation Menu
- Deprecated Link in the Navigation Menu
- Global Replacements with Template Files
- Global Replacements without Template Files
- Stylesheet

Index Link in the Navigation Menu

The following template files are used for generating the **Index** link in the navigation menu:

```
natdoc-menu-index.html
library/natdoc-menu-index.html
```

These template files contain the following placeholder variables:

Placeholder Variable	Description
`$DO_INDEX$`	For a single index, *index-all.html* is generated. When separate index pages are generated for each letter, the *index-files* subdirectory is generated containing the files *index-0.html*, *index-1.html* and so on.

Deprecated Link in the Navigation Menu

The following template files are used for generating the **Deprecated** link in the navigation menu:

```
natdoc-menu-deprecated.html
library/natdoc-menu-deprecated.html
```

These template files do not contain any placeholder variables.

Global Replacements with Template Files

The following template files can be used to add additional comments to the `<header>` sections of the generated pages to an add individual footer to the `<body>` sections of the generated pages:

```
natdoc($GENERATEDCOMMENT$).html
natdoc($FOOTER$).html
```

See also ***Header Comments*** and ***Body Footers***.

Global Replacements without Template Files

Template files are not necessary for global replacements using the following placeholder variables:

Placeholder Variable	Description
`$DO_INDEX$`	For a single index, *index-all.html* is generated. When separate index pages are generated for each letter, the *index-files* subdirectory is generated containing the files *index-0.html*, *index-1.html* and so on.
`$DO_NATDOCVERSION$`	The NATdoc version that is used for generation.
`$DO_TIMENOW$`	Time stamp that is used during generation.

Stylesheet

The following file defines colors, fonts and other style attributes for the generated pages:

```
stylesheet.css
```

If you want to change the default settings that are applied to the generated pages, modify the corresponding entries in this file.

IX Checking Natural Code with NATstyle

24 Checking Natural Code with NATstyle

- General Information 364
- Checking the Natural Code 364
- Clearing the NATstyle Violations 366
- Working with Result Files 366
- Invoking NATstyle from Outside Eclipse 367
- Overview of NATstyle Rules, Error Messages and Solutions 369
- DTDs Used by NATstyle 378

General Information

NATstyle helps you write Natural code which adheres to your coding standards. It can check the source code of all Natural object types.

Restrictions:

- NATstyle only checks the Natural objects in the Eclipse workspace. It does not check the objects on a Natural server.
- NATstyle does not download missing sources from a Natural server when checking code.
- NATstyle checks only the selected objects. However, if the **Parser check** option is active in the NATstyle preferences, any objects called using the `INCLUDE` and `USING` statements are also checked, if available in the Eclipse workspace.
- NATstyle uses only the settings from the *.natural* file (for example, the `DC` character or steplib settings). It does not download any settings (for example, Natural Security settings) from a Natural server.

Checking the Natural Code

NATstyle uses the settings in the Natural preferences in order to check your Natural code. You can either use the default settings or you can create your own configuration file(s) containing your own rules for checking the Natural code. For further information, see ***NATstyle*** in *Setting the Preferences*.

To check the Natural code

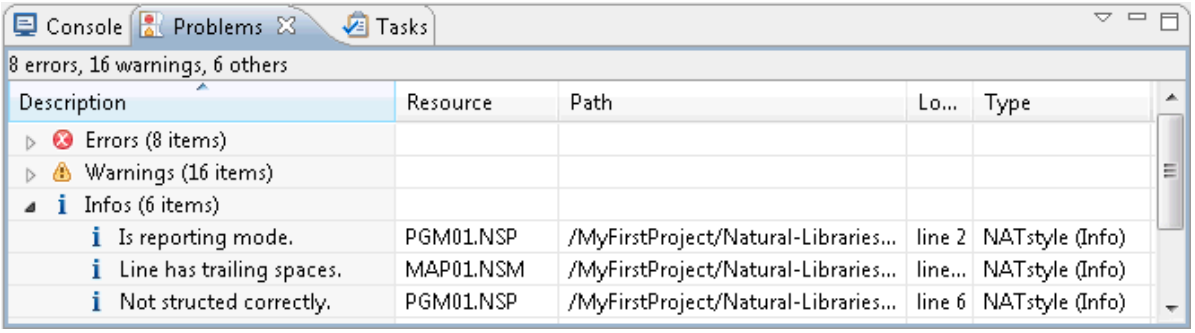
1. In the **Navigator** view or in the **Natural Navigator** view, select the project, library or source file for which you want to check the Natural code.
2. Invoke the context menu and choose **NaturalONE > NATstyle Check**.

   Or:

   Use the keyboard shortcut CTRL+SHIFT+C.

   If you check a large number of files, the check may take a while.

   Any NATstyle violations (that is, found errors, warnings and information messages) are shown in the **Problems** view. If you want to view, for example, the information messages, you expand the **Infos** node.



3 Double-click an entry in the **Problems** view.

The Natural editor for this type of object is invoked and the corresponding line is selected. One of the following NATstyle markers is shown for each problematic line:

Marker	Type
	Error
	Warning
i	Information

In addition to the information in the **Problems** view and the NATstyle markers in the editors, the icon of each node in the **Navigator** view or **Natural Navigator** view for which a NATstyle violation has been reported contains a corresponding decoration.

4 Change the code as appropriate.

Notes:

1. The error message "No NATstyle results, parser ended" means that the checked Natural object contains a parser error. Before it is possible to check a Natural object with NATstyle, you have to correct all parser errors.
2. The **source header** cannot be modified. When a line in the source header is selected, specify the appropriate information (for example, the programming mode) in the properties for that object.
3. The line number given for a map can be found on the **List** page of the map editor.
4. For DDMs, only general messages are provided which always refer to line 1.
5. There are messages for which an editor cannot be invoked, for example, "NATdoc library documentation (library.html) missing" which refers to the contents of a library.

Clearing the NATstyle Violations

When you correct your sources, any previously detected NATstyle violations are not automatically removed. This includes the entries in the **Problems** view, the NATstyle markers in the editors, and the decorations in the **Navigator** view or **Natural Navigator** view. If you want to remove the NATstyle violations, for example, for a specific library, you can clear them as described below.

Note: The **NATstyle Check** command automatically updates the existing NATstyle markers in the editors, the decorations in the **Navigator** view, and the entries in the **Problems** view.

To clear the NATstyle violations

1. In the **Navigator** view or in the **Natural Navigator** view, select the project, library or source file for which you want to clear the NATstyle violations.
2. Invoke the context menu and choose **NaturalONE > NATstyle Clear**.

   Or:

   Use the keyboard shortcut CTRL+SHIFT+D.

Working with Result Files

The following topics are covered below:

- Saving the Result of a NATstyle Check to a File
- Loading a NATstyle Result File

Saving the Result of a NATstyle Check to a File

You can write the result of a NATstyle check to an XML file. The result always applies to the entire project. In this case, no information is written to the **Problems** view, no decorations are applied in the **Navigator** view or **Natural Navigator** view, and no NATstyle markers are set in the editors.

When you save the results to the XML file, all Natural object types as specified in the **NATstyle** preferences are checked. Any previously generated NATstyle information in the **Problems** view is disregarded. Therefore, it is not necessary to invoke the **NATstyle Check** command prior to saving the results.

Note: The **NATstyle Save** command does not change or remove any already existing information in the **Problems** view, decorations in the **Navigator** view or **Natural Navigator** view, or NATstyle markers in the editors.

To save the result of a NATstyle check to a file

1 In the **Navigator** view or in the **Natural Navigator** view, select the project for which you want to save the results.

2 Invoke the context menu and choose **NaturalONE > NATstyle Save**.

   A dialog box appears, providing the file name *NATstyleResult.xml* as a proposal.

3 Specify a file name.

   It is recommended that you store the result file in the project for which it is created. This is helpful, if you later want to load the result file for this project.

4 Choose the **Save** button to create the NATstyle result file.

Loading a NATstyle Result File

You can load a previously saved NATstyle result file. This updates the existing NATstyle information in the **Problems** view, the decorations in the **Navigator** view or **Natural Navigator** view, and the NATstyle markers in any previously opened editors.

To load a NATstyle result file

1 In the **Navigator** view or in the **Natural Navigator** view, select the project for which you want to load the result file.

2 Invoke the context menu and choose **NaturalONE > NATstyle Load**.

   A dialog box appears showing the root directory of the selected project.

3 Select the result file which has been created for this project.

4 Choose the **Open** button.

Invoking NATstyle from Outside Eclipse

You can invoke NATstyle without having to invoke Eclipse. To do so, you invoke the class `com.softwareag.naturalone.natural.natstyle` with certain command line parameters.

For example, when you invoke NATstyle from within a batch file, the command line parameters can be specified as follows:

```
"%JAVAPATH%\bin\java.exe" -cp %CLASSPATH%
com.softwareag.naturalone.natural.natstyle.NATstyle -projectpath %PROJECTPATH%
-rootfolder -o %OUT%\NATstyleResult.xml >%OUT%\NATstyleGeneration.log
```

Note: Make sure to specify the above information in a single line.

It is important that the following classes can be found in your class path:

```
com.softwareag.naturalone.natural.auxiliary
com.softwareag.naturalone.natural.common
com.softwareag.naturalone.natural.parser
```

The following command line parameters are available:

Command Line Parameter	Description
`-projectpath` *`directory`*	Path to the directory which contains the Natural project. For example: *C:\Users\user01\workspace\MyProject*.
`-rootfolder`	Enable library root folder support. The folder *Natural-Libraries* is available. For example: *C:\Users\user01\workspace\MyProject\Natural-Libraries* .
`-sourcepath` *`directory`*	Path to the source files, including library and root folder.
`-c` *`file`*	Name of the configuration file, including path and extension. For example: *C:\Users\user01\NATstyleConfig.xml*.
`-o` *`file`*	Name of the output (result) file, including path and extension. For example: *C:\Users\user01\NATstyleResult.xml*.
`-sourcefiles` *`file-names`*	With this option individual sources can be selected for the NATstyle check. The names of the Natural sources (with file extension) preceded by the library container are specified. The library/source name pairs are separated using the semicolon (;) character. For example: `-sourcefiles MYLIB1/MYPGM1.NSP;MYLIB1/MYSUB1.NSN;MYLIB2/MYCPYCDE.NSC` **Note:** It is not necessary to specify the used libraries with the `-libraries` option.
`-libraries` *`library-names`*	Names of the Natural libraries to be checked. Separate the names using the semicolon (;) character. For example: `-libraries MYLIB1;MYLIB2`
`-exclude` *`library-names`*	Names of the Natural libraries to be excluded from checking. Separate the names using the semicolon (;) character.
`-quiet`	Do not display status messages.
`-help`	Display command line options and exit.

Overview of NATstyle Rules, Error Messages and Solutions

Rules for the following types of checks can be defined in the **NATstyle** preferences:

- Source Check
- Source Header Check
- Parser Check
- Error Check
- Resource Check
- Library Check

The rule names in the tables below are the names that are used in the **Configuration** group box of the NATstyle preferences.

Source Check

NATstyle provides the following rules for checking Natural sources:

Rule	Description	
Line length	Check the length of every source code line. For compatibility with the mainframe, the line length should be less than or equal to 72. In UNIX, OpenVMS and Windows environments, Natural allows lines which are up to 245 characters long.	
	Error Messages	Line too long. No input file. Value is not in range 10 - 245.
	Solution	Reduce the line length. Line constants may be split so that they use more than one line.
Whitespace	Check whether additional blank characters or tabs are used at the end of a source code line. Tab characters may cause problems in mainframe environments.	
	Error Messages	Line has trailing spaces. Line has trailing tabs. No input file.
	Solution	Remove any blank characters or tabs at the end of a line.
Tab character	Check whether tab characters are used. Tab characters may cause problems in mainframe environments.	
	Error Messages	Line contains tabs. No input file.
	Solution	Replace the tab characters with blanks.

Rule	Description	
Line numbers	Check the first four characters of every source line. If these characters are numeric and ascending, the source has line numbers. Numbered sources are only used in the server environment. NaturalONE does not use numbered lines.	
	Error Messages	File has line numbers. File has line numbers (not ascending). No input file.
	Solution	Remove the line numbers. The file may have been copied directly from the server environment (for example, using Natural Studio). It is recommended that you use the **Natural Server** view in order to **download** your sources into a project of your Eclipse workspace.
Empty lines	Check whether the source code contains empty lines. For compatibility with the mainframe, empty lines should be avoided because the Natural editors on the mainframe remove empty lines.	
	Error Messages	Empty line. No input file.
	Solution	Remove the empty line or mark the line as a comment.
Regular expression for single source lines	Check each source code line against a regular expression. For example, when you search for "find loop", this will only be found if the entire string is contained in a single line. For examples on regular expressions, search the internet for "java.util.regex.Pattern".	
	Error Messages	Regular expression not valid: %1%. Line matches regex: '%1%'. No input file.
	Solution	Remove or change the found code.
Regular expression for multiple source lines	Check the complete source against a regular expression. For example, when you search for "find loop", this will also be found if there is a line break between "find" and "loop". For examples on regular expressions, search the internet for "java.util.regex.Pattern".	
	Error Messages	Regular expression not valid: %1%. Source matches regex: '%1%'. No input file.
	Solution	Remove or change the found code.
Newline at end of line missing	Check whether a file ends with a newline character.	
	Error Messages	File does not end with newline. No input file.
	Solution	Add an empty line at the end of the file.

Rule	Description	
Maximum number of source lines	Check whether the maximum number of source lines (9999) has been exceeded.	
	Error Messages	Too many source lines. No input file. Value is not in range 10 - 9999.
	Solution	Reduce the number of source lines. For example, you can **externalize** parts of the program or delete comment lines.

Source Header Check

NATstyle provides the following rules for checking the Natural source header:

Rule	Description	
Header	Check whether the source has a valid Natural source header.	
	Error Messages	Has no source header. No Natural source type. No input file.
	Solution	Open the file with the source editor and save it. A source header will then be added automatically.
Has different Natural name	Check whether the source has a valid Natural source header, and check whether it contains a Natural object name (`:NatName`). The Natural object name is only shown in the source header when a **file name** exists for the object.	
	Error Messages	Source has different Natural name: %1%. No Natural source type. No input file.
	Solution	Rename the file with a name that matches the naming conventions.
Programming mode	Check the programming mode of the source.	
	Error Messages	Is structured mode. Is reporting mode. No Natural source type. No input file.
	Solution	Rewrite the program with the appropriate programming mode.

Rule	Description	
File name naming convention	If the source header contains a Natural object name, check the **file name** of the source.	
	Error Messages	Natural source name does not match naming conventions '%1%'. No Natural source type. No input file.
	Solution	Rename the file so that it matches your regular expression.
Struct	Check whether the source code lines are indented in such a way that the indentation reflects the structure of the program.	
	Error Messages	Not structed correctly. Struct ended with error: %1%. No Natural source type. No input file. Value is not in range 10 - 245. Value is not in range 1 - 9.
	Solution	Open the source with the source editor and use the **Struct** command.
DDM naming convention	Check whether the names of DDMs match the naming conventions.	
	Error Messages	DDM name does not match naming conventions '%1%'. No Natural source type. No input file.
	Solution	Rename the DDM so that it matches the naming conventions.

Parser Check

NATstyle provides the following rules for checks on parser level:

Rule	Description	
Data area define data	Check whether at least one variable has been defined in the `DEFINE DATA` block of a local data area, global data area or parameter data area.	
	Error Messages	Data area does not define any variable. Data area does not contain DEFINE DATA statement. No input file.
	Solution	Delete the data area. Then create a new data area and add at least one variable definition.

Rule	Description	
Unused local variables	Check whether the variables that are defined in the source are used.	
	Error Messages	Unreferenced variable: %1%. Unreferenced USING: %1%. No input file.
	Solution	Remove the variable, or remove the `USING` definition for a data area.
Unused search fields	Check whether the variables that are defined in the source are used as search fields.	
	Error Messages	Unreferenced search field: %1%. No input file.
	Solution	The view variable is only used as a search field and can be removed from the view.
Used AIV variables	Check whether the source uses application-independent variables (AIVs) that are not defined or whether they do not match a regular expression.	
	Error Messages	AIV variable: %1% used. No input file.
	Solution	Remove the AIV variable.
TODO comment	Check whether the sources contain `TODO` comments.	
	Error Messages	TODO: %1%. Error: DDM parser %1%. No input file.
	Solution	Remove the `TODO` comment.
FIXME comment	Check whether the sources contain `FIXME` comments.	
	Error Messages	FIXME: %1%. Error: DDM parser %1%. No input file.
	Solution	Remove the `FIXME` comment.
NATdoc source	Check the NATdoc header comments for consistency.	
	Error Messages	NATdoc missing. NATdoc first sentence missing. Error: DDM parser %1%. No input file.
	Solution	Add **NATdoc comments** and make sure that the first sentence ends with a period (.)

Rule	Description	
NATdoc variable	Check whether the source uses NATdoc field tags.	
	Error Messages	NATdoc field tag :out or :inout for %1% not valid for input-only field (BY VALUE). NATdoc field tag :in, :out or :inout for %1% missing. NATdoc field tag :redef for %1% missing. NATdoc field tag :in, :out or :inout for redefinition %1% missing. Error: DDM parser %1%. No input file.
	Solution	Add missing **field tags**.
NATdoc style	Check whether NATdoc header comments exist.	
	Error Messages	NATdoc field tag %1% missing. NATdoc field tag parameter %1% missing. Error: DDM parser %1%. No input file.
	Solution	Add missing **field tags**.
Upper case translation of source	Check whether all required strings have correctly been translated to upper case.	
	Error Messages	Source is not translated to upper case: %1%. No input file or parser not initialized.
	Solution	Enable **case translation** in the Natural preferences or change the string to upper case manually.
Local variables shadow view	Check whether a view field exists which has the same name as a variable of the Natural source.	
	Error Messages	Variable: %1% may shadow view field %2%. No input file.
	Solution	Rename the locally defined field.
Fully qualified variables	Check whether the variables in a Natural source are fully qualified. This rule applies to view fields and group fields.	
	Error Messages	Variable %1% not fully qualified as %2%. No input file.
	Solution	Add a group or view name to fully qualify the variable.
NPath complexity	Check the Npath complexity of the program.	
	Error Messages	NPath complexity of %1% (Highest = %2%).

Rule	Description	
		NPath complexity of subroutine %3% %1% (Highest = %2%). No input file.
	Solution	Split the program or reduce its complexity.
Cyclomatic complexity	Check the cyclomatic complexity of the program. The cyclomatic complexity number is also known as “McCabe metric”. To approximate this value, each Natural block statement increments the complexity by one. The complexity is evaluated separately for the main program and each inline subroutine.	
	Error Messages	Cyclomatic complexity of %1% (Highest = %2%). Cyclomatic complexity of subroutine %3% %1% (Highest = %2%). No input file. Value is not in range 10 - 9223372036854775807.
	Solution	Split the program or reduce its complexity.
Boolean complexity	Check Boolean expressions in `IF` and `DECIDE` statements of the Natural code. Count the number of the Boolean operators `OR` and `AND`.	
	Error Messages	Boolean expression complexity of %1% (Highest = %2%). No input file or parser not initialized. Value is not in range 10 - 9223372036854775807.
	Solution	Rewrite the program to reduce the complexity.
Function naming convention	Check whether the names of functions match the naming conventions.	
	Error Messages	Function name does not match naming conventions '%1%'. No input file.
	Solution	Rename the function.
Class naming convention	Check whether the names of classes match the naming conventions.	
	Error Messages	Class name does not match naming conventions '%1%'. No input file.
	Solution	Rename the class.
Subroutine naming convention	Check whether the names of inline subroutines and external subroutines match the naming conventions.	
	Error Messages	Subroutine name does not match naming conventions '%1%'. Regular expression not valid: %1%. No input file.
	Solution	Rename the subroutine.

Error Check

NATstyle provides the following rules for checking error message files:

Rule	Description	
Error file name	Check whether the names of error message files match the Natural naming conventions.	
	Error Messages	Name contains invalid language code. Name does not match naming conventions. No input file.
	Solution	Rename the error message file, or (even better) delete the error message file and **create** a new error message file.

Resource Check

NATstyle provides the following rules for checking Natural resources:

Rule	Description	
Resource file name	Check whether Natural resources with a specific file extension match a specific naming convention (as defined with the `regex` property).	
	Error Messages	File name does not match naming conventions '%1%'. Regular expression not valid: %1%. No input file.
	Solution	Rename the resource file.

Library Check

NATstyle provides the following rules for checking Natural libraries:

Rule	Description	
Library directly below root folder	Check whether a Natural library is located directly below the **library root folder**. The library root folder is used if new objects are created for this library and no folder is specified.	
	Error Messages	For library %1%, no library can be found directly below the library root folder. No library or library folder found.
	Solution	Create a new library below the library root folder.
Duplicate libraries	Check whether the project contains more than one folder with the name of a Natural library.	
	Error Messages	Project contains more than one library for Natural library %1%.

Rule	Description	
		No library or library folder found.
	Solution	Make sure that only one library is created. It should be placed directly below the "Natural-Libraries" root folder.
Library folder names	Check whether the **library folders** match the naming conventions. The following folders are not tested: a folder with the same name as a Natural library, a folder named after a Natural object type, and the **special folders** with the reserved names "SRC", "ERR" and "RES".	
	Error Messages	Library folder %1% does not match naming conventions (%2%). No library or library folder found.
	Solution	Change the name of the library folder.
Mixed object grouping	Check whether a library contains **group folders** or the **special folders** with the reserved names "SRC", "ERR" and "RES". Only one type of grouping should be used.	
	Error Messages	Library %1% contains a mix of special folders and group folders.
	Solution	Remove folders so that only one type of grouping is used.
Special folders	Check whether a Natural library only contains the **special folders** with the reserved names "SRC", "ERR" and "RES". All other folders will be marked as wrong.	
	Error Messages	Library %1% should only contain special folders. No library or library folder found.
	Solution	Remove all marked folders. Only special folders with the reserved names "SRC", "ERR" and "RES" are valid.
Group folders	Check whether a Natural library only contains **group folders**. All other folders will be marked as wrong.	
	Error Messages	Library %1% should only contain group folders. No library or library folder found.
	Solution	Remove all marked folders. Only group folders are valid (that is, folders which are named after a Natural object type or with the names "Resource" and "Error").
NATdoc library documentation (library.html)	Check whether a Natural library contains a **library comment file** with the name *library.html*.	
	Error Messages	NATdoc library documentation (library.html) missing. No library or library folder found.
	Solution	Add a library comment file with the name *library.html* to the root of the library.

DTDs Used by NATstyle

The following topics are covered below:

- Structure of the Configuration File
- Structure of a NATstyle Result File

Structure of the Configuration File

The following DTD defines the structure of the configuration file:

```
<!ELEMENT naturalStyleCheck (packages?, checks+)>
<!ATTLIST naturalStyleCheck version CDATA #IMPLIED>
<!ELEMENT packages (package+)>

<!ELEMENT package  EMPTY>
<!ATTLIST package  url      CDATA #REQUIRED>

<!ELEMENT checks   (check+)>
<!ATTLIST checks   type     (source|header|parser|error
                            |resource|library) #REQUIRED>

<!ELEMENT check    (property*)>
<!ATTLIST check    class    CDATA #REQUIRED
                   name     CDATA #IMPLIED
                   severity (error|info|warning) #REQUIRED
                   disabled (true|false) #IMPLIED>

<!ELEMENT property EMPTY>
<!ATTLIST property value    CDATA #REQUIRED
                   name     CDATA #REQUIRED>
```

The elements of the DTD are described in the following table:

Element	Description
`naturalStyleCheck`	Top-level element. Contains the `version` of the used `naturalStyleCheck` implementation.
`packages`	Contains one or more `package` definitions. Each `package` contains the `url` to a JAR file that contains user-written extensions to NATstyle. **Note:** User-written extensions to NATstyle are currently not supported.
`checks`	Checks are categorized for specific test groups. Each of these `type` categories contains different tests.
`check`	A check needs a type-specific implementation within a Java class. The implementation is identified by the `class` name and the category type of the checks. Each check has a `severity` containing the found violation. The `name` is a meaningful

Element	Description
	short description of the test and will be used as the name in the property pages. With the `disabled` flag, a check can be switched on and off.
`property`	Each check may contain different properties. Each `property` has a `name` and a `value`. The range of values is defined in the implementing class.

An implementation class can be specified for more than one check. It can be used, for example, if more then one "Regular expression to each source line" check is to be performed with different settings. The following document shows part of the default configuration file:

```
<?xml version="1.0" encoding=" utf-8"?>
<naturalStyleCheck version="1.0">
  <checks type="source">
    <check class="CheckLineLength" name="Line length" severity="warning">
      <property name="max" value="72"/>
      <property name="exclude" value="D3"/>
    </check>
    <check class="CheckWhiteSpace" name="Whitespace" severity="info">
      <property name="noTab" value="false"/>
      <property name="exclude" value="D3T"/>
    </check>
    <check class="CheckTabs" name="Tab character" severity="info"/>
    <check class="CheckHasLineNumbers" name="Line numbers" severity="warning"/>
    <check class="CheckEmptyLine" name="Empty lines" severity="info">
      <property name="exclude" value="D3T"/>
    </check>
    <check class="CheckRegExLine" name="Regular expression to each source line"
                                  severity="warning" disabled="true">
      <property name="exclude" value=""/>
      <property name="regex" value=""/>
    </check>
    <check class="CheckRegExSource" name="Regular expression to test source"
                                    severity="warning" disabled="true">
      <property name="exclude" value=""/>
      <property name="regex" value=""/>
    </check>
    <check class="CheckNewlineAtEndOfFile" name="Newline at end of file missing"
                                           severity="info">
      <property name="exclude" value="D"/>
    </check>
  </checks>
...
</naturalStyleCheck>
```

For more information on NATstyle configuration files, see ***NATstyle*** in *Setting the Preferences*.

Structure of a NATstyle Result File

The following DTD defines the structure of the generated result files:

```
<!ELEMENT NATstyle (fatal*, file*)>
<!ATTLIST NATstyle  version  CDATA #REQUIRED>
<!ELEMENT fatal EMPTY>
<!ATTLIST fatal     class    CDATA #REQUIRED
                    message  CDATA #REQUIRED
                    name     CDATA #IMPLIED>
<!ELEMENT file (error*, fatal*)>
<!ATTLIST file      type     (lib|err|res|src) #REQUIRED
                    location CDATA #REQUIRED>
<!ELEMENT error EMPTY>
<!ATTLIST error     class    CDATA #REQUIRED
                    message  CDATA #REQUIRED
                    severity (error|info|warning) #REQUIRED
                    name     CDATA #IMPLIED
                    line     CDATA #IMPLIED>
```

The elements of the DTD are described in the following table:

Element	Description
`NATstyle`	Top-level element. Contains the `version` of the used `NATstyle` implementation.
`file`	Each tested `file` is added to the result. The `location` of the file specifies the full path to the file. The `type` attribute categorizes whether the file is used as Natural source (`src`), Natural error message file (`err`), Natural resource (`res`) or Natural library (`lib`).
`error`	If a check is violated, an `error` element is added. The `error` element specifies the `class` that generated this entry, a meaningful `name` and the `severity` of the error. If available, the error `line` will be specified. The `message` describes the kind of violation.
`fatal`	In case the checks or a specific check on a file fails, a `fatal` entry is added. The `class` specifies the implementation that failed and the `message` contains detailed information about the failure.

The following is an example of a generated result file:

```
<?xml version="1.0" encoding="UTF-8"?>
<NATstyle version="1.0">

<file type="src" location="P:\SYSEXPG\Natural-Libraries\SYSEXPG\Programs\CNOTX04.NSP">
  <error severity="warning" line="11" message="Unreferenced search field: CITY."
    class="com.softwareag.naturalone.natural.natstyle.check.src.parser.CheckUnusedSearchFields"
    name="Unused Search Fields"/>
  <error severity="warning" message="NATdoc missing."
    class="com.softwareag.naturalone.natural.natstyle.check.src.parser.CheckNATdocSource"
    name="NATdoc Source"/>
  <error severity="warning" line="20" message="Variable BONUS not fully qualified as INCOME.BONUS."
    class="com.softwareag.naturalone.natural.natstyle.check.src.parser.CheckFullQualifiedVariable"
    name="Fully qualified variables."/>
</file>

<file type="src" location="P:\SYSEXPG\Natural-Libraries\SYSEXPG\Programs\COMPRX03.NSP">
  <error severity="info" line="29" message="Not structed correctly."
```

```
    class="com.softwareag.naturalone.natural.natstyle.check.src.header.CheckStruct" name="Struct"/>
  <error severity="warning" line="10" message="Unreferenced search field: CITY."
    class="com.softwareag.naturalone.natural.natstyle.check.src.parser.CheckUnusedSearchFields"
    name="Unused Search Fields"/>
  <error severity="warning" message="NATdoc missing."
    class="com.softwareag.naturalone.natural.natstyle.check.src.parser.CheckNATdocSource"
    name="NATdoc Source"/>
</file>

</NATstyle>
```

X Using the Natural Profiler

25 Using the Natural Profiler

- General Information 386
- Prerequisites 386
- Starting a Profiler Session 387
- Viewing the Profiler Output 388
- Managing the Profiler Sessions 392
- Displaying Natural Profiler Resource Data 396
- Application Programming Interface 398
- Overview of Event Types 398

General Information

The Natural Profiler is used to monitor the internal process flow of a Natural application and to collect trace data for selected events that are performed within a Natural application. It helps application programmers, administrators and quality engineers to analyze the logical flow of Natural applications and to trace the utilization of resources. A Natural Profiler analysis can be used for performance optimization of Natural applications as described in *Performance Analysis of Natural Applications* in the *NaturalONE in a Nutshell* documentation.

Prerequisites

If you want to use the Natural Profiler in a UNIX or Windows environment, Web I/O must have been enabled on the Natural server. Otherwise, a profiler session cannot be started.

The use of the Natural Profiler can be controlled by Natural Security. If Natural Security is being used, a Natural Profiler utility profile must have been defined, within which the **Start/Stop Profiler Tracing** option must be set. For more information, see the *Natural Security* documentation, which is part of the Natural documentation.

If you want to use the Natural Profiler in a UNIX or Windows environment, Natural Version 8.3.6 or above is required on the server.

If you intend to collect data for the used Natural statements and you are using a Natural server in a UNIX or Windows environment, the programs to be traced must have been cataloged with `GPGEN=(PROFILER=ON)`. With `GPGEN=(PROFILER=ON)`, statement events will be generated when the program is executed. If an object was catalogued with `GPGEN=(PROFILER=OFF)`, the program's icon is overlaid with a warning decorator on the **Hot Spots** page, and an explanatory tooltip is shown for the program node when hovering over it with the mouse.

If you want to use the Natural Profiler in a mainframe environment, a batch Natural Development Server is required. If you try to profile a program and your Natural Development Server uses the Natural CICS adapter, you will receive the following message:

```
Response Code: 102 (Initialization of the Profiler Data Pool failed (NATRDC1/I RC=-1))
```

Event data from a Natural batch profiling session on the mainframe is stored in a Natural resource file. By default, mainframe resource files are not shown in NaturalONE. If you want to view the event data in NaturalONE (see also ***Displaying Natural Profiler Resource Data***), you have to enable the display of the resource files in your Natural Development Server environment. For more information, see the prerequisites for the Profiler utility in batch mode in the *Utilities* documentation, which is part of the Natural documentation for mainframes.

If you profile a program in a mainframe environment, NaturalONE uses automatically the following parameter setting which activates the profiling:

```
RDCSIZE=2 RDCEXIT=NATRDC1
```

Therefore, there is no need to specify these parameters as session parameter or in a parameter module.

Note: If you overwrite the NaturalONE setting with other values as session parameter or in a parameter module, for example with

```
RDCSIZE=0
```

the Profiler will not deliver data and the NaturalONE Profiler output is empty.

Starting a Profiler Session

To use the Natural Profiler with a Natural program, you must first compile the program in your Natural environment in order to create a generated program in this environment. See also ***Updating the Objects in the Natural Environment***.

The library in which a profiler session for the generated program is started is determined by the mode you have defined for the project (either shared mode or private mode). See ***Steplibs*** in *Changing the Project Properties* for further information.

Important: The contents of the source in the workspace must be identical with the contents of the corresponding object on the server.

- Starting a Profiler Session and Displaying the Output

Starting a Profiler Session and Displaying the Output

When you start a profiler session for an application on a Natural server as described below, trace data for the defined event types is collected on the server during the profiler session. The event types are defined in the Natural preferences. For more information, see ***Profiler*** in *Setting the Preferences*.

For details concerning the **Default Launch** settings, see ***Launching Natural Applications***.

To start a profiler session and display the output

1. In the **Navigator** view or in the **Natural Navigator** view, select the program for which you want to start a profiler session.
2. Invoke the context menu and choose **Profile As > Natural Application**.

The profiler session is started and the following occurs:

- The profiler output is shown in the editor area. For more information, see *Viewing the Profiler Output*.
- An entry for the program for which you have started the profiler session is shown in the **Profiler Sessions** view. For more information on this view, see *Managing the Profiler Sessions*.

Viewing the Profiler Output

The profiler output is automatically shown in the editor area when you **start** a profiler session or **open** a Natural Profiler resource file. It is shown on the following profiler pages:

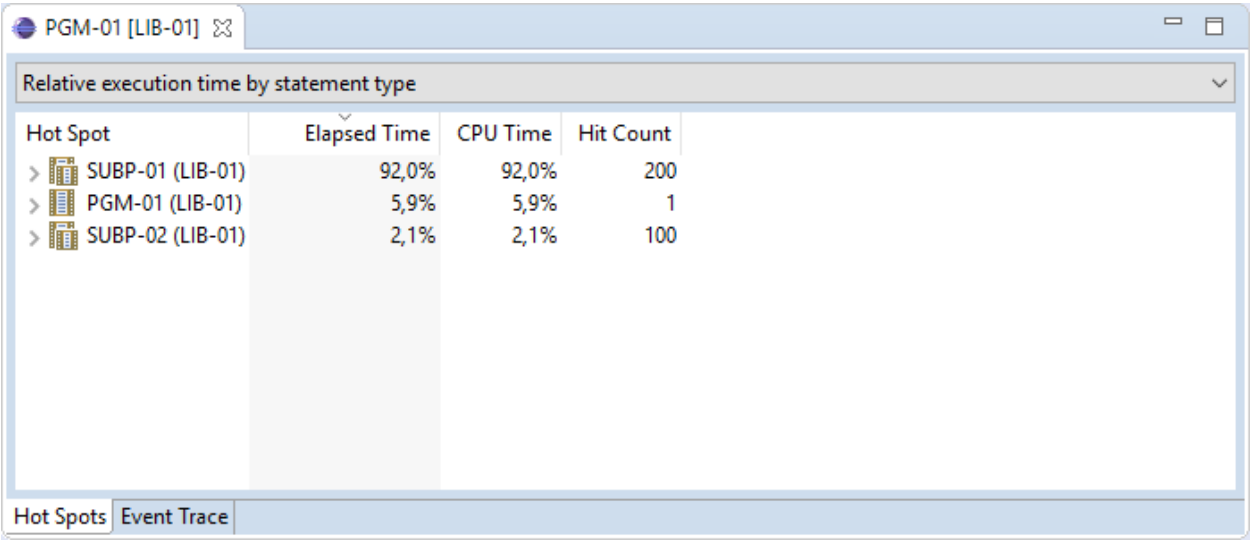
- Hot Spots
- Event Trace

Hot Spots

This page shows how the time was primarily spent during the execution of the application. It can be used to identify the parts of the application that can be most expediently optimized.

The hot spots are the involved Natural objects. The elapsed time and the CPU time for each Natural object are shown in percent. The hit count for each Natural object is also shown; this informs you how often the object has been invoked (for example, in a loop).

If your code contains an I/O statement (for example, an `INPUT` or `WRITE` statement), the elapsed time also includes the time that the system has waited for you to react on the output.



When the option **Profile statements** is enabled in the Natural preferences (see *Profiler > Hot Spots* in *Setting the Preferences*), expandable nodes are shown. When you expand a node, the values for the Natural objects are no longer shown. Instead, elapsed time, CPU time and hit count are shown for each item in the expanded node. This can be either the statement type or, when you expand the node further, a line number in the source. For example:

PGM-01 [LIB-01]

Relative execution time by statement type

Hot Spot	Elapsed Time	CPU Time	Hit Count
SUBP-01 (LIB-01)			
IF	28,0%	28,0%	20000
FOR	23,5%	23,7%	20200
ASSIGN/COMPUTE			
Line 10	20,4%	20,2%	20000
LOOP	16,2%	16,3%	20000
END	3,2%	3,2%	200
IGNORE	0,3%	0,3%	400
Initialization	0,3%	0,2%	200
PGM-01 (LIB-01)	5,9%	5,9%	1
SUBP-02 (LIB-01)	2,1%	2,1%	100

Hot Spots | Event Trace

You double-click a node to expand it. When a node is fully expanded (that is, when a line number is shown), you can navigate directly to the corresponding source code line. To do so, you double-click the entry which indicates the line number, or you select that entry and then choose **Edit** from the context menu.

If you want to reduce the number of events sent by the trace session, you can enable sampling in the Natural preferences. Or you can specify display limits in the Natural preferences. See *Profiler > Hot Spots* in *Setting the Preferences* for further information.

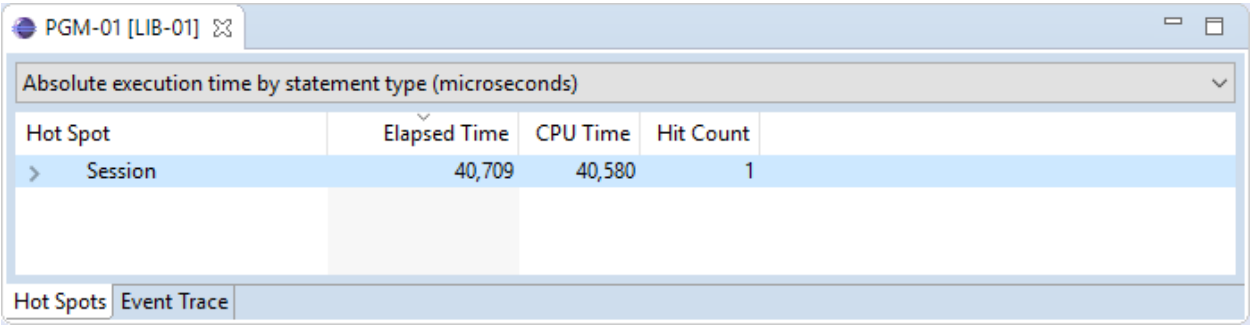
The drop-down list box which is shown at the top of the **Hot Spots** page allows switching between different built-in display layouts. In particular, the choice made here determines whether the execution times are displayed in relative (percentual) or absolute form and whether the statements are categorized by statement type as shown above.

Absolute execution time by statement type (microseconds)

Relative execution time by statement type
Relative execution time
Absolute execution time by statement type (microseconds)
Absolute execution time (microseconds)

The default layout that is initially used can be set in the Natural preferences. See *Profiler > Hot Spots* in *Setting the Preferences* for further information.

Note that when absolute execution time layouts are used, an extra top-level node ("Session") is introduced as shown below, allowing the absolute execution times for the entire session to be inspected.



Event Trace

This page is only shown if the event trace has been enabled in the Natural preferences. The event types to be collected are also defined in the Natural preferences. For further information, see *Profiler > Event Trace* in *Setting the Preferences*.

When the event trace has been enabled, this page shows the trace records for the events which have been collected on the server. These records provide the raw data on which the information on the **Hot Spots** page is based. For example, the **Before Database Call** (DB) and **After Database Call** (DA) trace records define the extent of a specific database event.

Note: For information on the abbreviations which are shown in the **Type** column of the **Event Trace** page (such as DB or DA), see *Overview of Event Types*.

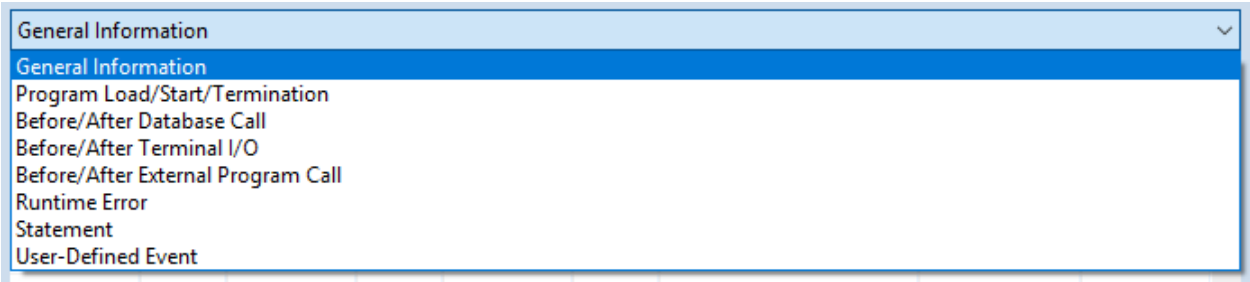
PGM-01 [LIB-01]

General Information

Record	Type	Pgm Lib.	Level	Curr. Pgm	Line	Event Time	Elapsed Time	CPU Time	CPU Delta
477	PL	LIB-01	1	PGM-01	110	00:00:01.784656	0.000006	0.021185	0.000006
478	PS	LIB-01	2	SUBP-01		00:00:01.784662	0.000180	0.021191	0.000179
479	PT	LIB-01	2	SUBP-01	16	00:00:01.784842	0.000005	0.021370	0.000005
480	PR	LIB-01	1	PGM-01	110	00:00:01.784847	0.000001	0.021375	0.000001
481	PL	LIB-01	1	PGM-01	120	00:00:01.784848	0.000005	0.021376	0.000006
482	PS	LIB-01	2	SUBP-02		00:00:01.784853	0.000004	0.021382	0.000003
483	PT	LIB-01	2	SUBP-02	13	00:00:01.784857	0.000004	0.021385	0.000005
484	PR	LIB-01	1	PGM-01	120	00:00:01.784861	0.000003	0.021390	0.000002
485	PL	LIB-01	1	PGM-01	110	00:00:01.784864	0.000005	0.021392	0.000005
486	PS	LIB-01	2	SUBP-01		00:00:01.784869	0.000179	0.021397	0.000179
487	PT	LIB-01	2	SUBP-01	16	00:00:01.785048	0.000005	0.021576	0.000005
488	PR	LIB-01	1	PGM-01	110	00:00:01.785053	0.000001	0.021581	0.000001
489	PL	LIB-01	1	PGM-01	120	00:00:01.785054	0.000005	0.021582	0.000005
490	PS	LIB-01	2	SUBP-02		00:00:01.785059	0.000004	0.021587	0.000004
491	PT	LIB-01	2	SUBP-02	13	00:00:01.785063	0.000004	0.021591	0.000004
492	PR	LIB-01	1	PGM-01	120	00:00:01.785067	0.000003	0.021595	0.000003
493	PL	LIB-01	1	PGM-01	110	00:00:01.785070	0.000005	0.021598	0.000005
494	PS	LIB-01	2	SUBP-01		00:00:01.785075	0.000179	0.021603	0.000179
495	PT	LIB-01	2	SUBP-01	16	00:00:01.785254	0.000005	0.021782	0.000004
496	PR	LIB-01	1	PGM-01	110	00:00:01.785259	0.000001	0.021786	0.000001

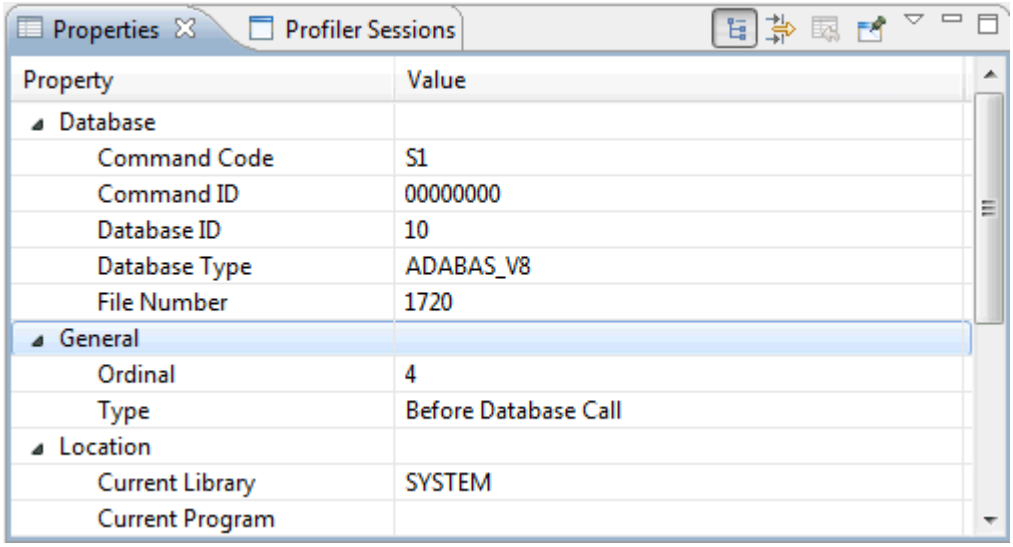
Hot Spots | Event Trace

The drop-down list box which is shown at the top of the **Event Trace** page determines the columns that are shown on this page. It lists the so-called “views” from the mainframe. The "General Information" view is shown by default. The columns of this view show general data which are common to all records. The other views show columns for event-specific data.



You can reorder the columns on the **Event Trace** page by dragging a column header to a different position.

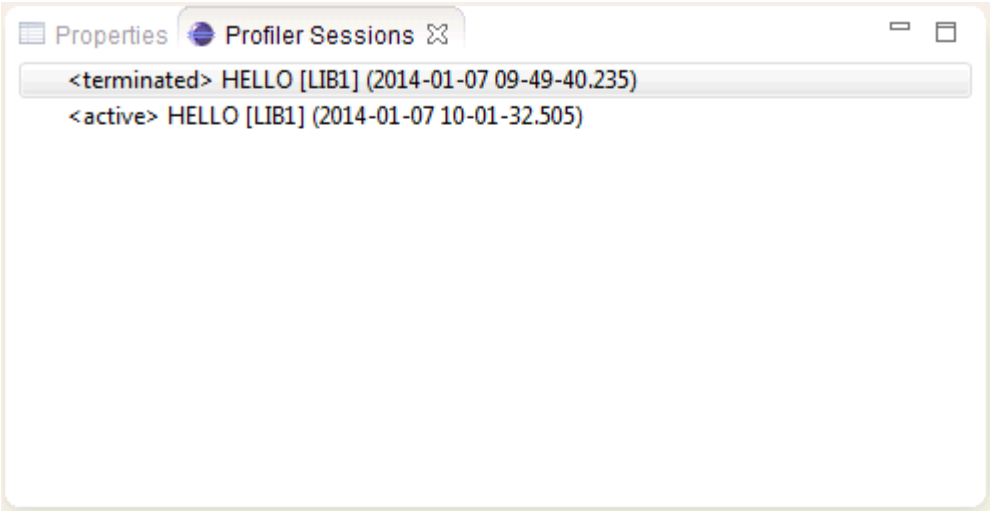
When you select an event on the **Event Trace** page, the contents of the corresponding trace record are automatically shown in the **Properties** view. Other than on the **Event Trace** page where only selected columns are shown, the **Properties** view shows all trace data for a record.



The information that is shown in the **Properties** view depends on the type of the event that is currently selected on the **Event Trace** page. For example, database ID and file number are shown for a database event, but not for a terminal I/O event.

Managing the Profiler Sessions

The **Profiler Sessions** view appears automatically when you start a profiler session for the first time. It shows an entry for each program for which you have started a profiler session. The list of profiler sessions in this view is not persistent. It shows only the profiler sessions which have been started during the current Eclipse session. If you want to keep the profiling results of a profiler session for a later Eclipse session, you can **save** them.



When you exit Eclipse, NaturalONE terminates all of your active profiler sessions on the server.

Note: The **Profiler Sessions** view is a Natural-specific view. If it is currently not shown, you can display it with **Window > Show View > Other > Software AG NaturalONE > Profiler Sessions**.

- Possible States for the Profiler Sessions
- Opening a Profiler Session
- Saving the Results of a Profiler Session
- Stopping a Profiler Session
- Removing a Profiler Session

Possible States for the Profiler Sessions

The profiler sessions that are shown in the **Profiler Sessions** view can be in different states. The current state is always indicated next to the session name. Possible states are:

- **Active**
  The sessions for which you have started profiling and which are still running are in the state "active".
- **Stopped**
  The sessions for which you have stopped profiling are in the state "stopped". See also *Stopping a Profiler Session*.
- **Terminated**
  The sessions which have been terminated on the server are in the state "terminated".

Opening a Profiler Session

When you start a profiler session, the profiler output is automatically shown in the editor area. If you close the output for a particular profiler session, you can reopen the session as described below.

Note: Keep in mind that the **Profiler Sessions** view only shows the profiler sessions of the current Eclipse session. If you want to open a saved profiler session from a previous Eclipse session, you have to do this from the **Navigator** view or **Natural Navigator** view. See also *Saving the Results of a Profiler Session*.

To open a profiler session

1. In the **Profiler Sessions** view, select the profiler session that you want to open.
2. Invoke the context menu and choose **Open**.

   Or:

   Double-click the profiler session that you want to open.

Saving the Results of a Profiler Session

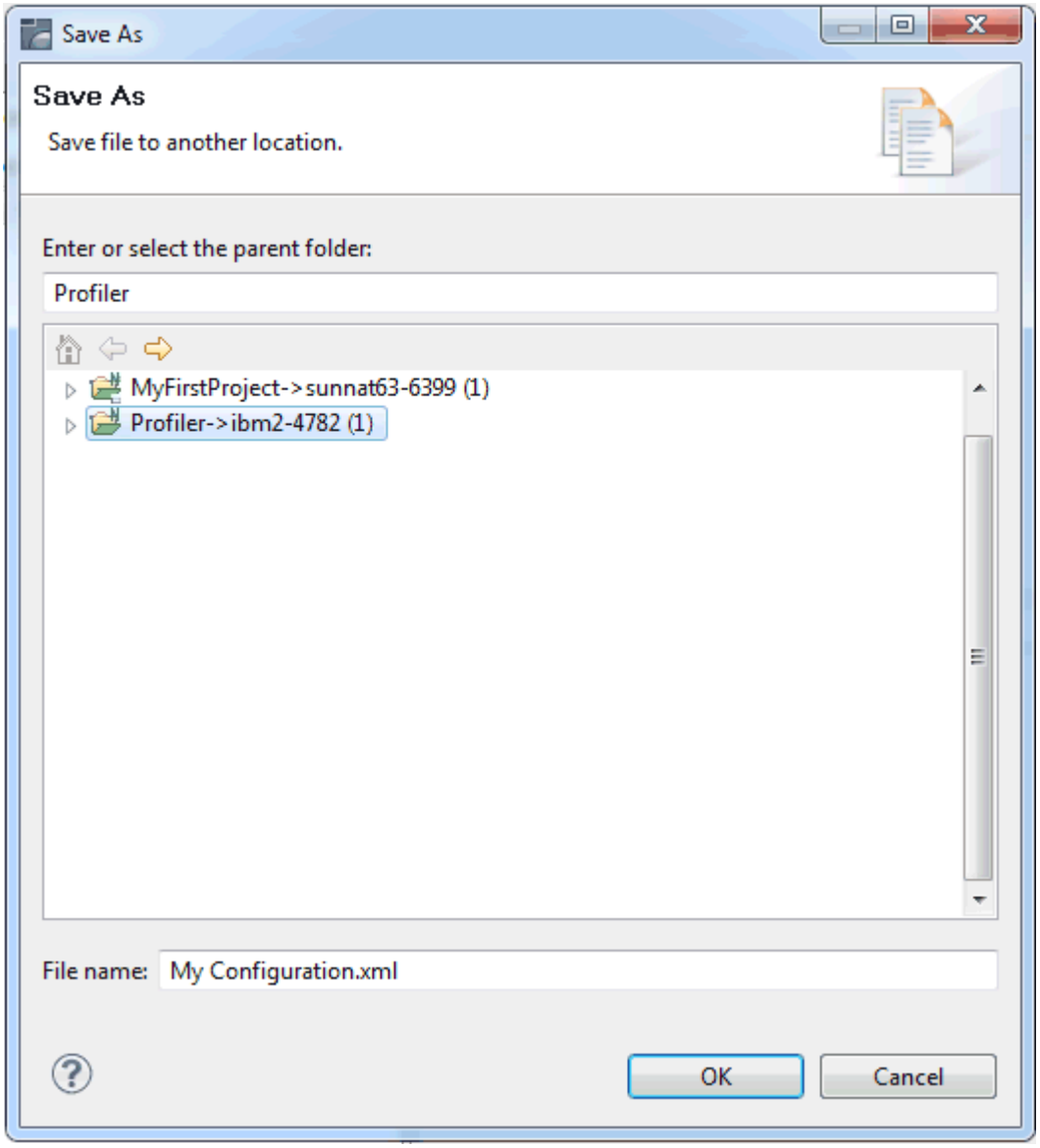
All profiler sessions that you start only exist as long as your Eclipse session is running. If you want to keep the results of a profiler session for a later Eclipse session, you have to save the session. This is helpful, if you want to compare the trace results of different profiler sessions in order to find out, for example, whether the performance of your application has improved.

A saved profiler session will be shown as an XML file in the **Navigator** view and in the **Natural Navigator** view. You can then open the output for this session in the editor area by double-clicking it or by using the **Open** command.

To save a profiler session

1 In the **Profiler Sessions** view, invoke the context menu for the session that you want to save and choose **Save As**.

2 In the resulting dialog box, select the project in which the profiler session is to be stored and specify a name for the session. For example:



3 Choose the **OK** button.

Stopping a Profiler Session

When a profiler session is currently active, you can stop profiling. In this case, the profiler output in the editor area will show incomplete data. Stopped profiler sessions cannot be restarted.

Note: Even though a session has been stopped, it continues running on the server until it terminates there.

To stop a profiler session

1 In the **Profiler Sessions** view, select the session that you want to stop.

2 Invoke the context menu and choose **Stop Profiling**.

The new state "stopped" is now shown next to the session name. The entire entry for the stopped session is shown in red color. This is to remind you that the corresponding profiler output in the editor area shows incomplete data.

Removing a Profiler Session

You can remove any terminated profiler sessions from the **Profiler Sessions** view.

When you remove a profiler session, this also indirectly causes the collected data for the removed session to be released from memory if nothing else in the Eclipse session is referencing it (for example, profiler output in the editor area).

To remove a profiler session

1 In the **Profiler Sessions** view, select the session that you want to remove.

2 Invoke the context menu and choose **Remove**.

Displaying Natural Profiler Resource Data

Beside the interactive profiling from within NaturalONE (see ***Starting a Profiler Session and Displaying the Output***), Natural provides a native profiling on the server side. Native profiling is performed in two steps:

1. Natural is configured such that the session is profiled and the generated profiling data is written to a resource file.
2. The profiling data contained in the resource file is read and displayed in NaturalONE.

The profiling of Natural batch applications on the mainframe is described in detail in the *Profiler Utility* section of the *Utilities* documentation, which is part of the Natural documentation for mainframes. For non-mainframe environments, please refer to the documentation for the `PROFILER` parameter in the *Parameter Reference*.

Note: For mainframe environments, the generation of a resource file is restricted to batch mode. For non-mainframe environments, native profiling is generally available. However, if the session is profiled interactively from within NaturalONE, the `PROFILER` parameter settings are ignored and no resource file is generated.

- Resource File Types
- Opening a Natural Profiler Resource File

- Filtering the Event Data

Resource File Types

There are two types of resource files used by the Natural Profiler:

1. "Flat" resource files (extension *.nprf*). These contain an individual record for each recorded event. As such, they allow tracing of events in the **Event Trace** page, in addition to containing the information for the **Hot Spots** page. However, they can quickly become very large, especially if Natural statements are being traced.
2. "Consolidated" resource files (extension *.nprc*). In this case, multiple events are combined into a single file record and the individual event information is not retained. As such, event tracing is not possible and they only contain the information for the **Hot Spots** page. However, they are invariably very much smaller than the flat files and are thus correspondingly quicker to process.

In mainframe environments, consolidated resource files can be created by setting the `CONSOLIDATE` keyword of the Profiler utility `INIT` function to `ON` (see the possible settings for the `CONSOLIDATE` keyword described in the *Initializing Profiling in the Profiler Utility* section of the *Utilities* documentation contained in the *Natural for Mainframes* documentation).

In non-mainframe environments, consolidated resource files are created directly in preference to flat files if the user has specified that individual events should not be traced (see the possible settings for the `EVENTTRACE` subparameter described in the *PROFILER Parameter Syntax* in the *Parameter Reference*).

The data of a flat resource file can be consolidated and written to a consolidated resource file with the Profiler utility `CONSOLIDATE` function which is available in all environments.

Opening a Natural Profiler Resource File

When you open a Natural Profiler resource file, the output is the same as described in ***Viewing the Profiler Output***, with the following restriction: On the **Hot Spots** page, the direct navigation to a source code line does not work with a Natural Profiler resource file.

Note: In mainframe environments, the display of resource files must have been enabled in your Natural Development Server environment (see also ***Prerequisites***).

To open a Natural Profiler resource file

1. In the **Natural Server** view, go to the library to which the resource file was written.
2. Expand the **Resources** node of the library and then the **NPRF File** or **NPRC File** node.
3. Select the resource file that you want to open.
4. Invoke the context menu and choose **Open with Natural Profiler**.

Or:

Double-click the resource file that you want to open.

A profiler session is started and the following occurs:

- The profiler output is shown in the editor area. For more information, see *Viewing the Profiler Output*.
- An entry for the program for which you have started the profiler session is shown in the **Profiler Sessions** view. For more information on this view, see *Managing the Profiler Sessions*.

Filtering the Event Data

When a Natural Profiler resource file is opened, the trace data collected in the resource is transferred to the client. The settings in the Natural preferences are used to filter the data on the client side. For more information, see *Profiler* in *Setting the Preferences*.

Note: The sampling setting for the profiler is ignored when viewing a Natural Profiler resource file.

Application Programming Interface

When a profiler session has been started, the profiler monitoring can be paused and restarted from the profiled Natural application by calling the application programming interface (API) `USR8210N`. The API can also be used to get the current state of the monitoring process. The API is delivered in the `SYSEXT` library. For more information, see the description of the SYSEXT utility in the Natural documentation for your platform.

Overview of Event Types

During a Natural session, different kinds of events can occur, for example, “starting a program”. Data specific to an event is collected in a trace record. The following event types are available:

Event Type	The event occurs ...
Session Initialization (SI)	... when the session is initialized during session startup.
Session Termination (ST)	... when the session is being terminated.
Program Load (PL)	... when a program is loaded or when it is already located in the Natural buffer pool on the server.
Program Start (PS)	... when a program is started.
Program Termination (PT)	... when a program is terminated.
Program Resume (PR)	... when control returns to one program level from another, including the case where control returns to level 0 (no program active).
Before Database Call (DB)	... before a database call is executed.
After Database Call (DA)	... after a database call has been executed.
Natural Statement (NS)	... when a Natural statement is executed. For technical reasons, there is no one-to-one relationship between a Natural source code statement and an object code in the cataloged object. Therefore, multiple Natural statements can be merged into one "Natural Statement" event and conversely, one Natural statement can cover multiple "Natural Statement" events.
Before Terminal I/O (IB)	... before a terminal input/output is executed.
After Terminal I/O (IA)	... after a terminal input/output has been executed.
Before External Program Call (CB)	... before an external program call (`CALL` statement) is executed.
After External Program Call (CA)	... after an external program call (`CALL` statement) has been executed.
Runtime Error (E)	... when a Natural runtime error has occurred.
User-Defined Event (U)	... when a user-defined event is generated using the Natural statement `CALL 'CMRDC' 'U' ' some-user-data'`. The first byte of the user data (a blank in this example) is interpreted as the event subtype.
Monitor Pause (MP)	... when event collection is temporarily interrupted (for example, because an FNAT program is currently being executed or because the profiler data pool is full). In addition, monitoring can be explicitly paused via the application programming interface `USR8210N`.

The abbreviations which are provided in the above table (behind the event type) are used in the **Type** column of the profiler's **Event Trace** page.

XI

- 26 Using the Natural Code Coverage 403
- 27 Using the Natural Coverage Plugin for Jenkins 409

26 Using the Natural Code Coverage

- General Information 404
- Quick Start 404
- Prerequisites 405
- Producing Natural Code Coverage Data 406
- Viewing Natural Code Coverage Data 406

General Information

The Natural Code Coverage is used to monitor the executed statements of a Natural application. It helps application programmers, administrators and quality engineers to analyze the code coverage of Natural applications and to investigate the coverage results in more detail with the Natural Source Editor.

A Natural Code Coverage analysis can be performed for finding unused or "dead" code in the Natural Application. This technique is typically used with automated testing. After having identified testing leaks with the Natural Code Coverage tool, quality engineers can add new or extend existing unit tests to finally get a better code coverage result.

Quick Start

The following is a quick description of how Natural Code Coverage could be used.

To use Code Coverage

1 Create a project containing the application sources using the command **Add to New Project**.

   Or:

   Create a project containing the application sources via checkout from subversion.

2 Set the parameters required for Code Coverage in the **Session parameters** field of the Natural runtime properties of the project:

   - `GPGEN=(COVERAGE=ON)`
   - `COVERAGE=(ACTIVE=ON,RESLIB=MYLIB,RESNAME=MYRES)`

3 Build the Natural application.

   With `GPGEN=(COVERAGE=ON)` the Natural GP information gets instrumented in the Natural environment to be used for code coverage..

4 Issue the command **Run As > Natural application**.

   Or:

   Press ALT+SHIFT+X, N.

   Or:

With the parameter `COVERAGE=(ACTIVE=ON,RESLIB=MYLIB,RESNAME=MYRES)`, Natural code coverage analysis is done while the application is running and the result of this analysis is written to a resource file *MYRES.ncvf* in library MYLIB.

5 Issue the command **Add to Existing Project** on the coverage resource file to add it to the NaturalONE project created in Step 1.

6 Select the code coverage resource file and issue the command **Open with Natural Code Coverage**.

The **Code Coverage** view is opened and shows the result of the code coverage analysis.

7 Double-click on the code covered Natural sources or use the **Edit** command from the context menu to open the code covered Natural sources.

The **Source Editor** is opened and all lines executed for this object are displayed with a green background.

Prerequisites

If you want to use the Natural Code Coverage in a UNIX or Windows environment, Natural Version 8.4.1 or above is required on the server.

The programs to be code covered must have been cataloged with the Natural parameter `GPGEN=(COVERAGE=ON)`. The GPGEN parameter is described in detail in the *Parameter Reference* section of the Natural documentation corresponding to your platform.

In order to activate Natural Code Coverage at runtime the COVERAGE parameter must be set to active: `COVERAGE=(ACTIVE=ON)`. The COVERAGE parameter is described in detail in the *Parameter Reference* section of the Natural documentation corresponding to your platform.

At end of application execution, a code coverage result file with the extension *.ncvf* is written.

Code Coverage in Mainframe Environments

By default, mainframe resource files are not shown in NaturalONE. If you want to get access to the *.ncvf* file, you have to enable the display of the resource files in your Natural Development Server environment.

For more information regarding resources and the Natural code coverage of Natural batch applications on the mainframe, refer to the *Profiler Utility* section of the *Utilities* documentation, which is part of the Natural documentation for Mainframes.

Producing Natural Code Coverage Data

To use the Natural Code Coverage with a Natural program in a Windows or UNIX environment, you must first compile the program in your Natural environment with the parameter `GPGEN=(COVERAGE=ON)`. This creates a generated program instrumented for code coverage in this environment.

See also ***Updating the Objects in the Natural Environment***.

The Natural Application must then be executed with the parameter `COVERAGE=(ACTIVE=ON)`. Regarding the execution of a Natural application please also refer to the section ***Executing Objects***.

For details concerning the **Default Launch** settings, see ***Launching Natural Applications***.

As a result of executing an application with the Natural Code Coverage, a resource file with the extension *.ncvf* is created in the Natural environment.

Viewing Natural Code Coverage Data

The contents of the code coverage resource file can be displayed in the **Code Coverage View**, either directly from within the corresponding Natural Development Server in the **Natural Server** view or from within a workspace project.

- Viewing the Results from within the Natural Development Server
- Viewing the Results from within the NaturalONE Project
- Natural Code Coverage View
- Natural Source Editor

Viewing the Results from within the Natural Development Server

To view the results from within the Natural Development Server

1 Select the code coverage resource file in the **Natural Server** view.

2 Open the context menu and choose **Open with Code Coverage**.

Note: With this approach only the **Code Coverage** view can be opened. Opening individual objects displayed in the **Code Coverage** tree view directly in the **Natural Source Editor** is currently not possible.

Viewing the Results from within the NaturalONE Project

To view the results from within the NaturalONE project

1 Transfer the code coverage resource file to a new project containing the application sources:

- Select the code coverage resource file in the **Natural Server** view.
- Open the context menu and choose **Add to New Project**

Or:

Transfer the code coverage resource file to an existing project containing the application sources:

- Select the code coverage resource file in the **Natural Server** view.
- Open the context menu and choose **Add to Existing Project**

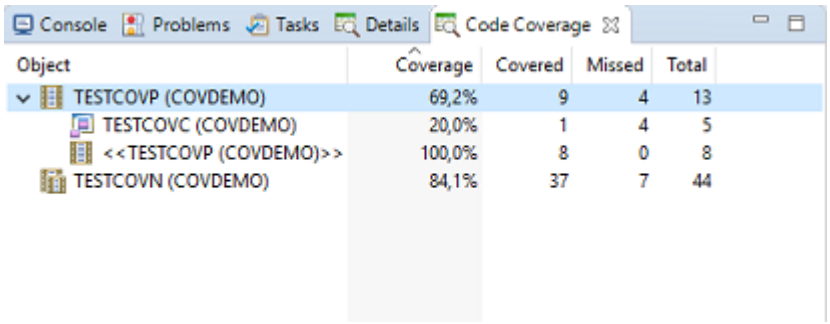
2 Select the NaturalONE project.

3 Open the context menu and choose **NaturalONE > Open with Code Coverage**.

Natural Code Coverage View

As a result of the previously used command the Natural **Code Coverage** view is opened which provides an overview of the covered code in the Natural application.

When copycode is used in a Natural object, the object node in the **Code Coverage** view also shows the copycode hierarchy with the distribution of the executed statements (in the **Coverage** column) in the copycode and the object itself:

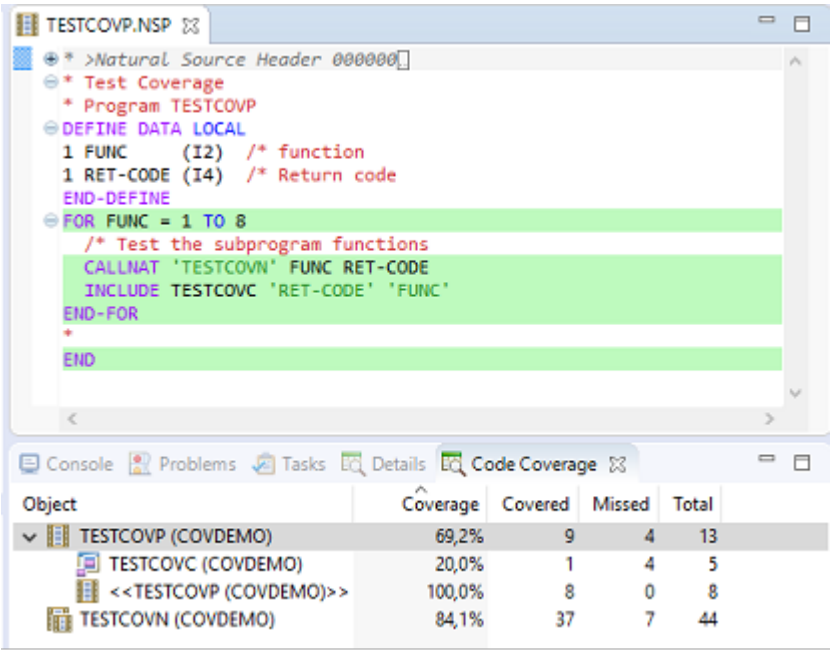


Object	Coverage	Covered	Missed	Total
TESTCOVP (COVDEMO)	69,2%	9	4	13
TESTCOVC (COVDEMO)	20,0%	1	4	5
<<TESTCOVP (COVDEMO)>>	100,0%	8	0	8
TESTCOVN (COVDEMO)	84,1%	37	7	44

For example, the code coverage result of the sample application above shows that the copycode TESTCOVC executed 20% of the statements available in the copycode whereas the object itself (enclosed in the << >> brackets) without the copycode executed 100% of the statements. The code coverage of the object including the copycode is finally 69,2%.

Natural Source Editor

To obtain more detailed information on the line coverage of a specific source displayed in the **Code Coverage** view, the Natural Source Editor can be used. Double-clicking on the Natural object or using the context menu **Edit** command on the Natural object will open the Natural Source editor showing the lines covered.



The background color green will be applied to the following source lines:

- Lines which are fully covered.
- Lines containing more than one statement where at least one statement is covered.
- The first line of statements spread over several lines.

The source code coloring disappears when the source file is edited and/or marked as modified.

The source code coloring is only valid when the Natural source file in the editor is identical to the one used to create the *.ncvf* file.

27 Using the Natural Coverage Plugin for Jenkins

- Principle of Operation 410
- Prerequisites 410
- Installation 410
- Configuration 411
- Using the Plug-In 411

Natural and NaturalONE provide code coverage functionality. Using code coverage, it is possible to exactly examine the statements of a Natural application that have been executed during a specific run and those that have not been executed. Tests can thus be improved so that they raise the number of touched statements of a Natural application when executed.

In many situations the Jenkins automation server is used for controlling the build and testing sequences. With the Natural Coverage Plugin for Jenkins, you can visualize the outcome of a Natural coverage cycle directly in the Jenkins job result pages. The plugin comes with the following features:

- Display a trend graph reflecting the changes in coverage over the last builds.
- Display a detailed view of all executed Natural objects and their covered and missed number of statements during the build.
- Support the detailed view also throughout the past builds in the job's build history.

Principle of Operation

When running a Natural session with the coverage functionality enabled, a result file (so-called Natural coverage file) with extension *.ncvf* is generated. This result file contains the collected events that describe the statements in a Natural application and the information about the exact statements that have been executed during the Natural session.

The Natural Coverage Plugin enables an existing Jenkins job to read such *.ncvf* files and to visualize the information stored within. Thus, it is possible to view the coverage of a Natural session from a higher per-application level down to a per-object level. However, for visualizing an even more detailed source code level the usage of NaturalONE is mandatory.

Prerequisites

Jenkins Version 1.625.3 or above must be available to run the Natural Coverage Plugin. The *.ncvf* files to be processed must be in ASCII format.

Installation

The Natural Coverage Plugin is provided with the file *NaturalCoverage.hpi* in the *naturalone/jenkins* subdirectory of a NaturalONE installation. In order to install the plugin perform the following steps:

1. Login as a Jenkins user who is allowed to install plugins.
2. In the **Manage Jenkins / Manage Plugins** configuration page select the **Advanced** section.

3. In the **Upload Plugin** dialog section browse to the *NaturalCoverage.hpi* file and press the **Upload** button.
4. Follow the instructions that are given by Jenkins.
5. When the installation has finished the new plugin **Natural Code Coverage Publisher** should be visible in the **Installed** view of the **Manage Plugins** page.

Configuration

An existing Jenkins job can be configured to post-process *.ncvf* files by performing the following steps:

1. Login as a Jenkins user who is allowed to configure Jenkins jobs.
2. On the job main page select the **Configure** action.
3. In the **Post-build Actions** section at the end of the job configuration page select the **Add post-build action** box.
4. From the available actions select **Natural code coverage publisher**. If the action is not available, check if the installation of the plugin has been performed successfully.
5. In the new dialog the *.ncvf* file path can be entered. Base for all file paths is the Jenkins workspace directory.
6. The new post-build action gets executed when running the next build.

Using the Plug-In

When the **Natural code coverage publisher** has been activated and when the *.ncvf* files have been detected at the specified location, a trend graph will be shown on the job's main page. Please note that the trend graph requires at least two consecutive builds to be correctly drawn. That same graph on a separate page is also shown when selecting the **Coverage Trend Graph** link on the left hand side of the job page.

When moving the mouse pointer across the graph, some base information will pop up as tool tips. Clicking inside the graph within the range of some specific build will open a detailed result page for this specific build. The same result page can be reached by selecting the link **Coverage Build Result** from within a specific build result page.

XII Using Natural Tools and Utilities

28 Using Natural Tools and Utilities

- General Information 416
- Starting Natural Tools and Utilities 417
- SYSUTIL Utility 417
- Rich GUI interface of SYSEXT 420
- Rich GUI interface of SYSEXV 423
- Message Retrieval 425
- Extend the List of Tools and Utilities 427
- Return Control to Natural Tools and Utilities 431

This section describes the SYSUTIL utility, the new rich GUI interfaces of the SYSEXT and SYSEXV utilities and the Message Retrieval utility.

General Information

The SYSUTIL utility is used to start selected Natural tools and utilities from NaturalONE. It may also be used to start user-defined tools. The tools are executed on the server side whereby any kind of server (mainframe, UNIX, Windows) is supported.

In addition to that, it can be used to open web pages from NaturalONE.

Restrictions

- For technical reasons, only rich GUI (Natural Ajax) and character-based tools are supported.

  Tools using the dialog interface are not supported.
- Tools which are implemented in NaturalONE such as SYSPROD or SYSPROF, are not provided by SYSUTIL.

Prerequisites

- Web I/O must be enabled on the Natural server. Otherwise, the output of a character-based tool cannot be displayed.
- The `PARSE XML` statement has to be enabled in a mainframe environment. Use the following parameter:

  ```
  XML=(ON,PARSE=ON)
  ```

  For further information, see the `XML` parameter description in the *Parameter Reference* section of the *Natural for Mainframes* documentation.

  On non-mainframe environments, the `PARSE` statement is enabled by default.
- Natural Version 9.1 or higher is required on the server side to run the Natural tools and utilities from NaturalONE. Otherwise you will receive a message like

  ```
  NAT0082 Invalid command, or Program SYSUTIL does not exist in library.
  ```

Starting Natural Tools and Utilities

To start the Natural tools and utilities

1 In your workspace, select the Natural project, library or object from which you want to start the Natural tools and utilities.

2 Invoke the context menu and choose **NaturalONE > Tools and Utilities**.

Or:

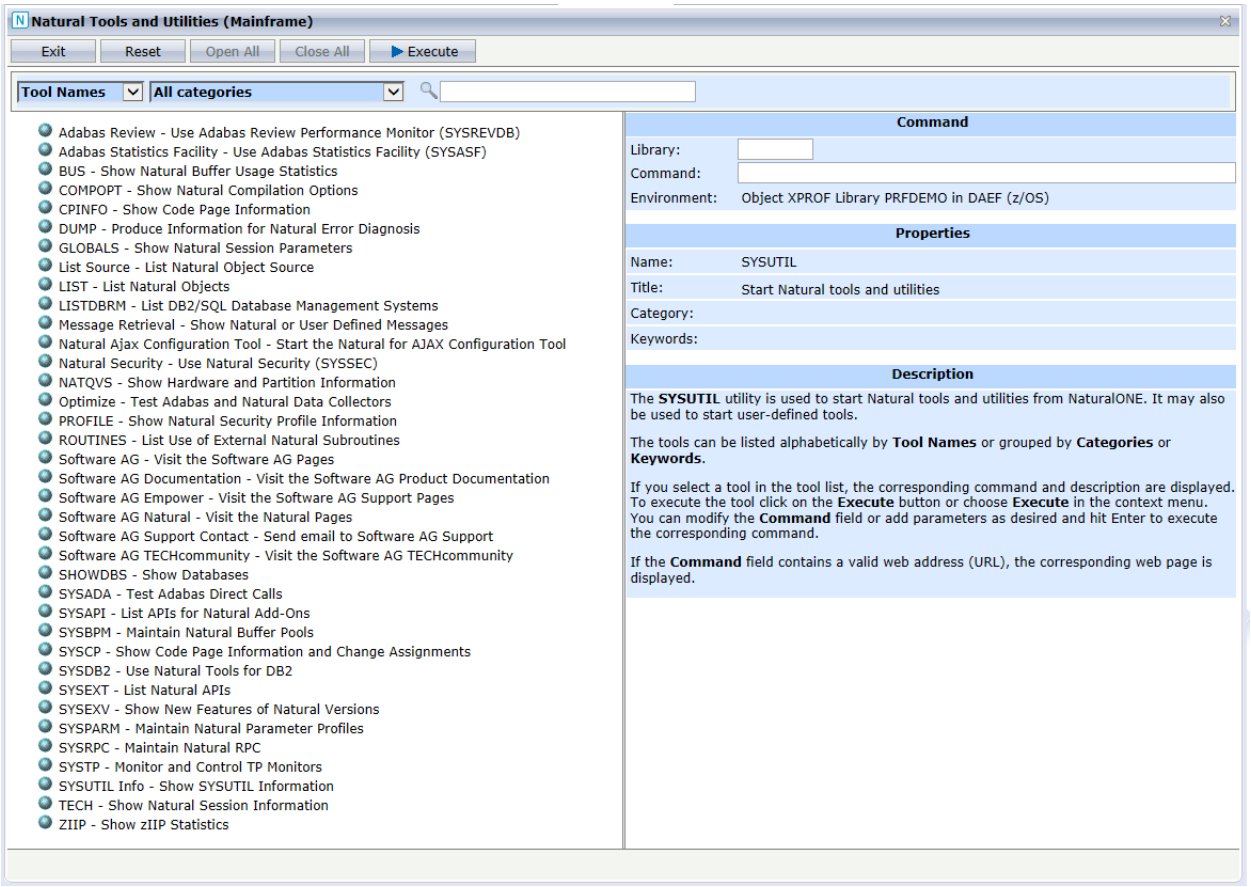
Press Alt+U.

This starts the SYSUTIL utility from which you can start selected Natural tools and utilities.

SYSUTIL Utility

The SYSUTIL utility lists Natural tools and utilities which can be executed from NaturalONE.

If Natural Security is installed in your environment, the list only displays tools you are authorized to use.



The tools can be listed alphabetically by **Tool Names** (default) or grouped by **Categories** or **Keywords**. By default, the list shows the tools of all categories. A specific category can be selected so that only tools belonging to this category are listed.

You can specify a search string so that only the tools containing the string in the name, title, category or a keyword are shown.

The buttons in SYSUTIL have the following meaning:

Button	Description
Exit	Exit SYSUTIL.
Reset	Reset all fields to their initial state.
Open All	Open all nodes in the tree. Only enabled if the tree is grouped by categories or keywords.
Close All	Close all nodes in the tree. Only enabled if the tree is grouped by categories or keywords.
Execute	Execute a selected tool. Works only if a tool has been selected or a command has been specified.

The entries on the right side of SYSUTIL have the following meaning:

Entry	Description
Library	SYSUTIL issues a LOGON to this library before it executes the command. If no library is specified, the command is executed in the library from where SYSUTIL has been started.
Command	The command or program to be executed. If the field contains a valid web address (URL), the corresponding web page is opened in a new window.
Environment	The environment from where SYSUTIL has been started.
Name	The name of the tool.
Title	The descriptive name of the tool.
Category	The category to which the tool belongs.
Keywords	Keywords associated to the tool.
Description	A short description of the functionality of the tool.

To start a specific Natural tool or utility

1 Select a tool in the SYSUTIL tree. The **Library** and **Command** fields are filled according to your selection.

2 Add parameters to the command as desired.

3 Click on **Execute**.

Or:

1. Double-click the tool entry in the SYSUTIL tree.

Or:

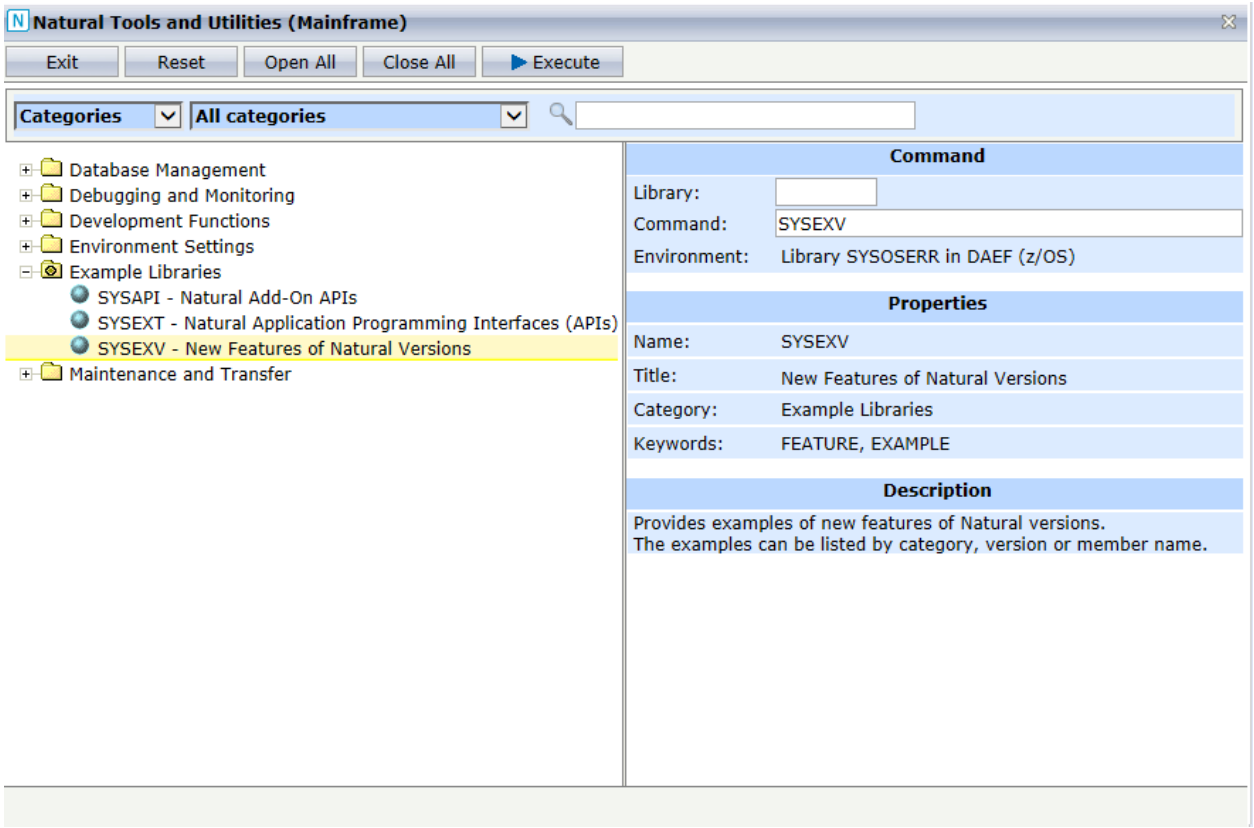
1. Select a tool in the SYSUTIL tree.
2. Invoke the context menu and choose **Execute**.

Or:

1. Specify a library and a command (including parameters if desired) in the corresponding fields.
2. Press Enter or click on **Execute**.

Example

In the following example, the Natural tools and utilities are grouped by categories and the SYSEXV utility in the **Example Libraries** category has been selected. The entries on the right side are filled with the values of the SYSEXV utilities. The utility will be started by clicking on **Execute**.



Rich GUI interface of SYSEXT

The utility SYSEXT is used to locate and test Natural Application Programming Interfaces (APIs) contained in the current system library SYSEXT.

This section describes the rich GUI interface to SYSEXT from NaturalONE. For further information regarding the SYSEXT utility, refer to the section *SYSEXT Utility* in the Natural documentation corresponding to your platform.

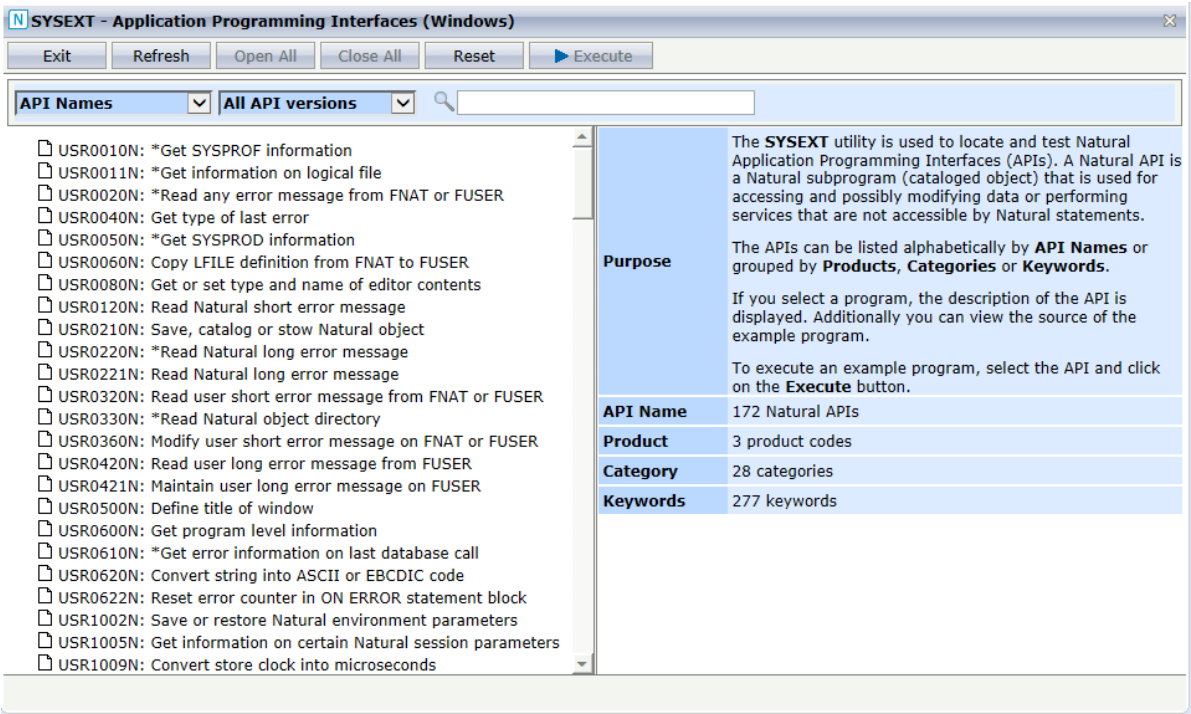
To start the SYSEXT utility

1 In your workspace, select any Natural project, library or object in the environment in which you want to start the SYSEXT utility.

2 Invoke the context menu and choose **NaturalONE > Tools and Utilities** to start the SYSUTIL utility.

3 Select **SYSEXT** in the tree and click on **Execute**.

The rich GUI interface of the SYSEXT utility is displayed.



The Natural APIs can be listed alphabetically by **API Names** (default) or grouped by **Products**, **Categories** or **Keywords**. By default, the list shows all versions of the APIs whereby previous versions of an API are indicated by an asterisk (*). If you want to see only the most recent version of an API in the list, select **Current API version**.

You can specify a search string so that only the APIs containing the string in the name, purpose, category or a keyword are shown.

The buttons in SYSEXT have the following meaning:

Button	Description
Exit	Exit SYSEXT.
Refresh	Update the API information. It reads all APIs and regenerates the input files which are used to fill the tree. This is only required if API descriptions have been changed.
Open All	Open all nodes in the tree. Only enabled if the tree is grouped by products, categories or keywords.
Close All	Close all nodes in the tree. Only enabled if the tree is grouped by products, categories or keywords.
Reset	Reset all fields to their initial state.
Execute	Execute the example program related to the API. Works only if an API has been selected.

On the right side of SYSEXT, the purpose, name, category and the keywords of a selected API are displayed. Additionally, you can choose whether you want to see the description of the API (member `USRnnnnT`) or the source of the corresponding example program `USRnnnnP`.

To start an example program of a Natural API

1 Select an API in the SYSEXT tree. The fields on the right side are filled according to your selection.

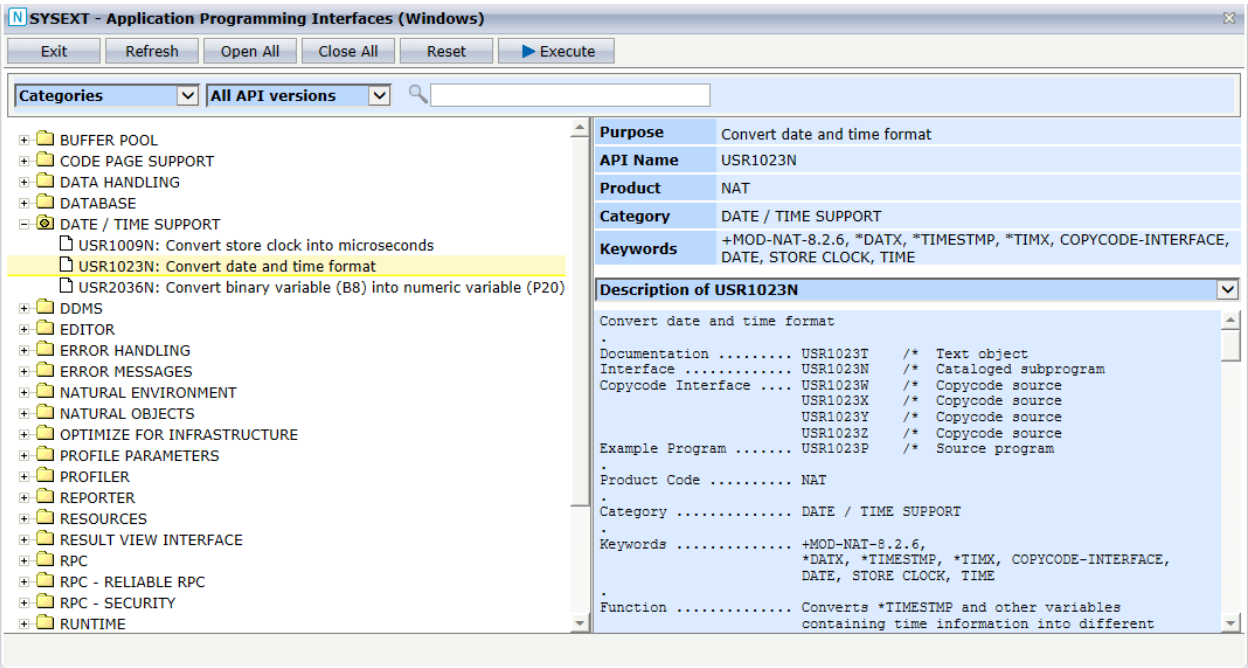
2 Click on **Execute**.

Or:

Double-click the API entry in the SYSEXT tree.

Example

In the following example, the APIs of SYSEXT are grouped by categories and the API `USR1023N` in the **DATE/TIME SUPPORT** category has been selected. The entries on the right side are filled with the values of the API `USR1023N`. The example program `USR1023P` will be started by clicking on **Execute**.



Rich GUI interface of SYSEXV

The utility SYSEXV provides example programs that demonstrate the use of Natural features introduced in the current or a previous version of Natural.

This section describes the rich GUI interface to SYSEXV from NaturalONE. For further information regarding the SYSEXV utility, refer to the section *SYSEXV Utility* in the Natural documentation corresponding to your platform.

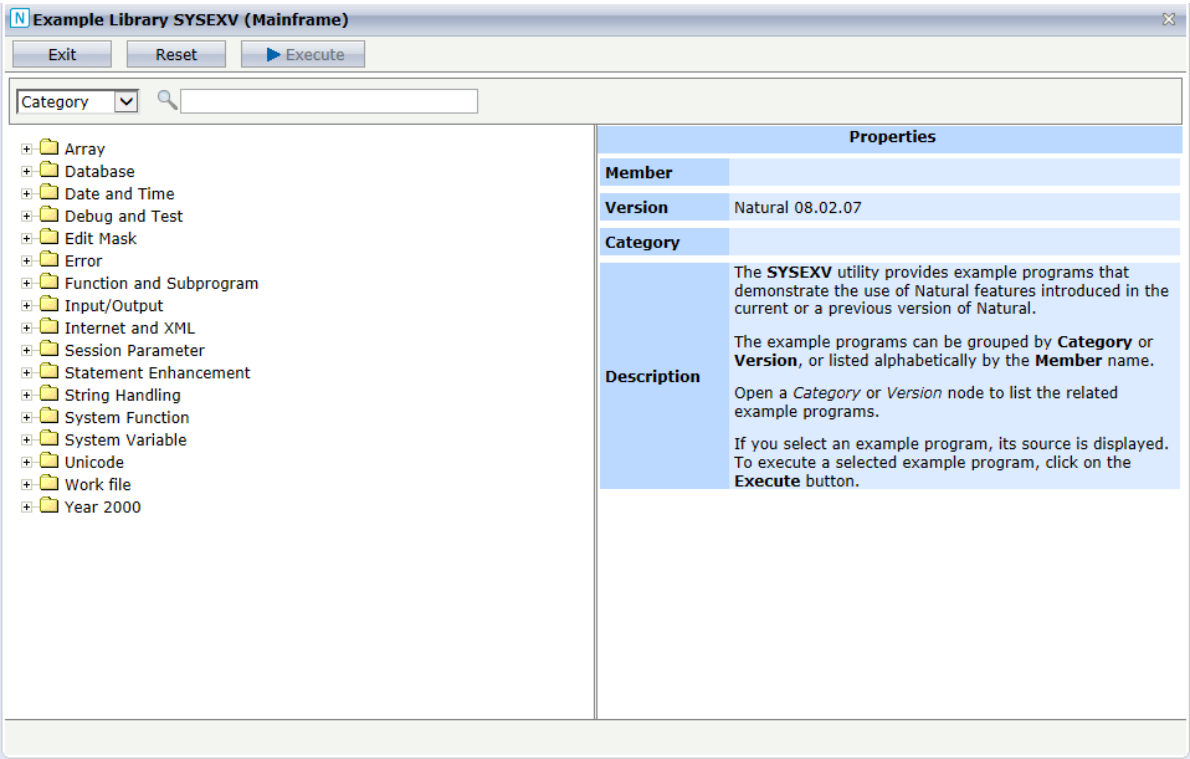
To start the SYSEXV utility

1 In your workspace, select any Natural project, library or object in the environment in which you want to start the SYSEXV utility.

2 Invoke the context menu and choose **NaturalONE > Tools and Utilities** to start the SYSUTIL utility.

3 Select **SYSEXV** in the tree and click on **Execute**.

The rich GUI interface of the SYSEXV utility is displayed.



The example programs can be listed alphabetically by **Member** names or grouped by **Category** (default) or **Version**. You can specify a search string so that only the example programs containing the string in the member name, version, category or description are shown.

The buttons in SYSEXV have the following meaning:

Button	Description
Exit	Exit SYSEXV.
Reset	Reset all fields to their initial state.
Execute	Execute the example program. Works only if an example program has been selected.

On the right side of SYSEXV, the member name of the example program, the version in which the Natural feature was introduced, the category and the description of the Natural feature are displayed. Additionally the source of the example program is provided.

To start an example program

1 Select an example program in the SYSEXV tree. The fields on the right side are filled according to your selection.

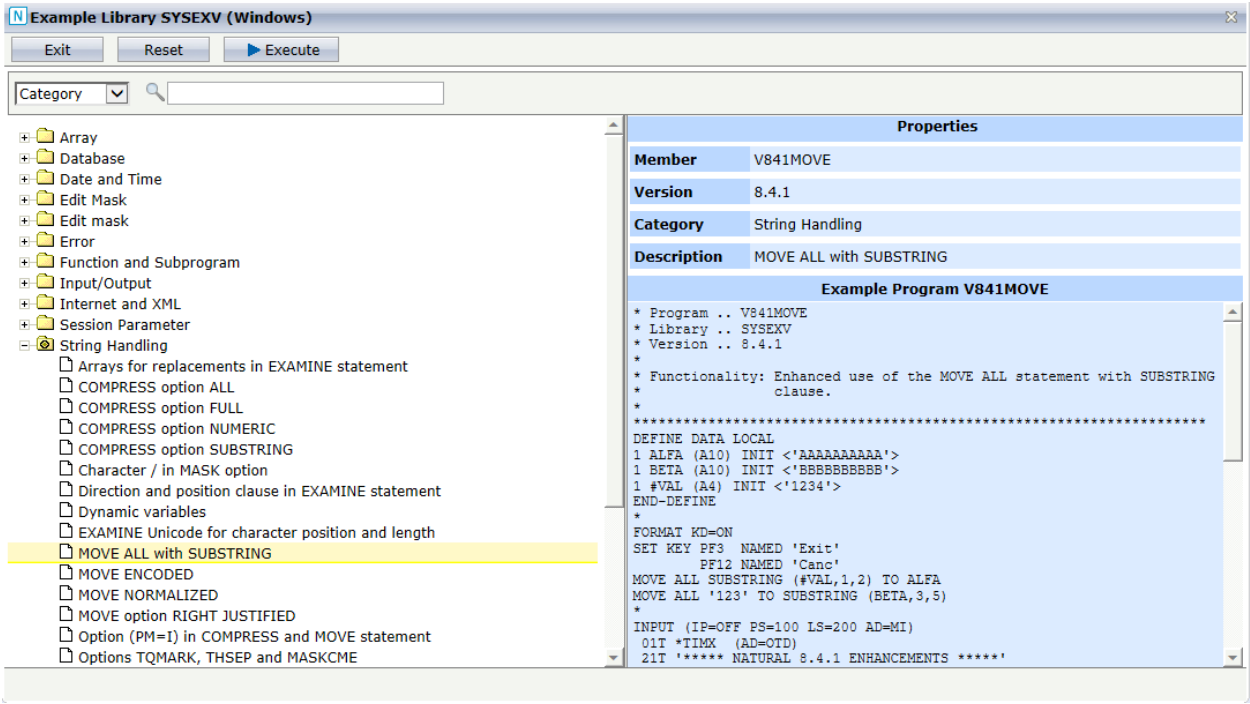
2 Click on **Execute**.

Or:

Double-click the example program entry in the SYSEXV tree.

Example

In the following example, the example programs of SYSEXV are grouped by categories and the example program for the **MOVE ALL with SUBSTRING** feature in the **String Handling** category has been selected. The entries on the right side are filled with the values of the example program `V841MOVE`. The example program will be started by clicking on **Execute**.



Message Retrieval

The SYSUTIL **Message Retrieval** function is used to read Natural or user defined messages.

To start the Message Retrieval

1. In your workspace, select any Natural project, library or object for which you want to start the message retrieval. If you have selected a library or an object in a library, this library is taken as default library for user defined messages.
2. Invoke the context menu and choose **NaturalONE > Tools and Utilities** to start the SYSUTIL utility.
3. Select **Message Retrieval** in the tree.
4. Add parameters to the command as desired.
5. Click on **Execute**.

   The Message Retrieval is displayed.

The SYSUTIL Message Retrieval function offers the following parameters:

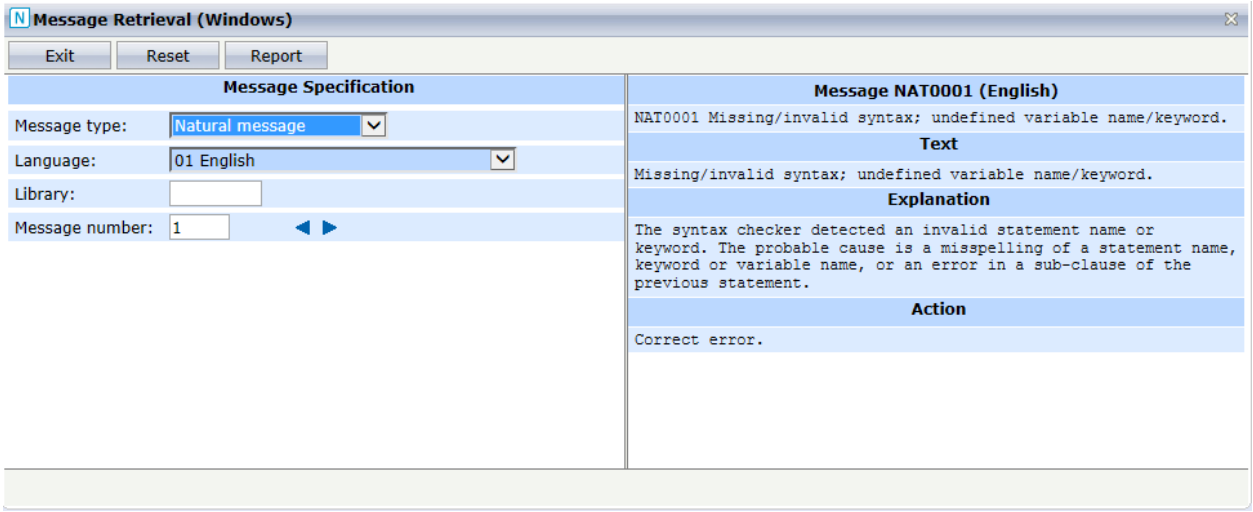
```
SYSUTIL -MSG [nnnn [library]]
```

Where *nnnn* is the message number and *library* is the library containing the user defined message.

Alternatively, you can start the SYSUTIL **Message Retrieval** with the following command:

```
? [nnnn [library]]
```

If the message number and the library are specified, the user defined message *nnnn* from library *library* is displayed. If only the message number is specified, the Natural message *nnnn* is displayed. If no parameter is given, the Natural message NAT0001 is shown initially.



The buttons in Message Retrieval have the following meaning:

Button	Description
Exit	Exit Message Retrieval.
Reset	Reset all fields to their initial state.
Report	A preview of the message is generated. It can be used for printing the message or converting it into a PDF member.

The entries on the left side of the Message Retrieval have the following meaning:

Entry	Description
Message type	Select the message type: Natural message or User defined message. If you view Natural messages, the **Library** field is cleared. If you view user defined messages, the **Library** field is filled with the previously used user library or initially, with the library from which you have started the Natural tools and utilities.
Language	Choose the language of the message.
Library	If you clear this field, Natural messages are displayed. Enter a library name to view the user messages of the given library.
Message number	The number of the message to be displayed. Click on the left or right arrow to show the previous or next message.

On the right side of the Message Retrieval, the short and long text of the message is displayed.

Example

The following example shows the PDF output of the Natural message 82.

Natural Message Retrieval (Windows)

Message NAT0082 (English)
NAT0082 Invalid command, or :1: :2: does not exist in library.
Text
Invalid command, or ... ... does not exist in library.
Explanation
One of the following has occurred:
- You entered a value in the command line which is neither a
Natural command nor the name of a Natural program contained
in the active library or in a library defined as a steplib.
- An object which is required during processing of a Natural
executable is not contained in the active library or in
a library defined as a steplib.
- Your Natural session is currently applying system files other
than those containing the object you specified.
Action
Enter a valid Natural command or the name of an existing Natural
object. Use the command SYSPROF to check whether you are using the
correct system file.

Extend the List of Tools and Utilities

The SYSUTIL utility can be extended to also list user defined tools in the tree.

To extend the Natural tools and utilities

1 Create an XML member according to the SYSUTIL standards.

2 Name it either `TOOL-*.XML` or `USER-uid.XML` where the asterisk (*) stands for any character combination and *uid* for a user ID.

3 Save the XML member in the *Resource* folder of the SYSUTIL library.

`TOOL-*.XML` contains customer specific tools. It is visible to all who share the same FNAT system file.

`USER-uid.XML` contains user specific tools. It is only visible to the user with the `*INIT-USER` user ID *uid*.

Notes:

1. `SAG-*.XML` contains Software AG tools. Do not modify these members.
2. If you are using Natural Security, define the system library SYSUTIL in Natural Security.
3. The resource `EXAMPLE.XML` in the library SYSUTIL contains an example XML. You may use it as template for creating your own XML members.

The following shows the standard layout of a SYSUTIL XML member:

```
<?xml version='1.0' ?>
<SYSUTIL VERSION='version' USAGE='usage'>
  <COMMENT>comment</COMMENT>
  <TOOL>
    <NAME>name</NAME>
    <TITLE>title</TITLE>
    <LIBRARY>library</LIBRARY>
    <COMMAND>command</COMMAND>
    <CLEARSCREEN>clear-screen</CLEARSCREEN>
    <CATEGORY>category</CATEGORY>
    <KEYWORDS>keywords</KEYWORDS>
    <DESCRIPTION>description</DESCRIPTION>
    <MACHINE-CLASS>machine-class:filter</MACHINE-CLASS>
  </TOOL>
  <TOOL>
    ...
  </TOOL>
</SYSUTIL>
```

The entries in the XML member have the following meaning:

Tag Value	Description
`version`	The version of the XML member. The entry is not interpreted by SYSUTIL.
`usage`	A short internal description of the XML member. The entry is not interpreted by SYSUTIL.
`comment`	More detailed internal description of the XML member. The tag is not interpreted by SYSUTIL.
`name`	The name of the tool. It is used as main index in the list of tools. Maximum size: 32 characters.
`title`	A descriptive title of the tool. It is used as supplement to the tool name in the list of tools. Maximum size: 61 characters.

Tag Value	Description
`library`	SYSUTIL issues a `LOGON` to this library before it executes the command. Default: If started on a project level: SYSTEM Otherwise: Current library
`command`	The command, program or URL to be executed. Parameters may be added separated by blanks. The following special values can be used in the command: ■ `$PGM$` is replaced by the current program or an asterisk (*) if none is selected. ■ `$LIB$` is replaced by the current library or a blank if none is selected.
`clear-screen`	Whether SYSUTIL clears the screen before command execution. This may be set to `TRUE` if a WEBIO application overwrites only a part of the screen. Possible values: `TRUE` or `FALSE`. Default: `FALSE`
`category`	The category to which the tool belongs. If appropriate, use an existing category of SYSUTIL.
`keywords`	Keywords associated to the tool. The keywords are written in upper case and separated by commas. If appropriate, use existing keywords of SYSUTIL.
`description`	The long description of the tool in HTML format. To avoid that special characters of HTML are interpreted by XML, replace the following characters: ■ `&` with `&` ■ `<` with `<` ■ `>` with `>` Alternatively, you can save the description in HTML format (without replacements) in a UTF-8 resource in the SYSUTIL library. Start the DESCRIPTION value with "HTML:" followed by the name of the HTML resource. For example: *HTML:MyDesc.htm*
`machine-class:filter`	`machine-class` The machine classes on which the tool is available. Specify the first characters of the machine classes (*MACHINE-CLASS) on which the tool is available. ■ M - Mainframe ■ P - PC (Windows) ■ U - UNIX (and LINUX) If the tool is available on all platforms, you may also use ■ A - All Default: A

Tag Value	Description
	`filter` The following list shows the available filters and the one-letter codes that are entered in the list of tools if one of the following applies: `L`: SYSUTIL was started from a library or the context menu of a Natural object. `P`: SYSUTIL was started from the context menu of a Natural object. `S`: Log on to the given library and execution of the given command are allowed by Natural Security (if appropriate). `X`: A cataloged program with the name of the `command` exists in the given library. If no filter is used, the colon (`:`) between machine-class and filter can be omitted.

Example

The following example shows the example XML member `TOOL-Admin.XML` for SYSUTIL. If you save it in the SYSUTIL resource folder, the `ReadCust` application will be added to the list of tools. The one-letter filter code (here: `S`) in the `MACHINE-CLASS` tag has the effect that the ReadCust application is only shown in the list of tools if the user is allowed by Natural Security to execute the application.

```
<?xml version='1.0' ?>
<SYSUTIL VERSION='1.0' USAGE='SYSUTIL entry for READCUST'>
  <COMMENT>

Customer data administration

  </COMMENT>
  <TOOL>
    <NAME>ReadCust</NAME>
    <TITLE>Read customer data</TITLE>
    <LIBRARY>CUSTLIB</LIBRARY>
    <COMMAND>READCUST</COMMAND>
    <CLEARSCREEN>FALSE</CLEARSCREEN>
    <CATEGORY>Administration</CATEGORY>
    <KEYWORDS>DATABASE,READ,CUSTOMER</KEYWORDS>
   <DESCRIPTION>Read all &lt;strong&gt;customer&lt;/strong&gt; data.</DESCRIPTION>
    <MACHINE-CLASS>A:S</MACHINE-CLASS>
  </TOOL>
</SYSUTIL>
```

Return Control to Natural Tools and Utilities

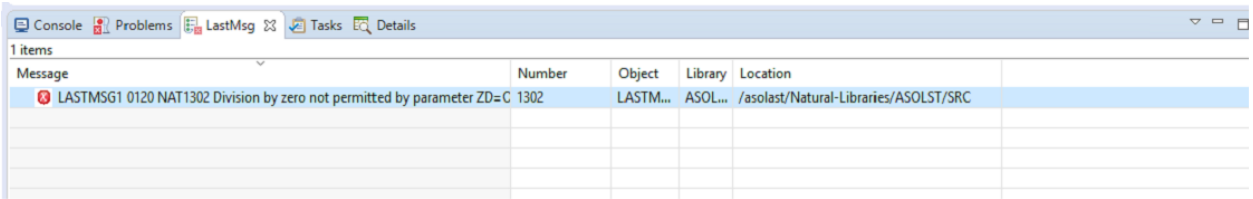
If Natural Tools and Utilities has been extended to call a user defined tool, it is a desired behavior that after the user tool has finished, control is given back to SYSUTIL. Therefore, SYSUTIL places itself on the Natural stack including all settings so that it is restarted automatically when the user defined tool ends. All selections and field contents of SYSUTIL are restored to the state it was in before the user defined tool had been called.

If the user defined tool releases the Natural stack with a `RELEASE STACK` command by any reason, the approach described above will not work. In this case, the tool may issue the statement `STACK TOP COMMAND 'RETURN'`. This will restart SYSUTIL, but all selections and field contents will be set to initial state. The information regarding the library and object selection will also be lost.

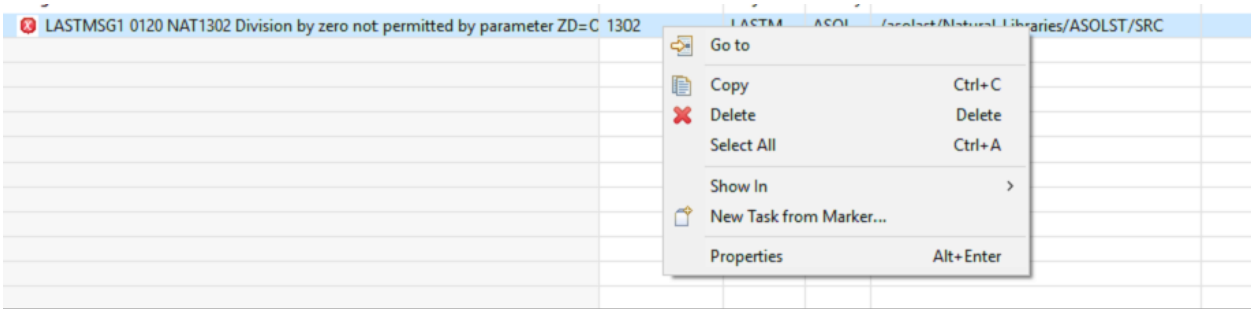
XIII Using the LastMsg View

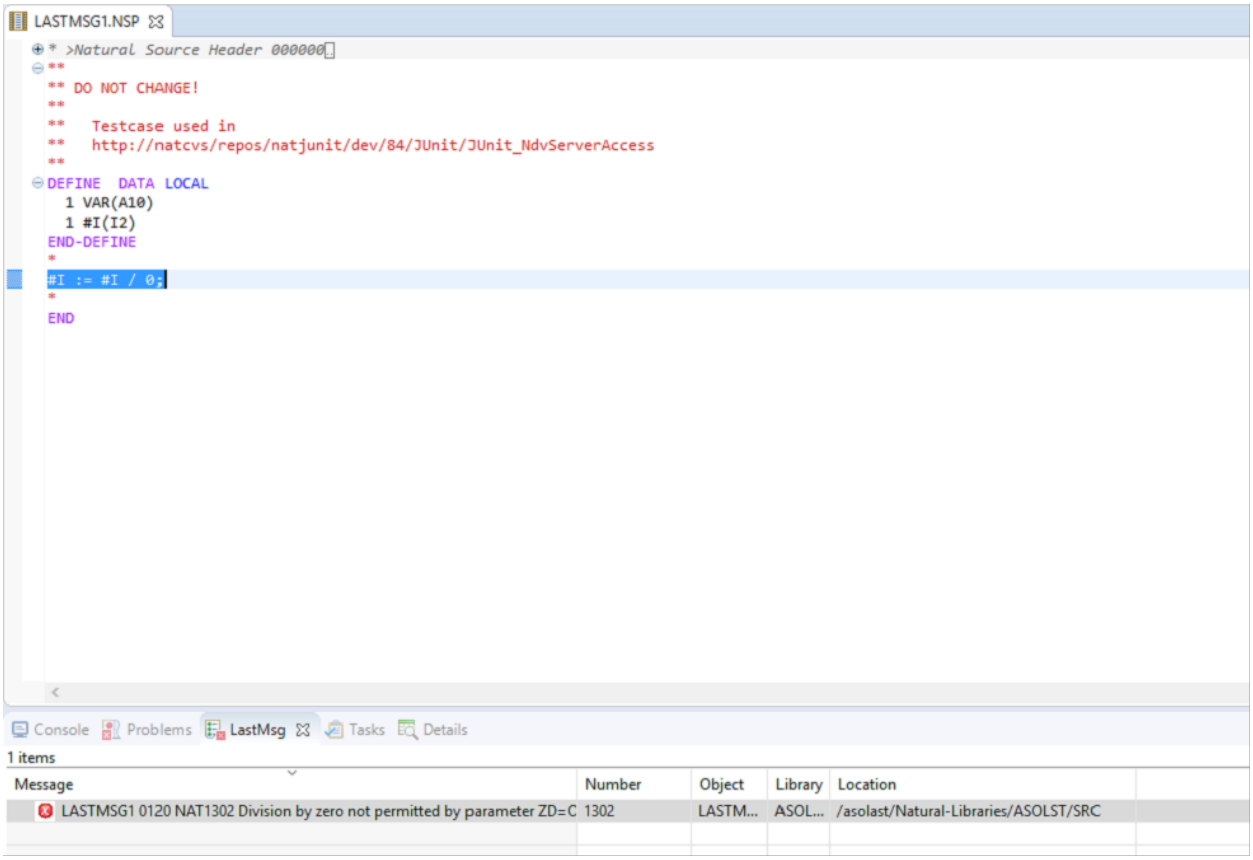
29 Using the LastMsg View

Any time a Natural related command is executed inside the NaturalONE workspace or inside the Natural server view, resulting error messages are transferred from the used runtime environment to the LastMsg view. If for example a program was executed in the runtime environment with the workspace command **Run as -> Natural application** the resulting runtime error(s) will be added.



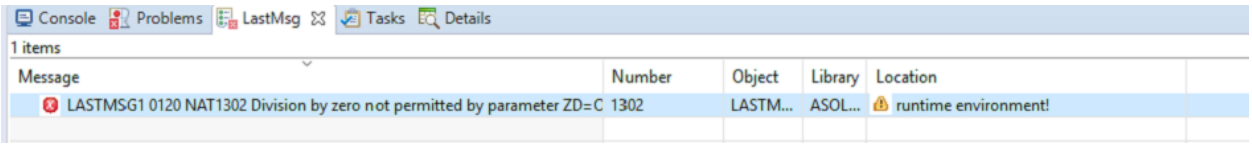
If the command is triggered from the workspace and the sources involved in the error situation are available in the corresponding Natural project (or in a referenced project), the **Go To** command or a double click on the corresponding error message will open the source and mark the line where the error occurred.



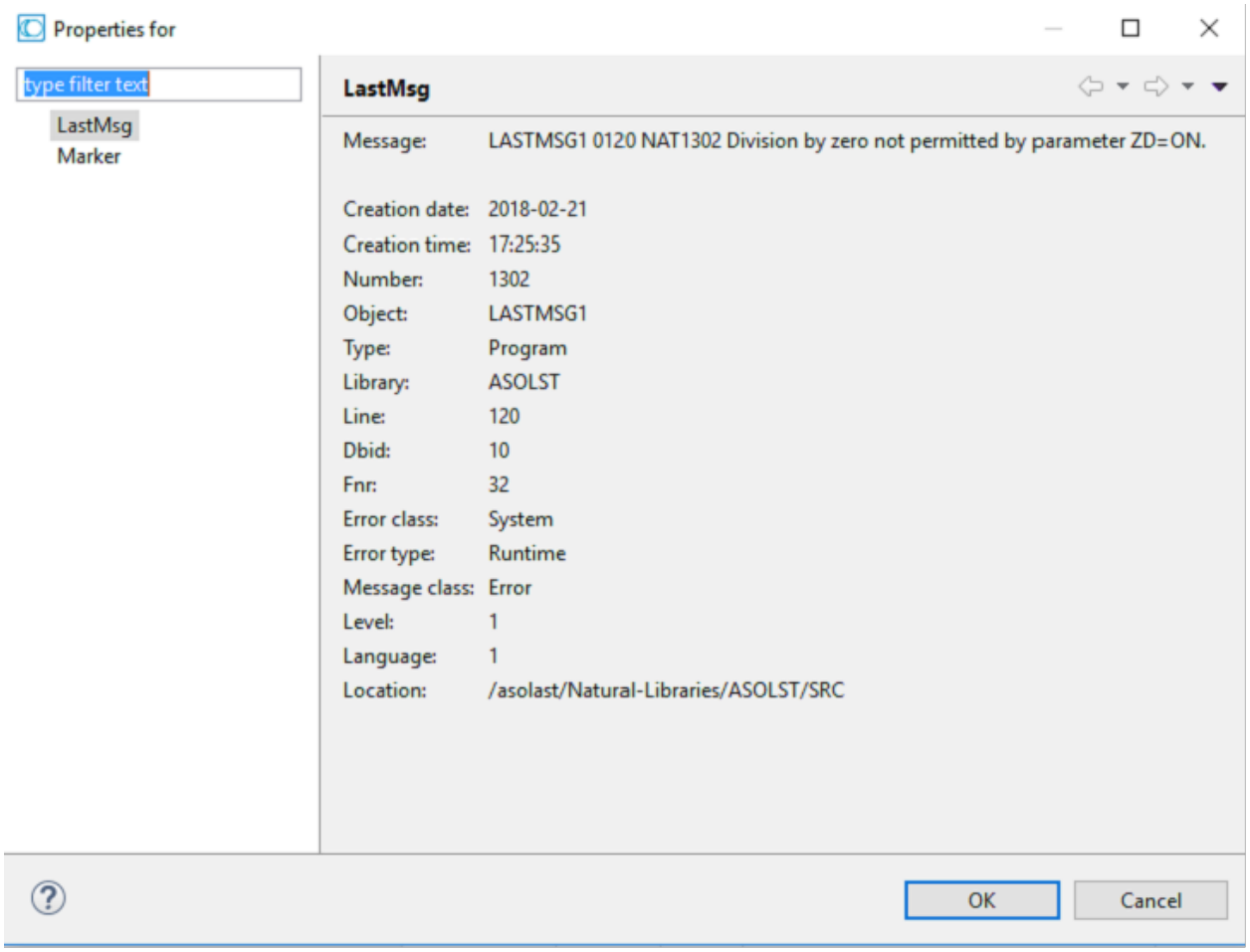


Hovering over the blue indicator on the annotation ruler on the left hand side of the Natural Source Editor causes the error message to be displayed in a tooltip window.

If the command is triggered from the Natural server view, for example using the command **Execute**, navigation to the source is not possible. This scenario is indicated with a corresponding warning ("runtime environment").

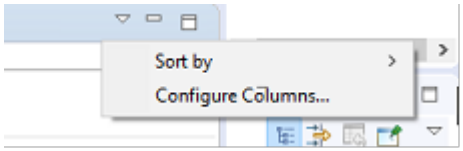


If you click on **Properties** in the context menu, all items of the selected error message will be shown:

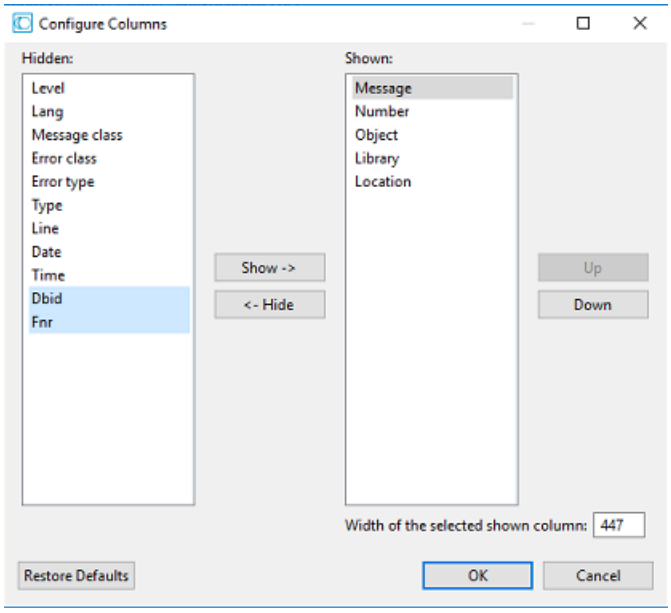


Since the LastMsg view is using the Eclipse resource marker technology, the error messages are persistent and still available after a restart of NaturalONE.

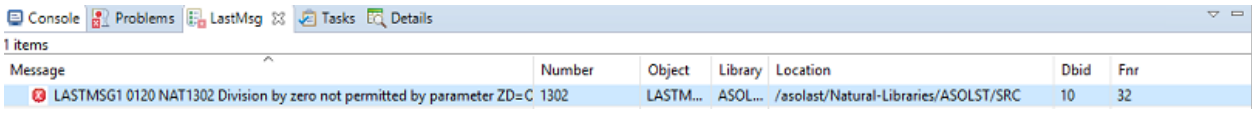
The columns of the LastMsg view can be configured with the **Configure Columns…** option.



For example, the **Dbid** and **Fnr** can be added to the columns of the LastMsg view using the **Show** option:



The LastMsg view then also displays the added items:



XIV Using the XML Toolkit

30 Using the XML Toolkit

- General Information 442
- Generating the Output Files 443
- Displaying the Settings of the Last Generation 446

General Information

The XML toolkit is the tool of choice when you want to generate aids for the processing of XML documents within Natural. It is a wizard which makes use of the objects in the Eclipse workspace. Different input files are possible:

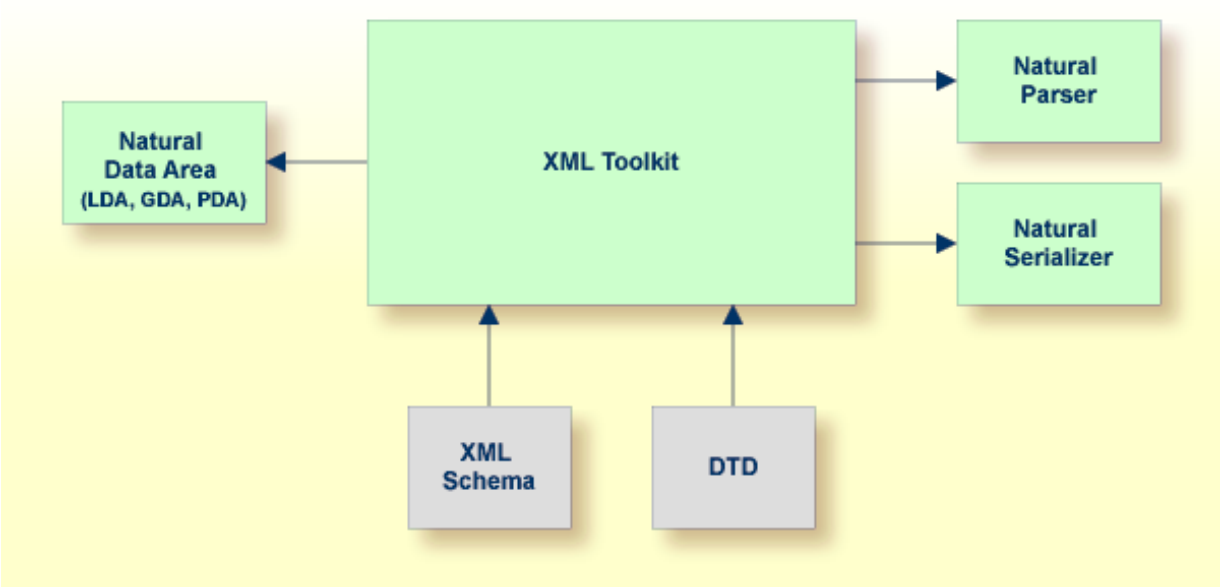
- XML schema.
- Document type definition (DTD).
- Natural data area (local data area, global data area or parameter data area).

Depending on the input file, the XML toolkit generates the appropriate output files:

- A parser implementation for the `PARSE` statement.
- A serializer implementation which uses the `COMPRESS` statement.
- A corresponding Natural data area.
- A corresponding XML schema.
- A corresponding DTD.

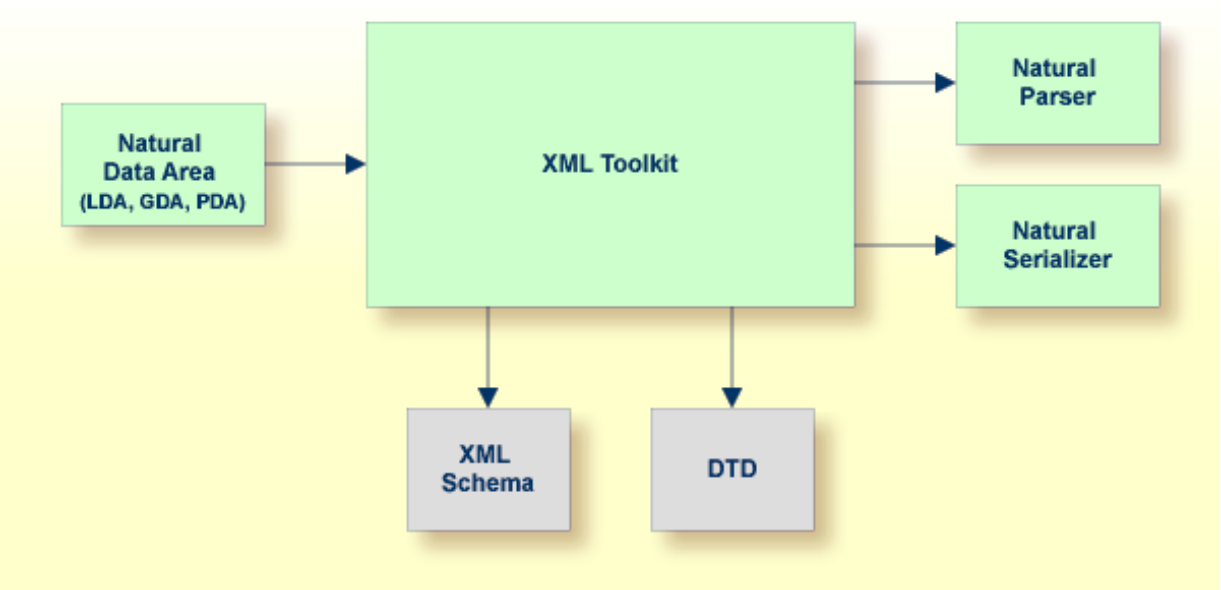
This is illustrated in the following graphics.

- **Input: XML Schema or DTD**
  When the input file is an XML schema or DTD, the XML toolkit can produce a Natural data area, a parser implementation and/or a serializer implementation.



Caution: Byte order marks (BOMs) in XML schema and DTD input files are currently not supported. If the input file contains a BOM, a parsing error will occur. As a workaround, you have to remove the BOM manually.

- **Input: Natural Data Area**
  When the input file is a Natural data area, the XML toolkit can produce a parser implementation, a serializer implementation, an XML schema and/or a DTD.



For more information on the XML toolkit, see *Web Technology* in the Natural documentation for the appropriate platform.

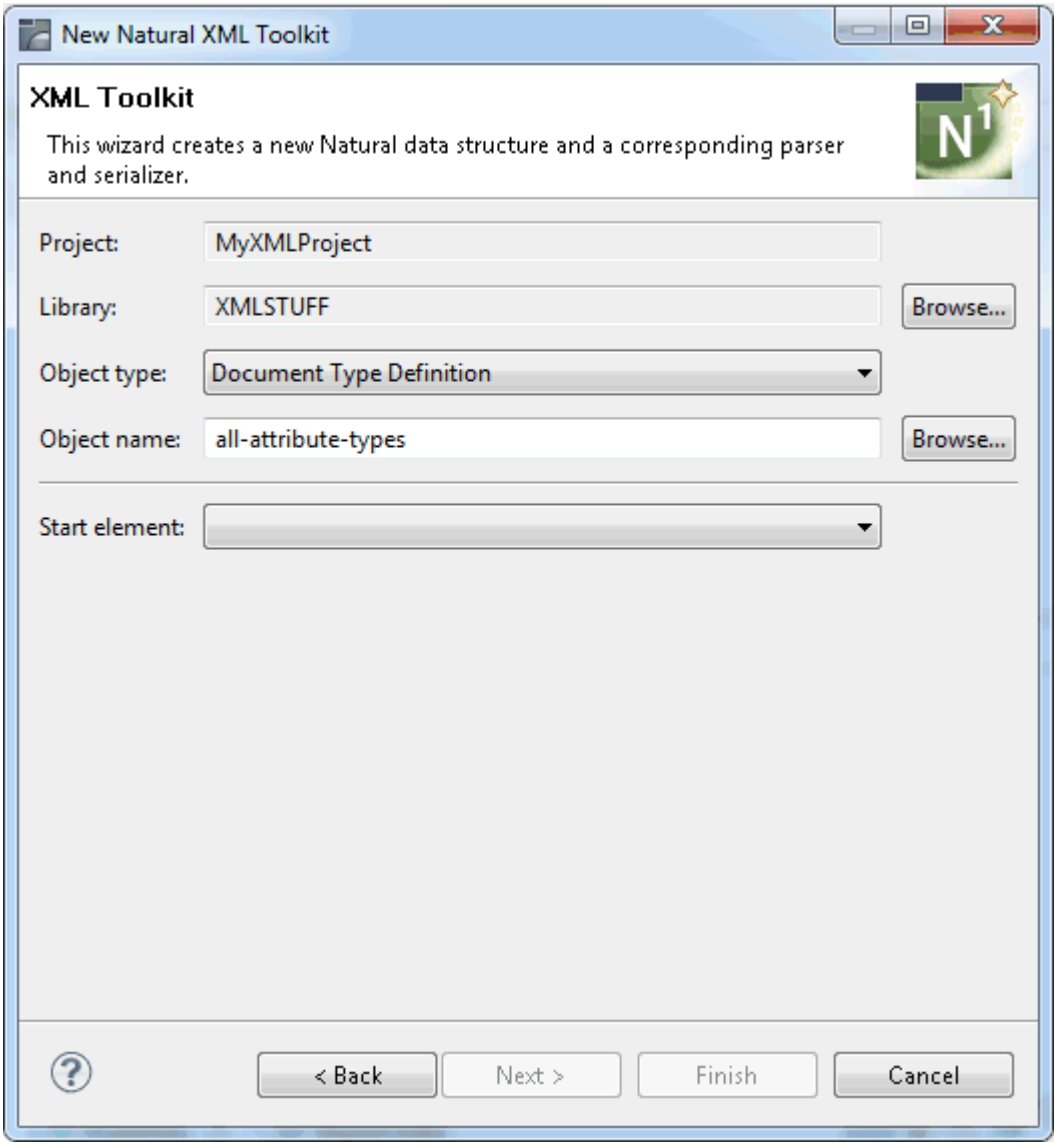
Generating the Output Files

In order to generate the above mentioned output files, you use the XML toolkit as described below.

To generate output files

1. In the **Navigator** view or in the **Natural Navigator** view, select the object that you want to use as the input file.
2. From the **File** menu, choose **New > Other**. In the resulting **New** dialog box, expand the **Natural** node, select **XML Toolkit** and then choose the **Next** button.

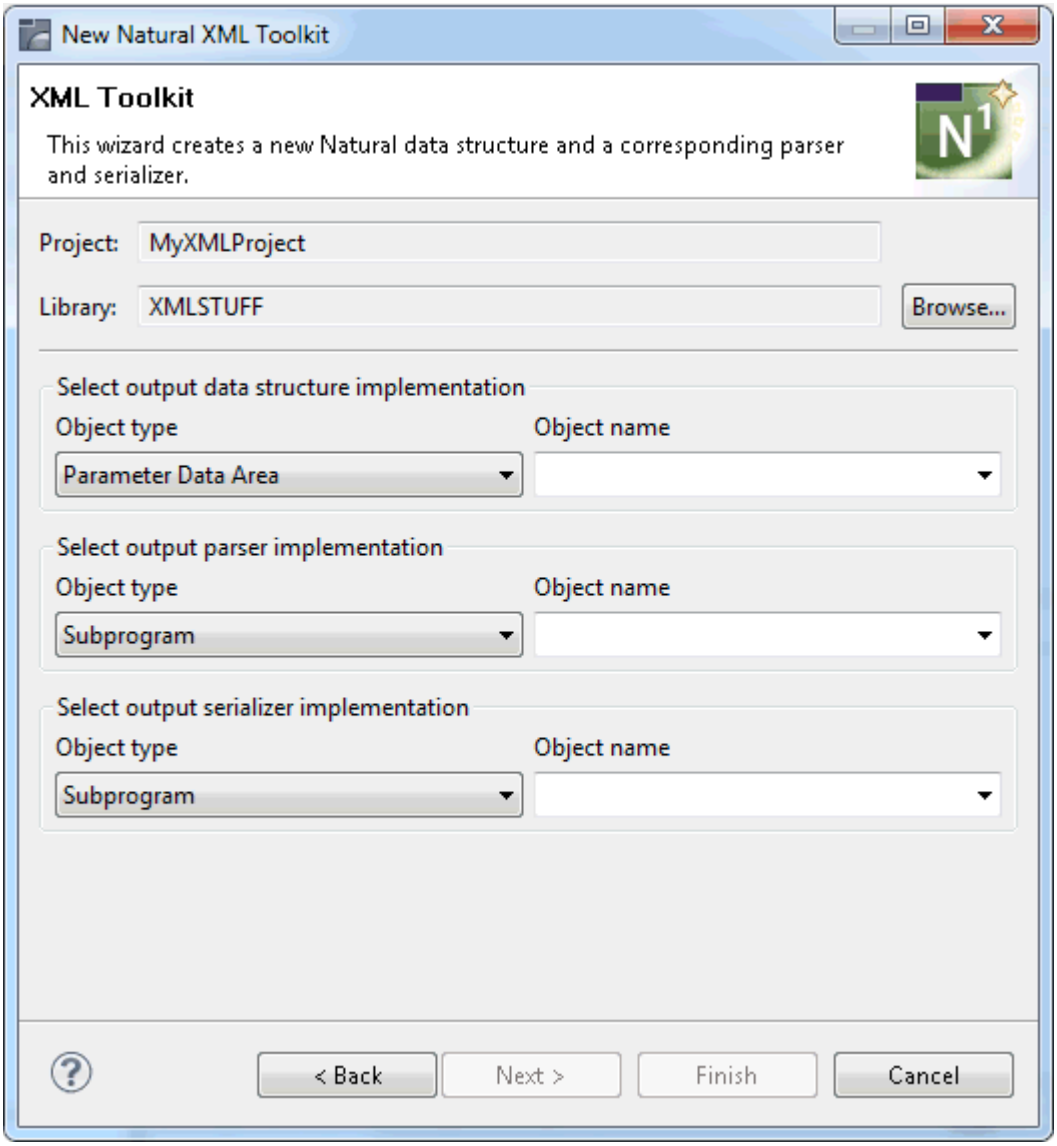
   The following dialog box appears (in this example, a DTD has been selected).



3 From the **Start element** drop-down list box, select the element that is to be used as the basis for the generated output file (for example, select the DTD element that is to be the basis for a new parameter data area).

4 Choose the **Next** button.

The following is now shown. You can create up to 3 different output files at the same time.



5 Enter a name for at least one output file and select the type of file (for example, copycode) that is to be generated.

6 Optional. If you want to change the default values for the generation, choose the **Next** button.

On the next two pages, you can change the Natural defaults and the XML defaults. These values are taken from the Natural preferences. See *XML Toolkit* in *Setting the Preferences* for further information on these defaults.

7 To start the generation, choose the **Finish** button.

As a result of the generation, an editor window is opened for each generated output file. Each new object is shown in the **Navigator** view or in the **Natural Navigator** view, in the same library as the corresponding input file.

Displaying the Settings of the Last Generation

When you display the **Properties** dialog box for the file that was used as the input object for the generation, you can see the settings (for example, the replacements for the Natural variable names) that have been used for the last generation.

XV Use SSL/TLS

31 Use SSL/TLS

With NaturalONE it is possible to use Natural Development (NDV) servers (Version 9.1.2 or above) running with SSL/TLS. This is achieved by activating the SSL/TLS option in the corresponding mapping or project definition dialogs, for example, as described in section ***Mapping a Natural Environment***, in section ***Changing the Project Properties*** or ***Launching Natural Applications***.

NaturalONE supports SSL/TLS connections both with and without authentication. Please also refer to the **SSL/TLS preferences** for additional information.

XVI Deploying Applications

This part describes the different types of deployment that are available with NaturalONE. It covers the following topics:

Deploying Natural Applications
Deploying Natural Applications with Jenkins
Deploying Java Applications
Using a Master Deployment

Note: For information on how to deploy web applications that have been developed using Ajax Developer, see *Deploying the Application* in the *Natural for Ajax* documentation.

32 Deploying Natural Applications

- General Information 454
- Using the Deployment Wizard for Natural Applications 454
- Controlling the Scope of Files to be Processed 468
- Starting the Deployment from Eclipse 469
- Starting the Deployment from the Command Line 470
- Status Code Handling 471
- Checking the Time Stamps in the Natural Environment 472
- Reducing the Amount of Logging Information 475
- Project References Handling 475
- Versioning Repository Handling 476

General Information

You can deploy a Natural application from a version control system to a Natural server. The Natural sources and any other resources may reside in a versioning repository on one machine, and the ready-to-use application may be deployed to a different machine via a Natural server connection.

NaturalONE offers a deployment wizard which collects all required information (such as the access information for the versioning repository and the Natural server) and writes it to an Ant script which is used to start the deployment process. This Ant script makes the deployment task highly configurable and repeatable, and allows you to run the deployment process unattended.

The deployment process can either be started from within the NaturalONE Eclipse environment or via the Ant command line utility. It performs the following steps:

- check out a Natural application from a version control system (either CVS, Subversion or GIT); this is done outside of Eclipse,
- transfer the Natural objects to a Natural server,
- catalog the Natural objects on the Natural server.

Using the Deployment Wizard for Natural Applications

The deployment wizard creates a Natural deployment file in your project root. This is an Ant script. You can create one or more Natural deployment files for a project, and you can also load an existing Natural deployment file and modify the current settings.

The deployment file contains information on whether it has been created for a secured or unsecured environment, that is, whether the environment is protected by Natural Security or not. This setting cannot be changed after the deployment file has been created. Thus, if you change a project from secured to unsecured (or vice versa), you have to create a new deployment file.

To use the deployment wizard

1. In the **Navigator** view or in the **Natural Navigator** view, select the Natural project for which you want to create the deployment file.

   Or:

   If you want to load the settings of an existing Natural deployment file, select this file in the **Navigator** view or in the **Natural Navigator** view.
2. From the **File** menu or from the context menu, choose **New > Other**.

3 In the resulting **New** dialog box, expand the **Natural** node, select **Deploy Natural Ant** and then choose the **Next** button.

   The first page of the wizard appears (see below).

4 Specify all required information as described in the topics below. Use the **Next** button repeatedly to proceed from the first page of the wizard to the last page.

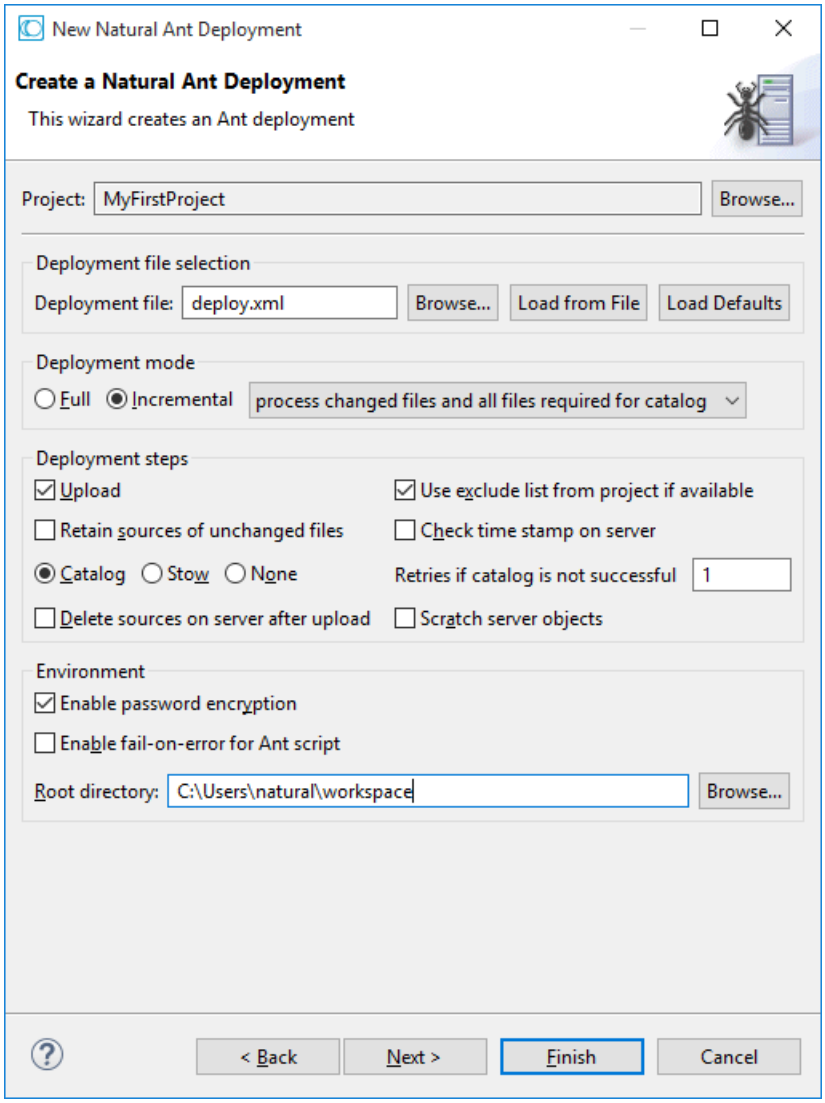
5 When all required information has been provided, choose the **Finish** button.

The different pages of the deployment wizard are described in the following topics:

- General Settings
- Source Types
- Repository
- Server Environment
- SSL/TLS Configuration
- Project References
- Steplibs
- Mappings
- Logging

General Settings

On the first page of the wizard, you define general settings for the deployment.



Deployment file

The default name for the Natural deployment file is *deploy.xml*. This name is shown in this text box when an existing Natural deployment file was not selected while invoking the wizard. However, when an existing Natural deployment file was selected, the name of the selected file is shown and the settings from this file are automatically loaded.

You can enter any other name for your new deployment file. It is recommended that your new deployment file also has the extension ".xml".

Note: If you keep the name *deploy.xml*, the settings from an existing Natural deployment file with the same name are loaded the next time you select the project and invoke the wizard.

If you want to load an existing Natural deployment file, choose the **Browse** button. A dialog appears, providing for selection all Natural deployment files in the current project. Next, you

have to choose the **Load from File** button. Otherwise, the settings in this file are not shown in the wizard and may thus be overwritten unintentionally.

If you want to return to the default settings of the deployment wizard (this also includes the information that can be specified on the other pages of the wizard), choose the **Load Defaults** button.

Deployment mode

Select one of the following option buttons:

- **Full**
  With a full deployment, the sources and resources in the project are processed completely each time the deployment process is started.
- **Incremental**
  With an incremental deployment, only those sources and resources are processed which have been changed in the versioning repository since the last run of the deployment. You determine the files that are affected by a subsequent upload (catalog) by selecting one of the following options from the drop-down list box:
  - **process only changed files**
    Only the files that have changed in the workspace are processed.
  - **process changed files and their dependents**
    All files that have changed in the workspace and all files which have dependencies to the changed files are processed. In special cases when PDAs or DDMs have been changed, it is necessary to process the dependencies of the dependencies also.
  - **process changed files and all files required for catalog**
    In addition to the files of the previous option, all files that are required to catalog the sources are also processed.

    Example: When a program is changed that uses a global data area (GDA), both the program and the GDA are uploaded to the server. In addition, all other sources which use the same GDA are also uploaded. All uploaded sources are then cataloged on the server.

See also *Controlling the Scope of Files to be Processed*.

Note: You can run more than one incremental deployment inside a single Natural project. Each of the incremental deployments maintains its own state of changed files. This is helpful, if you want to deploy your application into different environments, for example, into a testing environment and later into a production environment.

Upload

When enabled, the sources and resources are uploaded to the Natural server.

Use exclude list from project if available

When enabled, an exclude list is used. The files specified in the exclude list will not be used for the deployment. See also *Excluding Objects from Processing in the Natural Environment*.

Retain sources of unchanged files

When enabled, only the sources of changed files are updated on the Natural server. The sources of unchanged files are neither uploaded nor stowed on the server. In case **Stow** is also selected, the sources of unchanged files are cataloged instead. See also *Controlling the Scope of Files to be Processed*.

Note: This option only has an effect when an incremental deployment mode is selected. In full deployment mode, all files in the project are assumed to be changed files.

Check time stamp on server

When enabled, the time stamps of the sources to be uploaded are checked against the time stamps of the sources on the Natural server. If the sources on the server have been changed since the last deployment run, a time stamp conflict is detected and the corresponding sources on the server are not overwritten. For further information, see *Checking the Time Stamps in the Natural Environment*.

Catalog / Stow / None

When **Catalog** is selected, the uploaded sources are cataloged on the Natural server. When **Stow** is selected, the uploaded sources are saved and cataloged on the Natural server (the time stamps of the sources and cataloged objects are then identical). When **None** is selected, the uploaded sources are neither cataloged nor stowed on the Natural server.

Retries if catalog is not successful

When **Catalog** or **Stow** is selected, you can specify the number of retries for cataloging. This is helpful since it will not always be possible to catalog all sources in the proper order in one pass. When you specify a number greater than 1, the erroneous sources are recataloged until either no more errors occur or the specified number of retries has been reached. Default: 1.

Delete sources on server after upload

When enabled, the sources are deleted on the Natural server after they have been uploaded and (if **Catalog** or **Stow** is selected) cataloged.

Scratch server objects

When enabled, Natural objects on the server are scratched in case the corresponding objects in the workspace have been deleted.

Enable password encryption

When enabled, all passwords that are used in the deployment file are stored in an encrypted format.

Enable fail-on-error for Ant script

When enabled, the Ant script reports errors and terminates in the case of a build failure.

When disabled, the Ant script still reports errors but build failures are only triggered in severe situations (see also *Status Code Handling*).

Root directory

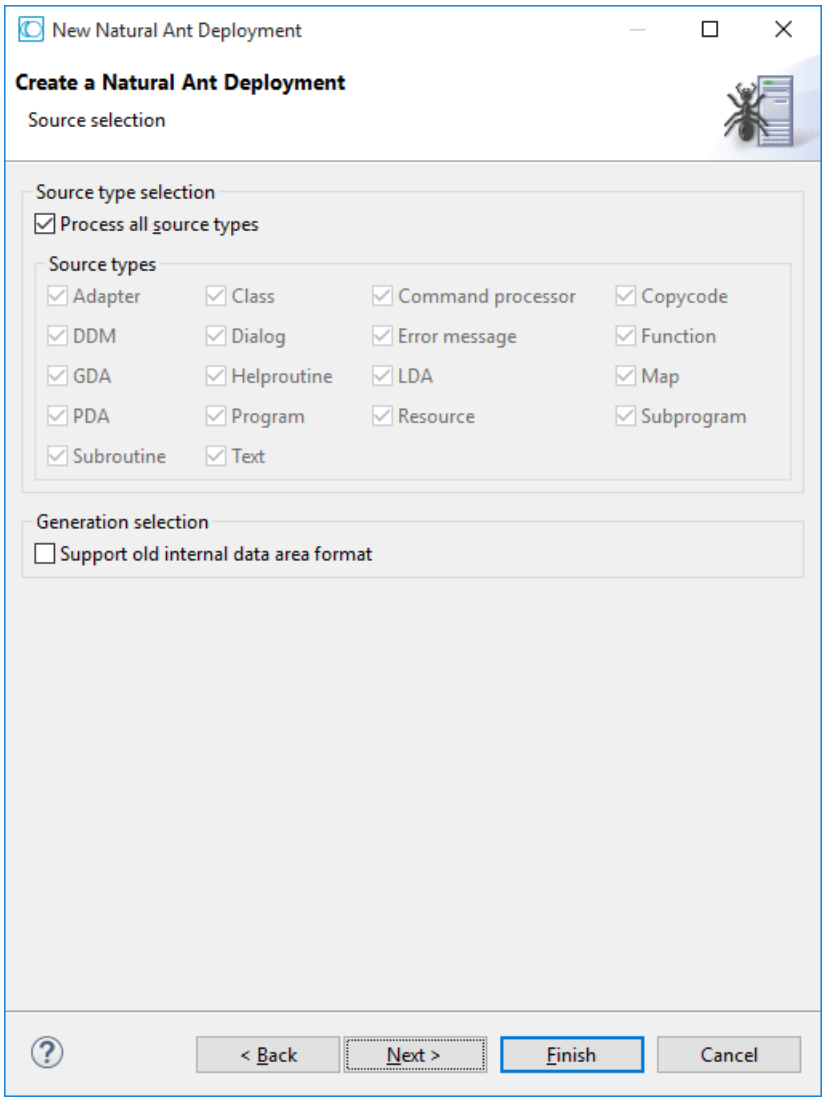
This path should only be changed when you intend to start the deployment from the command line (that is, when the deployment is not to be started from Eclipse).

Specify the directory in which the selected project is to be checked out and where the processing takes place. When the deployment is supposed to run on a different machine, you can insert the desired root path via copy-and-paste.

Source Types

On the this page of the wizard, you can specify the source types that are to be processed.

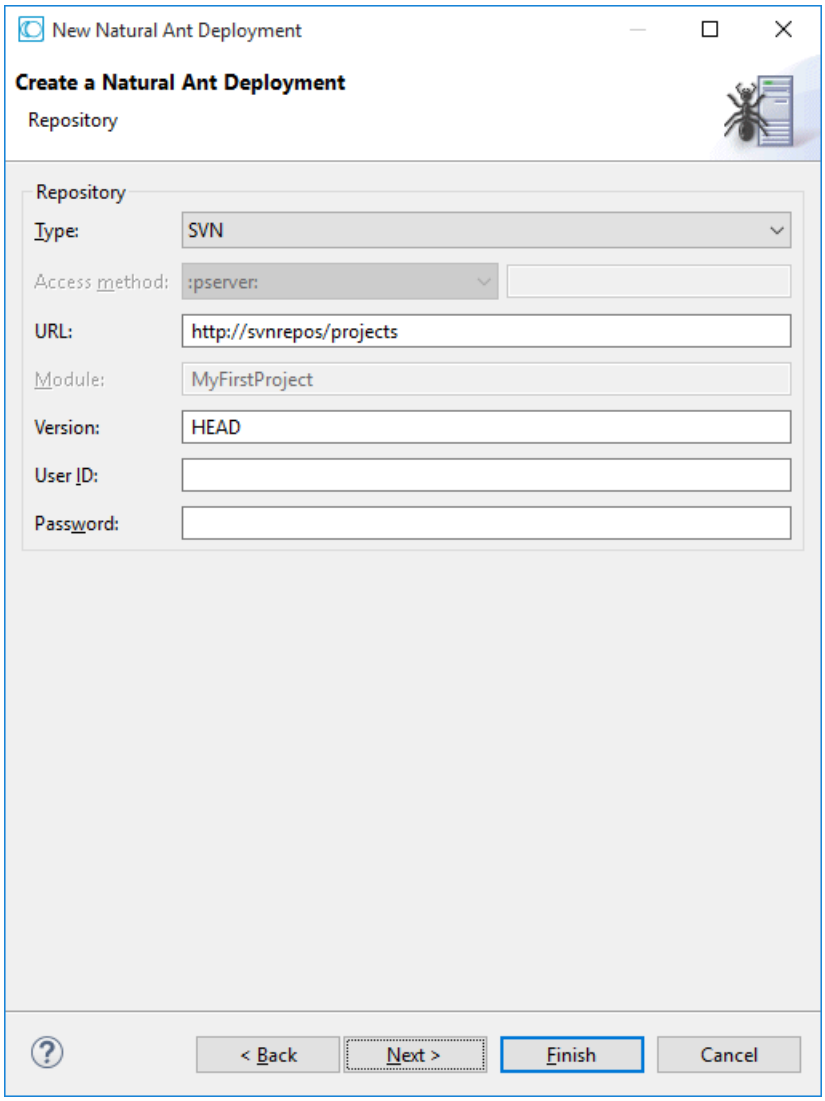
When the **Process all source types** check box is not selected, you can deselect any sources types which are not to be processed. For example, if you deselect the **GDA** check box, global data areas are not processed (that is, they are not uploaded and cataloged) during the deployment.



Select the **Support old internal data area format** check box only, if you require data areas in the old format that are to be used with Natural Version 5.1 or below. See also ***Natural*** in *Changing the Project Properties*.

Repository

On this page of the wizard, you define all settings related to the versioning repository. This can be either Subversion (SVN), GIT or CVS.

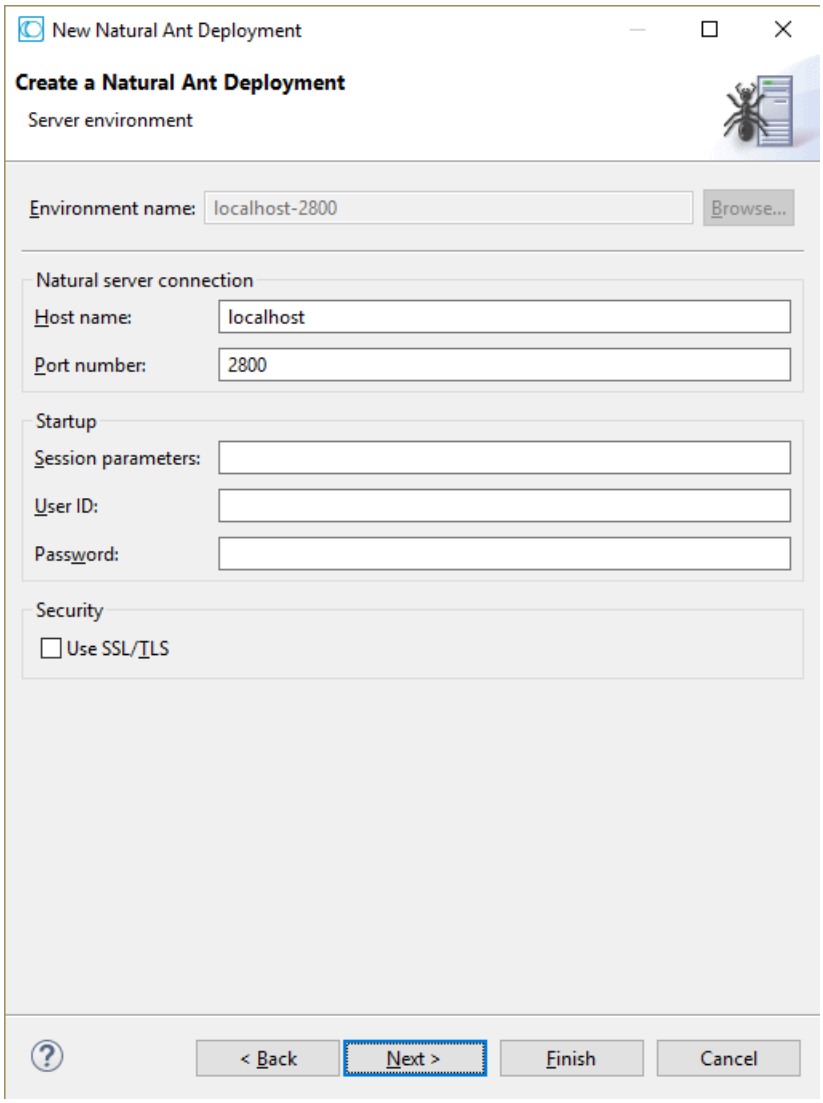


From the **Type** drop-down list box, select the type of versioning repository that you are using, and then specify all required information. The names of the text boxes and their availability changes according to the selected type.

The wizard usually collects a set of default information as given for the selected project. In most cases, only minor corrections have to be made to the defaults, for example, user ID and password may have to be provided.

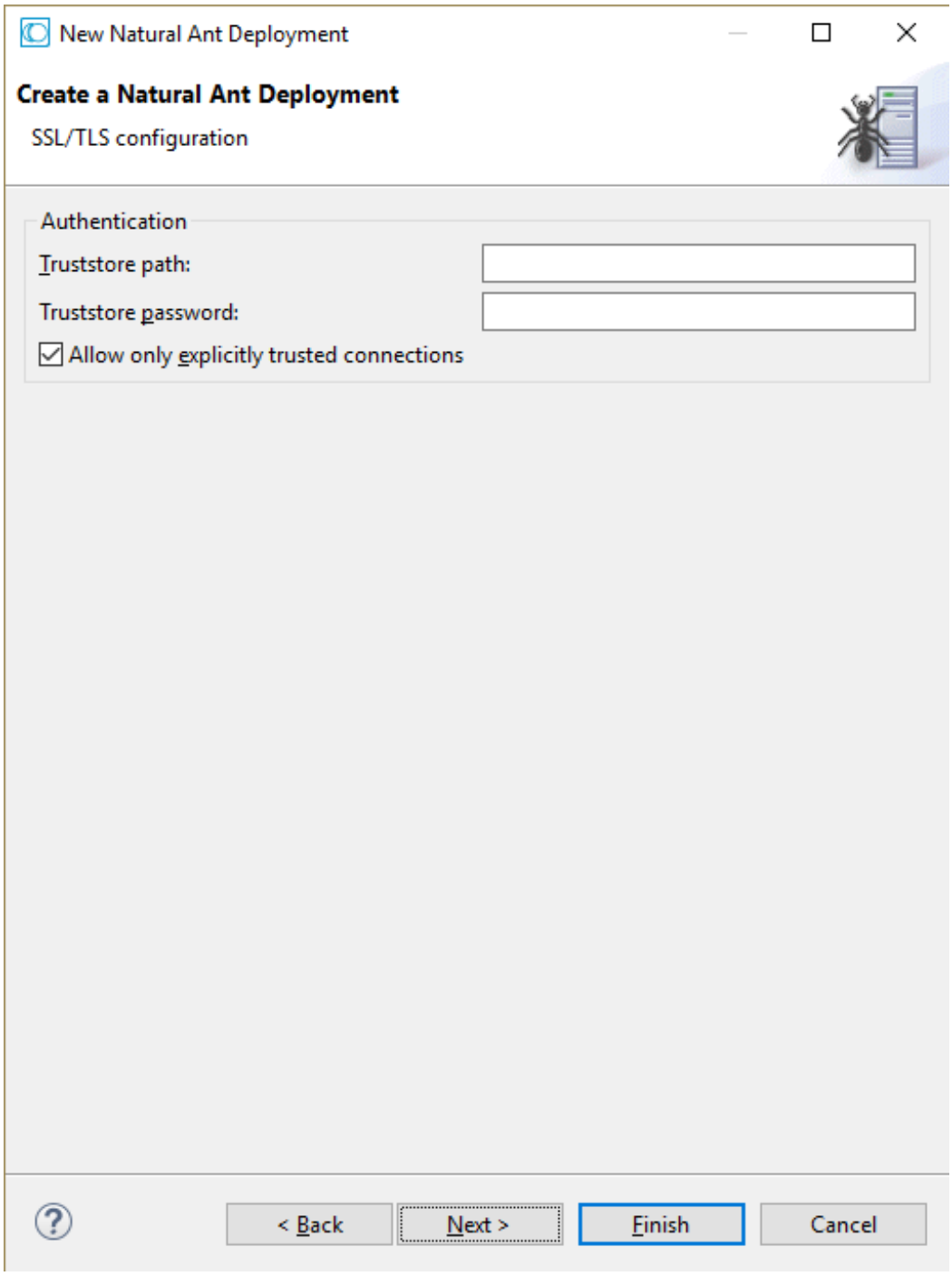
Server Environment

The next page of the wizard shows information which applies to the Natural server to which the selected project belongs (such as host name and port number). You can change the settings according to your requirements. For information on the options on this page, see *Mapping a Natural Environment*.



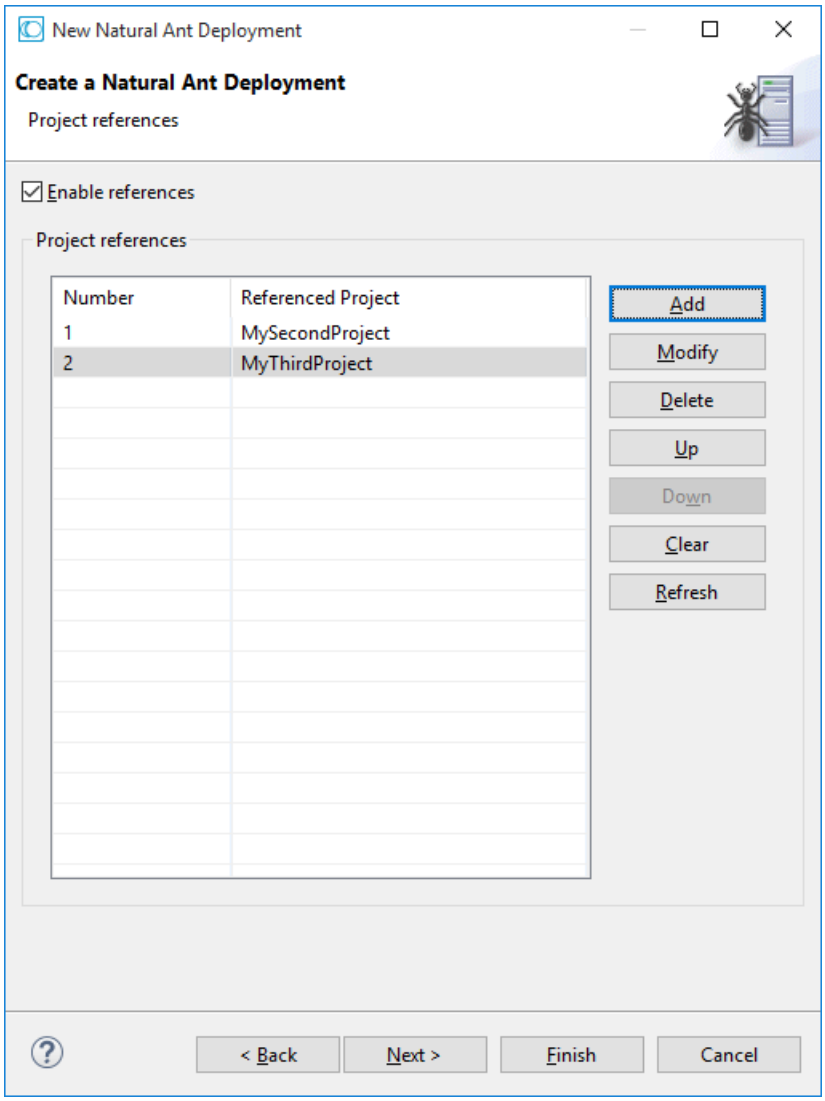
SSL/TLS Configuration

This wizard page is used for setting information related to the usage of SSL/TLS and as such is only shown if the **Use SSL/TLS** option on the previous page is selected. For information on the options on this page, see *SSL/TLS* under *Setting the Preferences*.



Project References

On this page of the wizard, you define the project references for the current project.



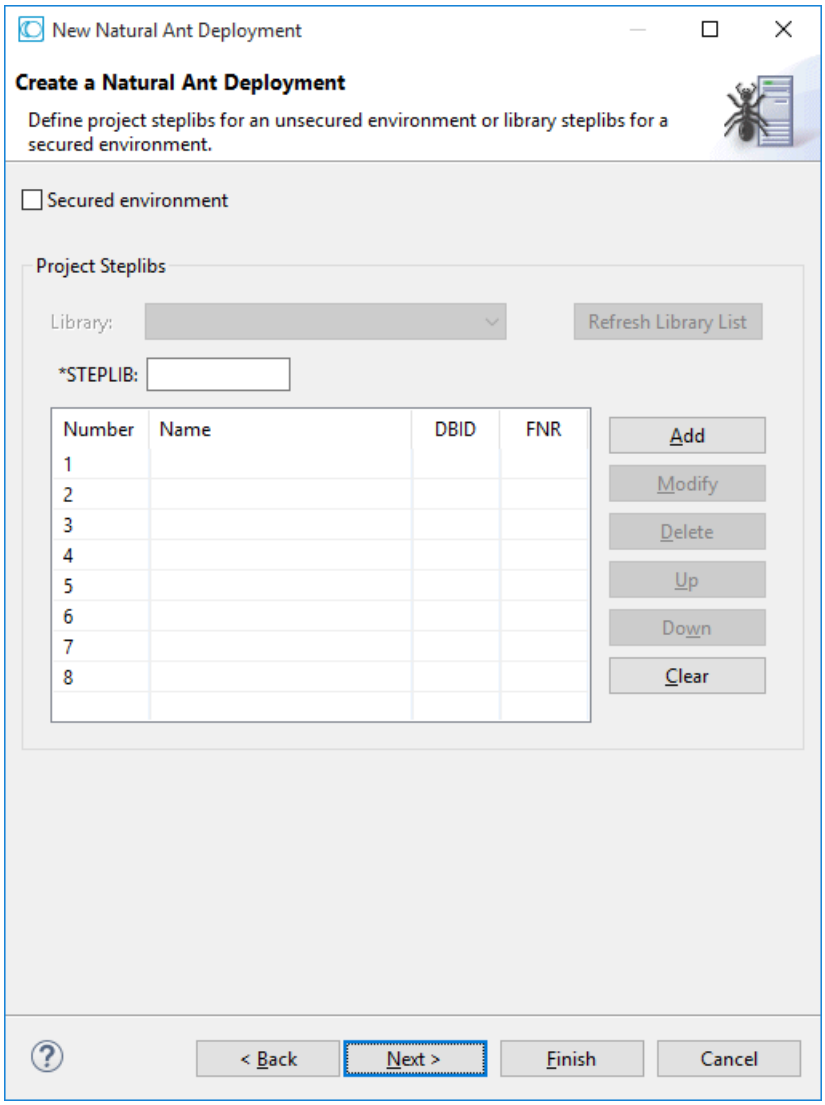
The wizard collects an initial set of project references from the current project's property settings. When enabling references, you can modify the list accordingly. The content of subsequent wizard pages (**Steplibs** and **Mappings**) will be affected corresponding to the project references being enabled.

Refer to *Project References Handling* for a more detailed description of this feature.

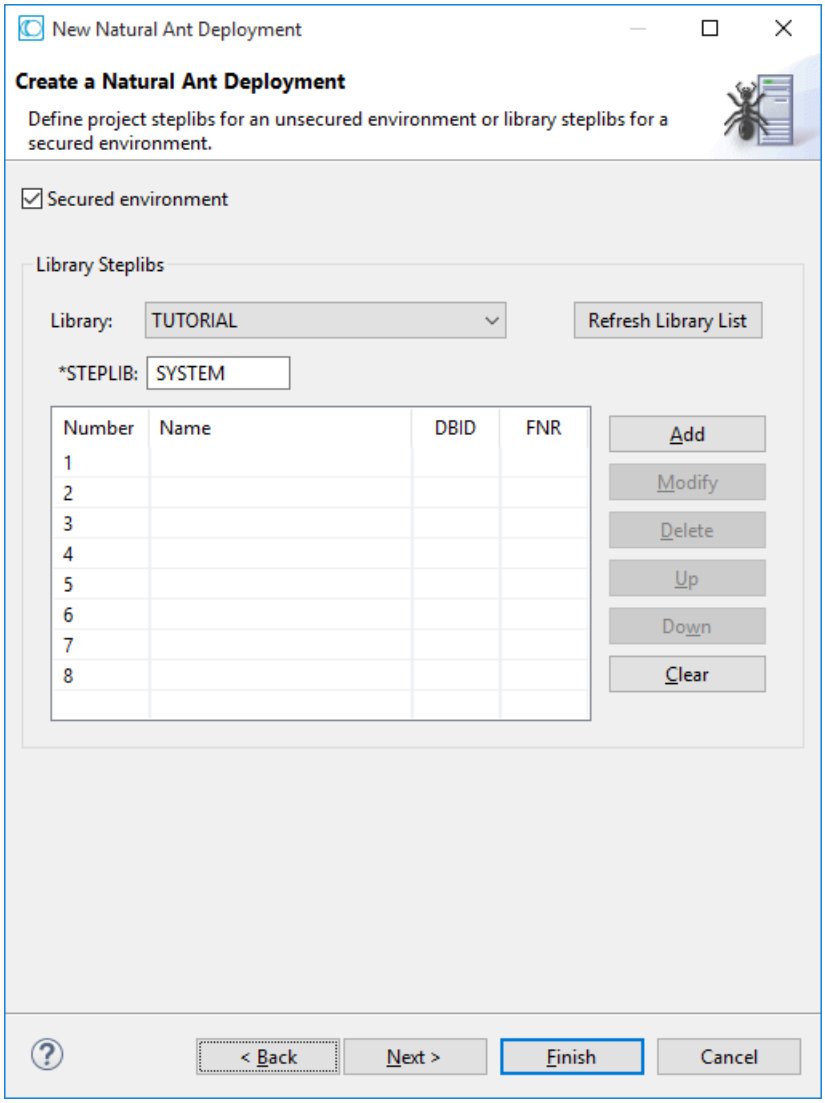
Steplibs

The fifth page of the wizard shows the steplib settings for the current project, depending on whether the current environment is protected by Natural Security or not.

The following page is shown for an unsecured environment. In this case, project steplibs can be defined.



The following page is shown for a secured environment. In this case, library steplibs can be defined.



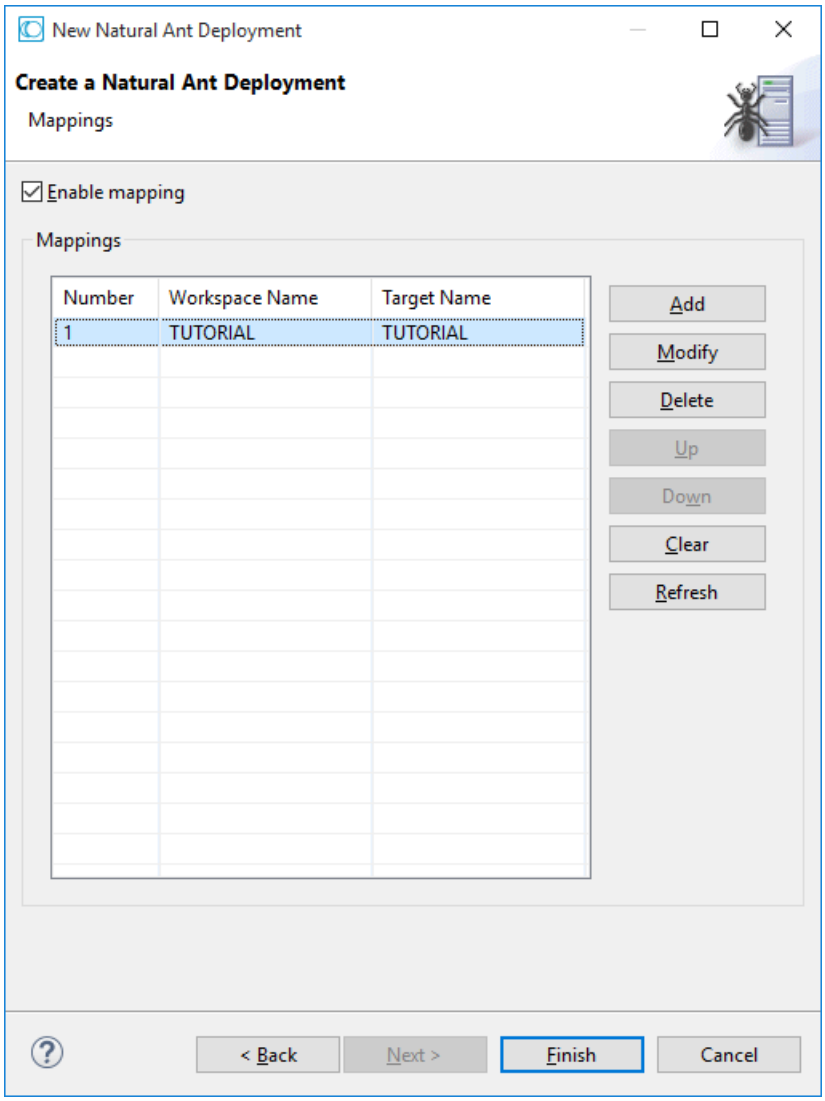
In many cases, the given settings are sufficient. For information on the steplib options on this page, see the description of the property page **Steplibs** in *Changing the Project Properties*.

When project references have been enabled, you can refresh the list of libraries and control the steplib settings also for libraries residing in a referenced project.

Note: The steplib SYSTEM without any explicit DBID/FNR assignment always refers to the FUSER of the actual connected server. If you want SYSTEM to refer to the FNAT of the actual server then the FNR can explicitly be set to "-2".

Mappings

On this page of the wizard, you can enable the mapping of library names.



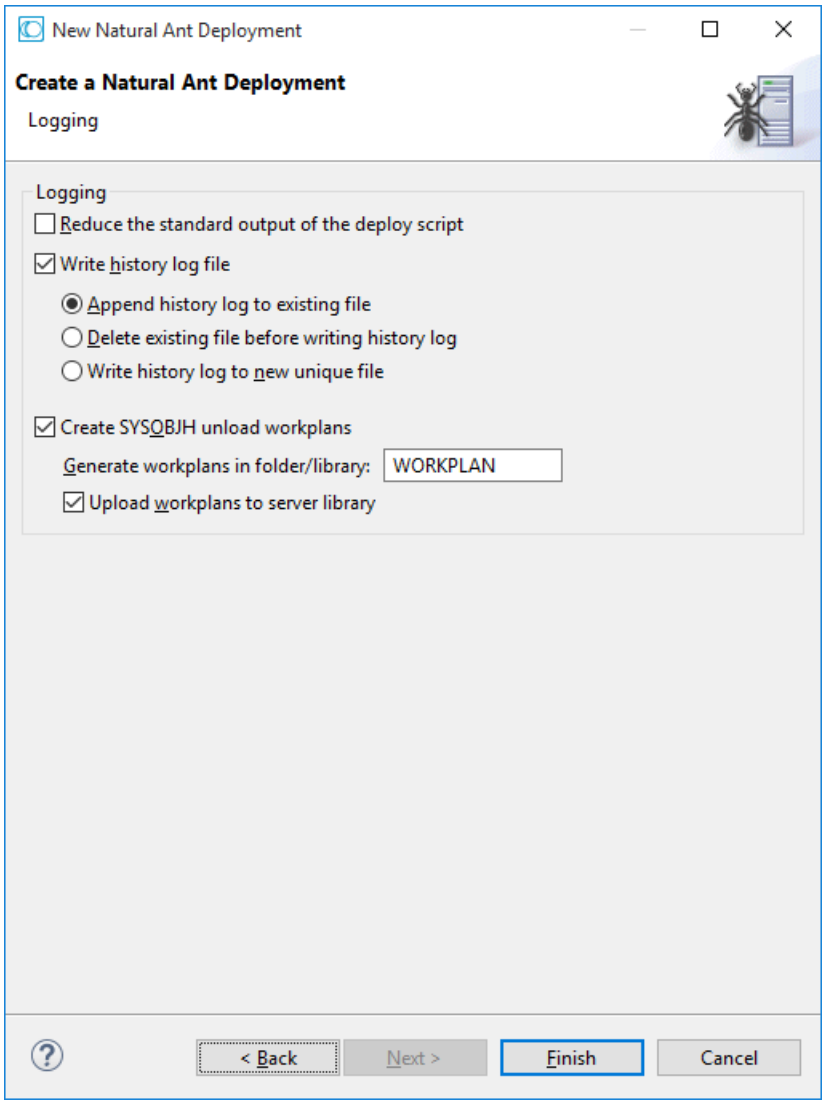
When the **Enable mapping** check box is selected, you can change the mapping information for all required libraries. The workspace name is the library name that is used in the versioning repository and in the current Eclipse workspace. The target name can be a different library name that is to be used on the Natural server.

To add a new mapping, choose the **Add** button and specify both the workspace name and the target name.

When project references have been enabled, you can refresh the list of libraries by pressing the **Refresh** button. Mappings for libraries from referenced projects can be controlled here.

Logging

On the last page of the wizard, you can control the logging capabilities of the deployment.



When **Reduce the standard output of the deploy script** is selected, the report of uploaded and cataloged files will be restricted to only those events that produced errors. So especially for deployments consisting of a huge amount of objects, the size of the script output reduces significantly.

When **Write history log file** is selected, the deployment generates an additional file with a similar format as described in section *Logging* under *Managing Natural Projects*.

For each object that has been processed (uploaded/cataloged/stowed/scratched) an output line is generated describing the action that has been performed. With the particular buttons you can control whether

- each deployment run appends its history log at the end of the only file,

- each deployment run deletes the only file before starting the log or
- each deployment run writes it history log into a new uniquely created file.

When **Create SYSOBJH unload workplans** is selected, the deployment generates Natural text objects that represent SYSOBJH workplans of type PROCEDURE and LIST. These workplans comprise the actions that have been performed during the Ant deployment run and can be used to unload exactly those objects in the server environment that were updated during the deployment run.

Hence, the unloaded objects can be transferred to another server environment without having to execute another Ant deployment run. Furthermore, in case objects have been scratched during the Ant deployment run the workplans also contain unload delete instructions that are executed in the server target environment if the "delete allowed" option has been configured properly in SYSOBJH.

Depending on the amount of objects processed, one or more workplans of type PROCEDURE (named PROC*nnn*.NST) and type LIST (named LIST*nnn*.NST) are generated. The workplans only have basic content and may be enhanced by the user with server specific information like workfile locations before they are executed with SYSOBJH. For performing the full unload via SYSOBJH all workplans of type PROCEDURE need to be executed consecutively.

In the wizard, you can use the **Generate workplans in folder/library** field to specify the name of the folder/library where the workplans are generated. In the Ant workspace, the folder name is prefixed with a dot. The workplans can automatically be uploaded to the corresponding server library by selecting **Upload workplans to server library**.

For more information about workplans and their execution, refer to the *Object Handler* description in the *Natural Tools and Utilities* documentation.

Controlling the Scope of Files to be Processed

When you deploy a Natural application, you can control the scope of the files that are to be processed in the Ant workspace. This is done by choosing one of the following deployment modes on the **first** page of the deployment wizard for Natural applications:

- **Full Deployment Mode**
  When the full deployment mode is selected, all files in the project are processed. The files are uploaded to the Natural server and are then cataloged on the server. The full deployment mode treats each file in the project as if it has been changed since the last run of the deployment script.
- **Incremental Deployment Mode**
  When the incremental deployment mode is selected, the scope of the files to be processed can be influenced using the options in a drop-down list box. Based on a set of changed files in the Ant workspace (which is provided, for example, by a versioning system), a superset of files is

calculated for further processing. Using the above-mentioned drop-down list box, you can select to

- process only the changed files,
- process the changed files plus their dependencies, or
- process the changed files, their dependencies, plus all files required for cataloging them.

See also *General Settings* for a description of each option.

Depending on your selection, the superset of files to be processed may be larger than the set of files that have actually been changed in the Ant workspace. If you want to process such a larger amount of files in the Ant workspace but do not want to upload and stow all the unchanged files on the server, you can select the **Retain sources of unchanged files** option on the **first** page of the deployment wizard. When this option is selected, the Ant deployment just uploads the changed files and assumes that all other necessary files are already present on the server. The already present files are only cataloged; their sources on the server remain unchanged during the deployment.

Starting the Deployment from Eclipse

When you start the deployment process from Eclipse, it is not possible to execute the `checkout` and `update` targets of the deployment file since these targets would access the versioning repository, and this is not feasible from within an Eclipse environment. If you want to check out a specific revision from the versioning repository or if you want to update your project with sources from the versioning repository, you have to start the deployment from the command line as described below.

For testing purposes, for example, it is helpful to start the deployment process from Eclipse. Since the Natural deployment file is an Ant script, the built-in Eclipse functionality of starting Ant scripts is used here. The `build` target of the Ant script will then be executed.

To start the deployment from Eclipse

- In the **Navigator** view or in the **Natural Navigator** view, select your Natural deployment file, invoke the context menu and choose **Run As > Ant Build**.

  The deployment process is started, and the output of the deployment file is written to the **Console** view.

  Note: If you want to change the limit for the console output, you can do this in the general Eclipse preferences under **Run/Debug > Console**.

Starting the Deployment from the Command Line

You can start the deployment process from a Windows command line such as the Command Prompt (*cmd.exe*) or from a shell command line on a Linux system. When you start the deployment from the command line, special requirements must be met.

The following topics are covered below:

- Prerequisites
- Starting the Deployment

Prerequisites

The following must be installed and accessible:

- Apache Ant 1.7.1
- Java Development Kit (JDK) 1.7.0_11 or above.
- Either the Subversion command line tool 1.5.2 or above, the CVS command line tool 1.11 or above, or the GIT command line tool 1.9.5 or above.

You have to copy the following JAR files, which contain the necessary processing code, from the NaturalONE Eclipse installation to the new directory (when using a **master deployment file**, it is not required to manually copy the files mentioned below):

- *com.softwareag.naturalone.natural.ant_*`<version>`*.jar*
- *com.softwareag.naturalone.natural.auxiliary_*`<version>`*.jar*
- *com.softwareag.naturalone.natural.ndvserveraccess_*`<version>`*.jar*
- *com.softwareag.natural.tools_*`<version>`*.jar*
- *com.ibm.icu.charset_*`<version>`*.jar*
- *com.ibm.naturalone.icu_*`<version>`*.jar*

You have to copy your deployment file, which has been created by the deployment wizard, into the directory which has been specified in the wizard as the root directory (this is the base directory for processing).

Starting the Deployment

When all prerequisites are in place, the deployment can be started by issuing specific Ant calls. This section just provides some examples (where the default name *deploy.xml* is used).

- Print the help screen of the Ant script:

```
ant -lib path-to-mylib -f deploy.xml help
```

- Perform an initial checkout of the project sources from the versioning repository:

```
ant -lib path-to-mylib -f deploy.xml checkout
```

- Perform an update of the project sources from the versioning repository:

```
ant -lib path-to-mylib -f deploy.xml update
```

- Deploy the sources to the Natural server and perform the deployment actions as specified in the deployment wizard (such as upload, cataloging, mappings):

```
ant -lib path-to-mylib -f deploy.xml build
```

- With a single call, perform an update of the project sources from the versioning repository first, and then deploy the sources:

```
ant -lib path-to-mylib -f deploy.xml update build
```

- In the above examples, the logging information is written to standard output. If logging information is to be written to a file, use a call such as the following:

```
ant -lib path-to-mylib -f deploy.xml update build -logfile mylogfile.txt
```

Status Code Handling

The Ant deployment script can run in two status code modes. The mode can be toggled by specifying the command line parameter `-Dnatural.ant.failonerror` as described in the following table.

Command Line Option	Description
`-Dnatural.ant.failonerror=no`	Only severe errors such as missing project directories will lead to a build failure with a status code other than 0. Other errors such as catalog or stow errors will be reported but will not trigger a status code other than 0 and hence will not lead to a build failure. This is the default mode.
`-Dnatural.ant.failonerror=yes`	In addition to the severe errors described above, errors occurring during checkout, update, catalog, stow and delete will also lead to a status code other than 0 and hence will lead to a build failure.

Note: The default mode can also be changed on the **first page** of the deployment wizard.

When the additional status code handling has been enabled, the Ant tool as well as the internally used tools such as SVN, GIT or CVS clients may issue specific status codes. In case the status codes are unclear, refer to the documentation of these tools.

In addition, the following NaturalONE-specific status codes may be issued:

Status Code	Description
0	No build errors occurred.
11	An error occurred while reading or writing the Natural Development Server configuration data. Check whether the Natural Development Server is accessible.
12	It is not possible to connect to the Natural Development Server. Check whether the Natural Development Server is running and accessible.
13	An error occurred while uploading Natural objects. Check whether the Natural Development Server allows uploading and saving of Natural objects in the affected libraries.
14	An error occurred while cataloging Natural objects. Check the affected objects and correct the errors.
15	An error occurred while deleting Natural objects. Check whether the affected objects are available on the server

Note: Under Windows, Ant currently maps all error codes other than 0 to 1. Thus, `%ERRORLEVEL%` is either 1 for errors or 0 for no errors.

Checking the Time Stamps in the Natural Environment

When deploying Natural applications in the usual way, the sources on the Natural server are exclusively updated by the Ant deployment scripts. In this case, the sources in the Ant workspace and the sources on the Natural server are always identical. If the sources in the Ant workspace are updated with newer revisions from a version control system, the Ant script can easily detect the new sources and directly transfer them to the Natural server without conflicts.

In special cases where the sources on the Natural server are not exclusively controlled by the Ant deployment scripts but, for example, are directly edited on the server, the deployment process would simply overwrite the modified sources when they have also been changed in the versioning repository.

To avoid overwriting of sources that have been modified directly on the server, it is possible to enable time stamp checking in the deployment wizard for Natural applications (see ***General Settings*** under *Using the Deployment Wizard for Natural Applications*).

The following topics are covered below:

- Prerequisites
- General Information on Time Stamp Checking
- Maintaining Time Stamps in the Ant Workspace
- Detecting Time Stamp Conflicts with checkts
- Resolving Time Stamp Conflicts

Prerequisites

Time stamp checking works as of the following versions:

- **Mainframe**
  Natural Development Server Version 2.2.7 cumulative fix 9 or above.
- **UNIX or Windows**
  Natural Development Server Version 2.2.6 or above.

  Natural Version 6.3.10 or above.

General Information on Time Stamp Checking

When time stamp checking has been enabled in the deployment wizard for Natural applications, the following happens during the deployment process:

- For each source in the Ant workspace, the time stamp of the last successful upload to the Natural server is maintained. This time stamp is collected from the Natural server when the last upload has succeeded and has been stored in the Ant workspace.
- When a source is to be uploaded to the Natural server, the collected time stamp of the last successful upload is compared with the current time stamp on the Natural server.
- When the time stamps are identical, the source is uploaded. The new time stamp is collected and stored in the Ant workspace. The upload is successful.
- When the time stamps differ, the content of the source in the Ant workspace is compared with the content of the source on the Natural server:
  - If the sources are identical, it may be possible that the time stamp difference occurred, for example, due to a `CATALL * STOW` system command on the server which changes the time

stamp of the source but not the content. In this case, the source from the Ant workspace is uploaded. The new time stamp is collected and stored in the Ant workspace. The upload is successful.

- If the sources are not identical, a time stamp conflict has occurred and the source in the Ant workspace is not uploaded. A specific error message is shown. The time stamp is not collected and is not updated in the Ant workspace.

Maintaining Time Stamps in the Ant Workspace

The time stamps in the Ant workspace are stored in the file *timestamp_<project-name>.properties*. When time stamp checking is enabled in the deployment wizard, this file is created during the deployment process. The time stamps in this file are updated each time an upload is successful.

If time stamp checking is disabled, an existing time stamp file in the Ant workspace will be deleted during the next deployment run. This is necessary because the time stamps in this file must be most accurate, and this is not the case when an in-between deployment is run without time stamp checking.

Detecting Time Stamp Conflicts with checkts

The `checkts` target of the Ant deployment script simulates the next run of the script without making any changes. It just reports the time stamp conflicts so that you are able to resolve these conflicts before starting the usual deployment process.

The following is an example of starting the Ant script with the `checkts` target:

```
ant -lib path-to-mylib -f deploy.xml checkts
```

Resolving Time Stamp Conflicts

Generally, resolving time stamp conflicts is a task left to the user of the deployment process because the reason for a time stamp conflict may be very specific and cannot be solved with a general rule or processing method.

When the time stamp conflicts have been resolved, you can synchronize the time stamps of the sources that were in conflict. To do so, you start the Ant script with the `checkts` target and the additional option `-Dnatural.ant.resolve.time.stamp.conflicts=yes`. For example:

```
ant -lib path-to-mylib -f deploy.xml checkts ↩
-Dnatural.ant.resolve.time.stamp.conflicts=yes
```

If the Ant script is started like this, it updates the time stamps in the Ant workspace that were in conflict with the current time stamps from the server. When the next usual deployment run is then started using the `build` target, the sources that previously caused the time stamp conflicts are also uploaded.

Keep in mind that the above Ant call only updates the time stamps in the Ant workspace for the sources that previously had time stamp conflicts. It does not perform the upload itself and it does not resolve the reason for the conflict, for example, by merging sources.

Reducing the Amount of Logging Information

When using the Natural Deployment the log generated by the Ant script might be to comprehensive, especially when the output is being analyzed with tools like Jenkins. Therefore, the Ant script can be started with an option `-Dnatural.ant.short.log=yes` which reduces the log output to a minimum. For example:

```
ant -lib path-to-mylib -f deploy.xml -Dnatural.ant.short.log=yes
```

Project References Handling

Natural applications can be spread across several Natural projects. To achieve this project references can be added in the Natural project properties. Such project references are controlled and administered by Eclipse internally.

The NaturalONE deployment offers a way to handle project references outside of Eclipse. The Natural project for which the deployment script will be generated is the so-called base project. To enable and use project references in the Ant deployment script the following shall be done:

1. Enable project references on the **Project References** wizard page when generating the Ant deployment script for the base project.
2. An initial set of referenced projects has already been loaded from the base projects properties and is displayed in the references table on the wizard page.
3. Project references can be added or deleted with the appropriate controls on the table's right-hand side.
4. Enabling project references might affect the list of libraries in subsequent wizard pages like **Steplibs** or **Mappings** because a referenced project might introduce additional libraries for the application. The Steplib library refresh button and the Mapping table refresh button enable you to refresh the list of project libraries.

5. An enhanced deployment Ant script is generated with specific code for referenced projects when the **Finish** button is pressed.

When an Ant deployment script, where project references have been enabled, is started it handles the objects in the referenced projects as if they were in the Natural base project. Therefore, it is important that the base project and all referenced projects are below the same root directory, i.e. all the projects must be side-by-side in one directory. During deployment all affected objects are uploaded/cataloged, regardless if they are from the base project or from a referenced project. So the deployment script does not process single projects anymore but complete applications across project boundaries.

The internal configuration files for tracking changed objects, timestamps etc. for the base project and for the referenced projects are all stored inside the base project. The referenced projects remain untouched.

The following restrictions apply to the deployment of base projects and their referenced projects:

- The deployment script does not handle references from references. That means all project references must be defined in the base project in the appropriate wizard page. Hence, some reference from a referenced project must be added manually here.
- All steplib specifications, also those for the referenced projects, must be done in the appropriate steplib wizard page. The wizard page allows you to also describe steplibs for libraries in referenced projects.
- The deployment script for the base project cannot handle checkout/update actions of the underlying versioning system for the referenced projects. If it is necessary to issue such actions by deployment scripts, an option would be to generate specific scripts for each referenced project and run them with the checkout/update command beforehand.

Versioning Repository Handling

In most cases when using the Ant deployment a versioning repository is used to maintain the Natural objects that are to be deployed. For accessing projects in versioning repositories the specific Ant script targets checkout and update, which can be used as well as other external tools of your choice. Here are some remarks that need to be considered in that context.

When running the Ant deployment script from Eclipse the complete versioning repository functionality of the Ant script is disabled. This is necessary because otherwise it would interfere with the versioning plugins of Eclipse. If you run the Ant deployment script from Eclipse, you need to use the Eclipse versioning tool functionality to update the Natural objects before you run the Ant script.

The Ant deployment script currently supports CVS, Subversion and GIT as versioning repositories. There is a distinct difference in the handling of repositories of type CVS and Subversion on the one hand and GIT on the other hand when running the Ant deployment script outside of Eclipse.

With CVS and Subversion the projects in the root directory that is defined in the Ant script have the following layout:

<root directory>/MyProject1/<contents of the project>
<root directory>/MyProject2/<contents of the project>

The root directory contains the project folder which then contains the project contents like the .natural and the .project files.

With GIT as the versioning repository the layout is different:

<root directory>/MyProject1/MyProject1/<contents of the project>
<root directory>/MyProject1/.git
<root directory>/MyProject2/MyProject2/<contents of the project>
<root directory>/MyProject2/.git

Here an additional directory with the same name as the project directory it contains is created. The additional enclosing directory is necessary because a cloned GIT repository always comes with a .git directory at the same level as the project directory. An enclosing directory avoids conflicts of two or more different projects, each coming with a .git directory of its own.

Please also note that when using other tools to checkout the projects from the versioning repository the Ant script still expects the abovementioned directory layout to exist, according to the repository type set during the script generation.

33 Deploying Natural Applications with Jenkins

- General Information 480
- Basic Assumptions 480
- Generating and Verifying Deployment Scripts with NaturalONE 480
- JAR Files for Deployment 481
- Versioning Repository Handling 481
- Files Generated by Natural Deployment Scripts 481
- Jenkins Jobs 482

This chapter covers the following topics:

General Information

This chapter deals with some general guidelines and recommendations for using the Natural deployment scripts with a Jenkins server. The chapter is not an introduction to Jenkins server in general, nor can it be a comprehensive tutorial on integrating the Natural deployment scripts with a Jenkins server environment. It is rather a collection of ideas, hints, best practices as well as issues when setting up a new deployment environment on Jenkins. Users of other automation servers may also find some valuable hints here.

For a detailed description of the Natural deployment scripts, see **Deploying Natural Applications**.

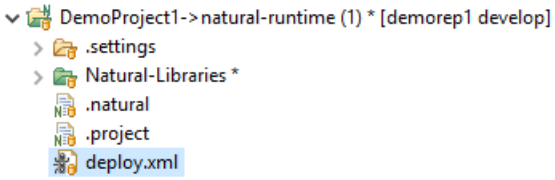
Basic Assumptions

The most basic assumption is that a Natural application has been developed with the help of NaturalONE and has been stored in a source code versioning system, such as GIT. The main goal of the deployment set up via Jenkins is to automatically extract the Natural sources from GIT, detect changes, and transfer the changed sources to a Natural development server where they can be cataloged.

Generating and Verifying Deployment Scripts with NaturalONE

When a new deployment script has been generated with NaturalONE, it is always a good idea to test the script in order to avoid the most obvious errors. You can test directly from within NaturalONE, as described in **Starting the Deployment from Eclipse**.

If the script works as expected, it can be added to the GIT repository, so that it is automatically checked out with the rest of the project when used with the Jenkins server. An example of a NaturalONE project with a deployment script named *deploy.xml* might look like this:



JAR Files for Deployment

The jar files described in **Starting the Deployment from the Command Line** must be available on each Jenkins node that is used for running the deployment scripts. The directory with the jar files must be accessible to the user (agent) running the Jenkins job. The jar files represent the runtime environment for the Natural deployment scripts, so it is recommended that both are from the same NaturalONE version. When upgrading to a new general release version of the product, the deployment scripts shall be regenerated and the jar files shall be updated as well.

Versioning Repository Handling

The Natural deployment scripts are designed to run in environments where no NaturalONE and no Jenkins or similar system is available. For example, the Natural deployment scripts could be run via the Windows Task Scheduler or the Linux cron daemon utility. Therefore, the Natural deployment scripts contain more functionality than necessary when running with Jenkins.

To access the sources in the repository with Jenkins, it is recommended that you use the versioning tool plugins from Jenkins itself instead of the built-in functionality of the Natural deployment script (i.e. checkout and update). This way it is possible to bootstrap the deployment scripts from the repository as well, changelog files are created automatically and continuous build features like polling can be used.

Files Generated by Natural Deployment Scripts

As a result of running a Natural deployment script, some files are generated in the project directory:

- *cache_*.properties* contains the hash codes of each file in the project and is used to identify modified files.
- *timestamp_*.properties* is generated when timestamp checking has been switched on. For further information on time stamps, see **Checking the Time Stamps in the Natural Environment**.
- *history_*.txt* is generated when history logging has been switched on. For more information on logging capabilities, see **Logging**.

These files represent the current status of the Natural deployment task. *cache_*.properties* must not be deleted, otherwise the deployment process will start over with an initial full deployment.

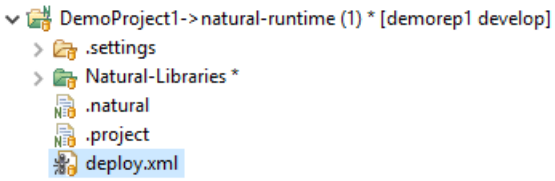
Jenkins Jobs

Jenkins allows for various job types to be defined, two of which shall be observed a bit closer, namely freestyle jobs and pipeline jobs. With freestyle jobs you can bring together all your already existing build jobs, test jobs, tools, etc. and administer them on the Jenkins web UI. Pipeline jobs, on the other hand, allow the user to define the workflow of the required build activities and to split them into separate stages. Unlike freestyle jobs, which are defined with the Jenkins Web UI, pipeline jobs are coded scripts.

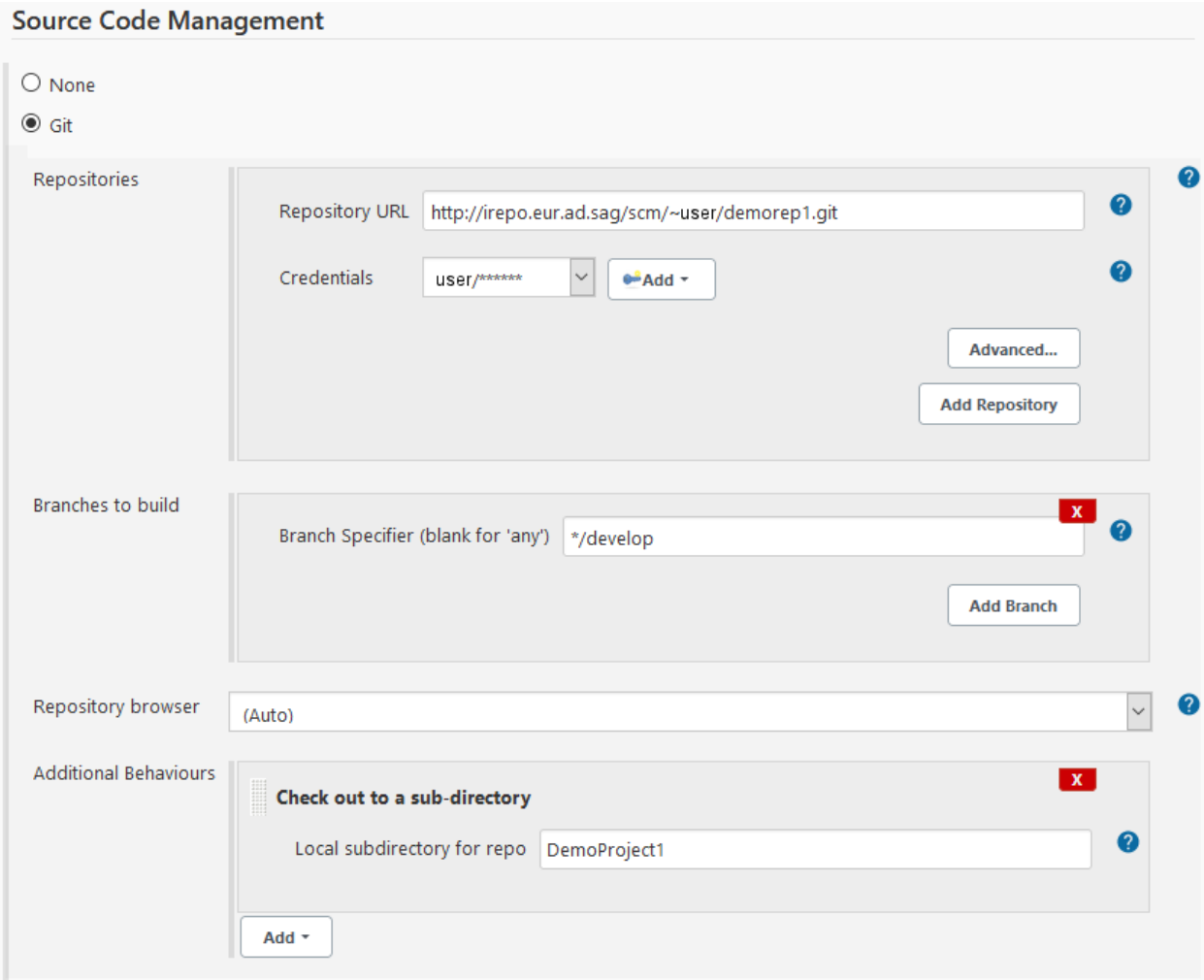
- Jenkins Freestyle Jobs
- Jenkins Pipeline Jobs
- Using Jenkinsfile

Jenkins Freestyle Jobs

The project we want to build looks like this and is stored in a GIT repository:

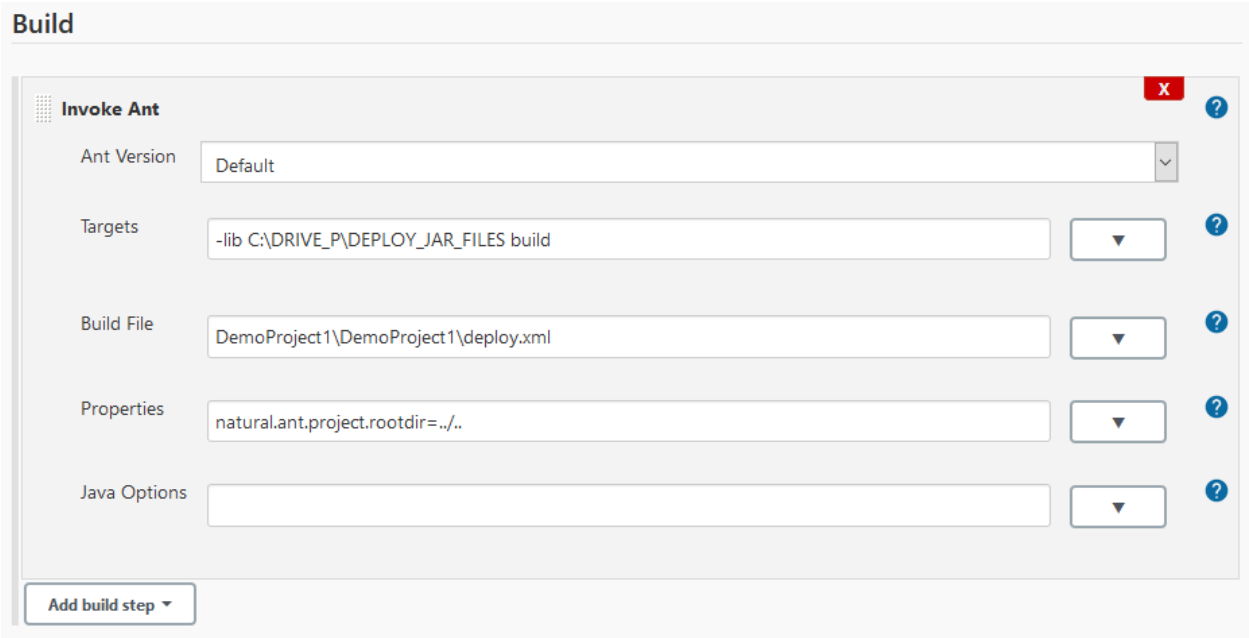


One of the first things that needs to be defined in a new freestyle job is the versioning repository connection which can look like this:



As usual, the repository URL, the credentials to access that repository, and the branch that is to be checked out need to be defined. Please pay attention to the definition of an additional behavior which specifies a special subdirectory for the checkout. As explained in **Versioning Repository Handling**, it is necessary for the Natural deployment script using GIT as the repository type to add an additional directory above the project directory. This additional directory must have the same name as the project directory itself.

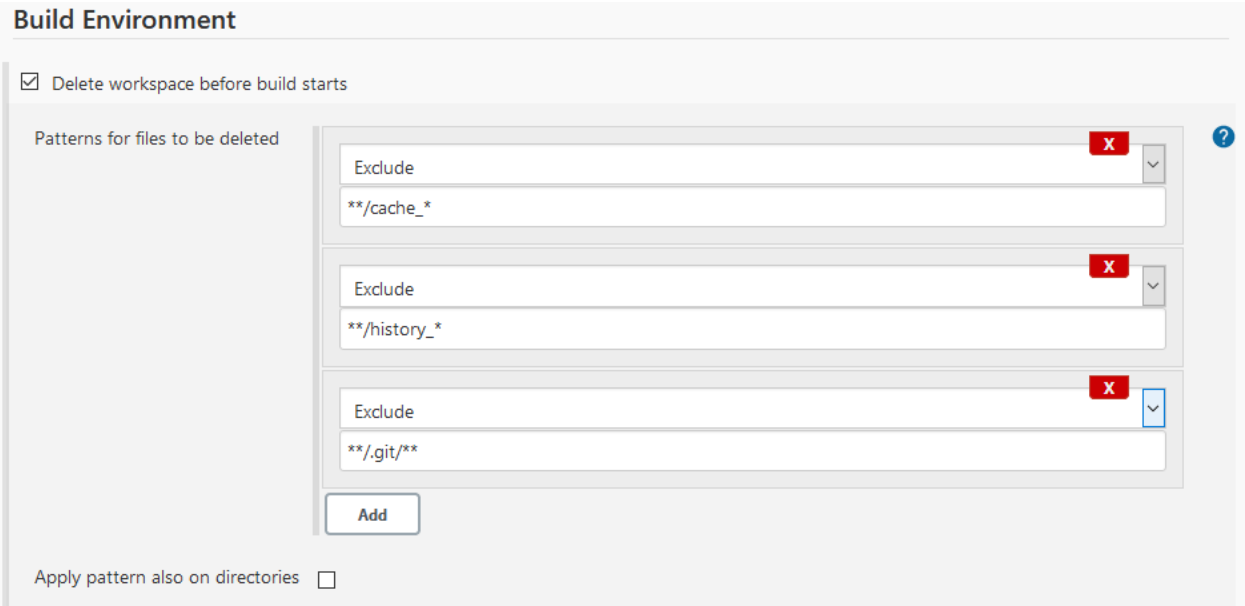
Later in the Jenkins job you must define the build task which eventually executes the Natural deployment script:



The directory with the JAR files and the target of the Natural deployment script are specified in the `Targets` field. Because the project has been checked out into a directory with the same name as the project name, the deployment script is located in a *<projectdir>/<projectdir>* directory. Finally, the project root directory used inside the Natural deployment script must be set to the top-level directory. This can be done with a relative path as shown in the example but you could also specify it via the Jenkins environment variable `WORKSPACE` as *natural.ant.project.rootdir=${WORKSPACE}*.

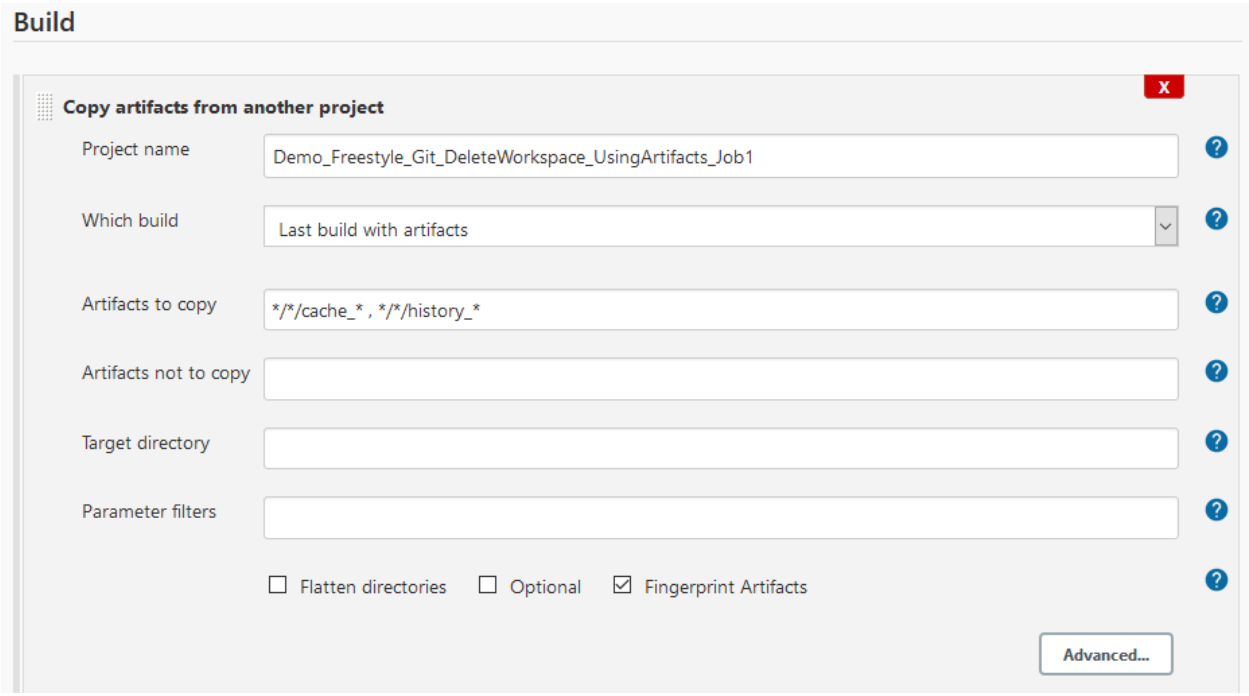
Deleting the Workspace

In case it is necessary to delete the Jenkins workspace between two consecutive job runs you must make sure to save the files mentioned in **Files Generated by Natural Deployment Scripts**. If you use the Workspace Cleanup plugin to delete the workspace, you could configure it to exclude specific files from deletion like shown in this example:



When excluding files from deletion it is also important to exclude the *.git* directory from deletion. Otherwise when cloning the repository next time, GIT will show an error.

Alternatively, you could also save the files as artifacts after a run of the Natural deployment script and copy the saved artifacts back in the next job run after the workspace has been deleted. You will need the Jenkins Copy Artifact plugin for this and you will need to configure it as shown to copy the files back from the artifact store:



You must specify the same project name, select from which build the artifacts should be taken and then specify all files you want to copy back into the workspace. Then you must add a post-build action to store the artifacts after the run of the deployment script:

Jenkins Pipeline Jobs

Defining a Jenkins pipeline job that performs a similar task to the freestyle job on the same NaturalONE project is accomplished the following way:

All tasks that have to be specified separately in the freestyle job are now integrated into the pipeline script. This means the pipeline script defines stages for cleanup workspace, checkout, build, etc.

Here is an example of a full script that also performs the save and restore of the Natural deployment files as artifacts:

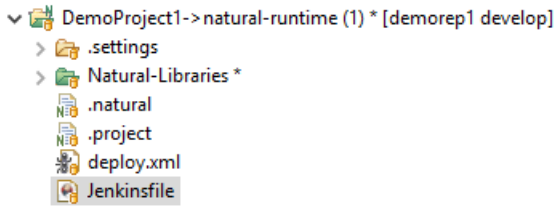
```
pipeline {

    agent any

    options {
        buildDiscarder logRotator(
                    daysToKeepStr: '15',
                    numToKeepStr: '10'
            )
    }

    environment {
        APP_NAME = "DemoProject1"
        APP_ENV  = "DEV"
    }

    stages {

        stage('Cleanup') {
            steps {
                cleanWs()
                bat """
                echo "Clean-up of Workspace for ${APP_NAME}"
                """
            }
        }

        stage('Checkout') {
            steps {
                checkout([
                    doGenerateSubmoduleConfigurations: false,
                    extensions: [
                        [$class: 'RelativeTargetDirectory',
                        relativeTargetDir: 'DemoProject1']],
                    submoduleCfg: [],
                    $class: 'GitSCM',
                    branches: [[name: '*/develop']],
                    userRemoteConfigs:
                          [[credentialsId: 'f1a44b44-bc76-43f9-af44-805cf8b6fccb',
                           url: 'http://irepo.eur.ad.sag/scm/~stco/demorep1.git']]
                ])
            }
        }

        stage("Restore Cache") {
            steps {
                script {
                    try {
                        copyArtifacts(projectName: currentBuild.projectName,
                                      filter:'*/*/cache_*',
                                      selector: lastSuccessful())
                        copyArtifacts(projectName: currentBuild.projectName,
```

```
                                        filter:'*/*/history_*',
                                        selector: lastSuccessful())
                    }
                    catch(err) {/*empty*/}
                }
            }
        }

        stage('Build') {
            steps {
                withAnt(installation: 'Ant Installation') {
                    dir("DemoProject1/DemoProject1") {
                        script {
                            if (isUnix()) {
                               sh 'ant -f deploy.xml -lib /usr/lib/deploy_jar_files
                                         build -Dnatural.ant.project.rootdir=../..'
                            } else {
                                bat 'ant -f deploy.xml -lib  ↩
C:\\DRIVE_P\\DEPLOY_JAR_FILES
                                         build -Dnatural.ant.project.rootdir=../..'
                            }
                        }
                    }
                }
            }
        }

        stage("Save Cache") {
            steps {
                archiveArtifacts('*/*/cache_*')
                archiveArtifacts('*/*/history_*')
            }
        }

    }
}
```

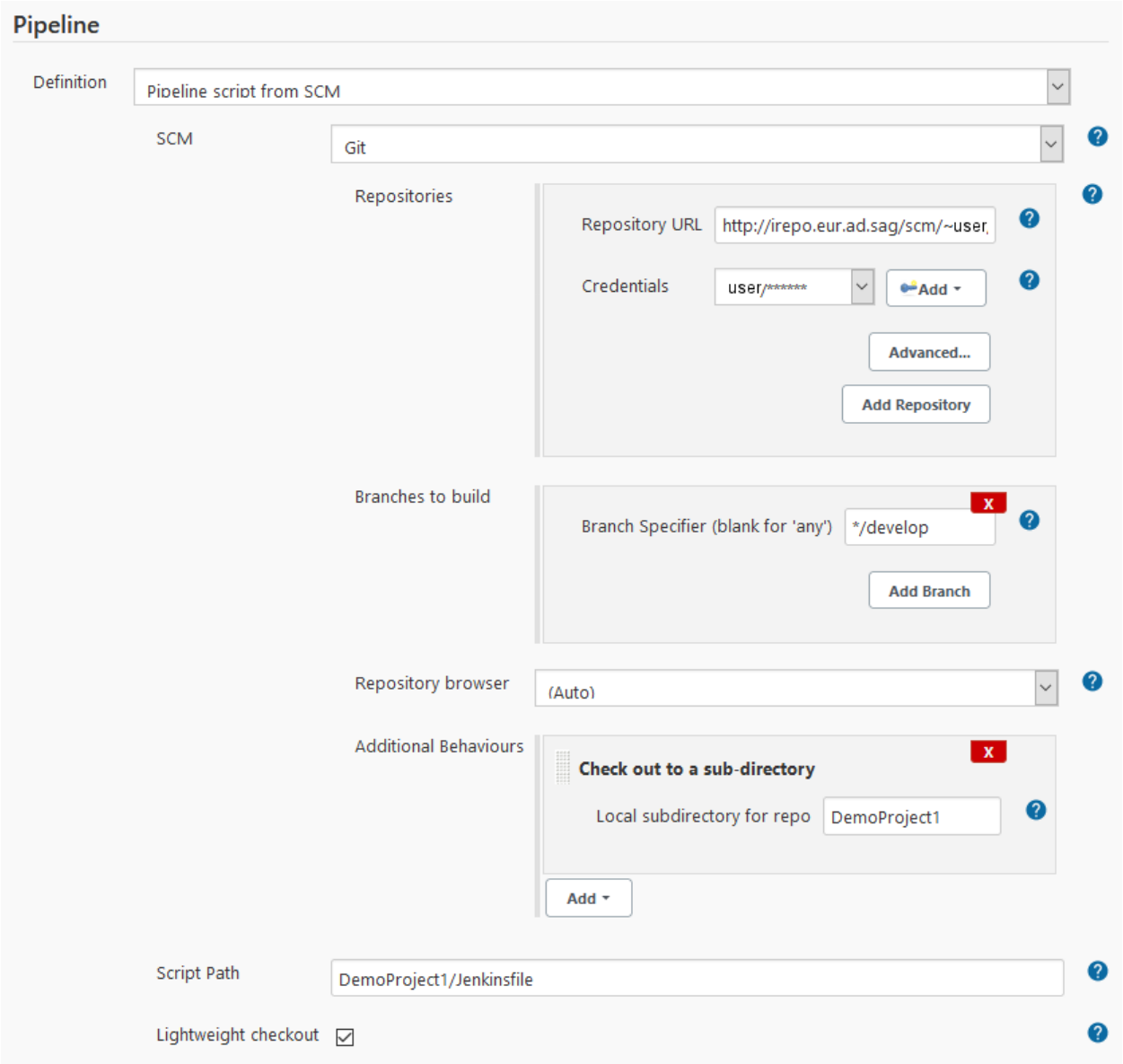
Using Jenkinsfile

One important benefit of Jenkins pipeline jobs compared to Jenkins freestyle jobs is that you specify the build, test and deployment phases as code and not as a document on a Web server. This code can be reviewed in the same way as the code of the application itself.

To go one step further, you can add the pipeline script code to the same versioning repository that also maintains your application code and your Natural deployments scripts. Then the project would look like this:

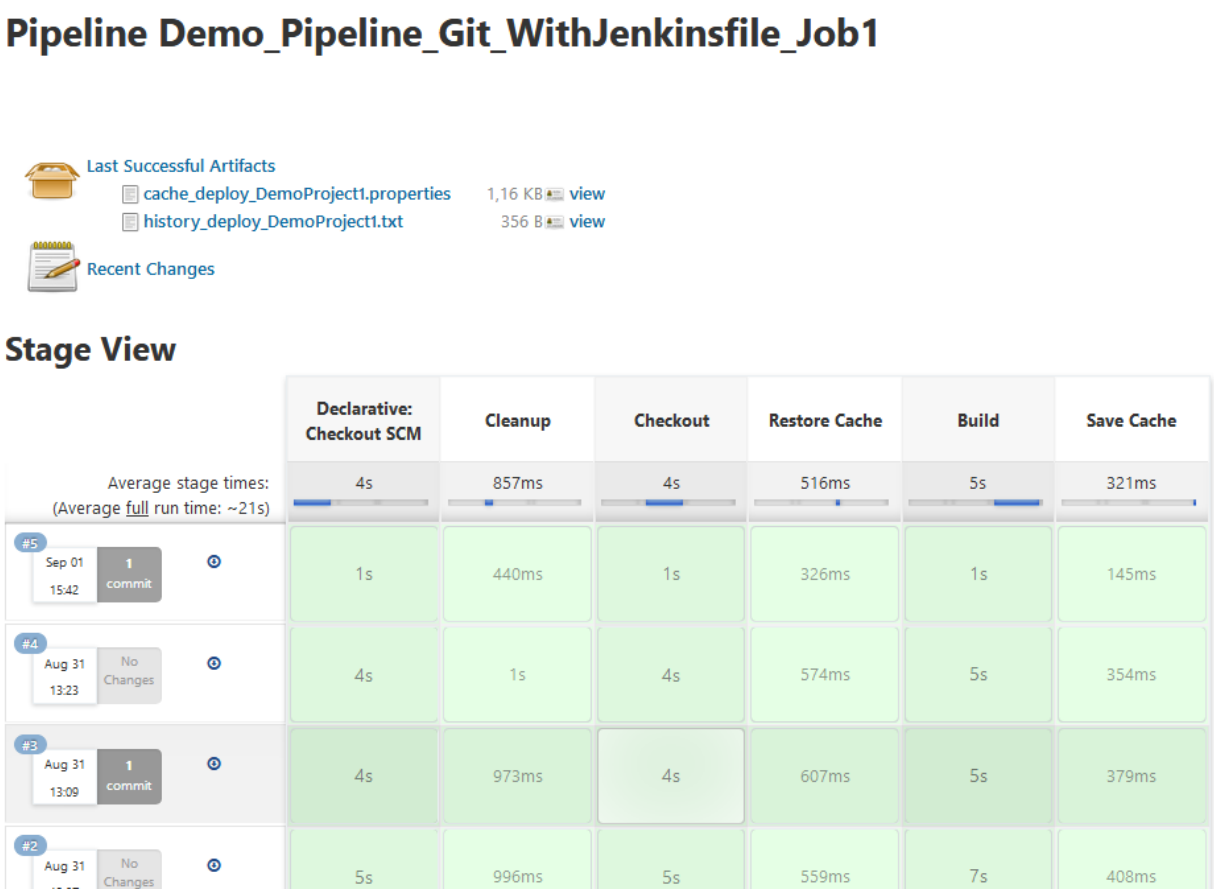


Here the Jenkins pipeline script from **Jenkins Pipeline Jobs** has been checked in as Jenkinsfile at the top level of the project. You can then create a Jenkins pipeline job with the following definition:



You must only specify where to find the Jenkinsfile in the repository. When the job is run, Jenkins extracts the Jenkinsfile and executes it. You can make changes to the Jenkinsfile the same way you would make changes to the application. After committing to the versioning repository, Jenkins

will automatically extract the newest file during the next job run. The result of such a job run is shown in the Web UI of Jenkins:



34 Deploying Java Applications

- General Information 492
- Using the Deployment Wizard for Java Applications 492
- Starting the Deployment from Eclipse 496
- Starting the Deployment from the Command Line 496
- Status Code Handling 498

General Information

You can deploy a Java application in a way similar to deploying a Natural application. Prerequisites:

- The Java project must be available as a project in Eclipse.
- The Java project must contain an Ant build script. Such a build script can be generated, for example, with the export functionality of Eclipse.
- The prerequisites defined by the Ant script (such as Java compiler or Eclipse installations) must be available when you deploy the Java application with NaturalONE.

The purpose of the NaturalONE Java deployment is not the generation of some build script for Java projects. This has to be done separately, with other tools (such as Eclipse). The purpose of the NaturalONE Java deployment is to add some base functionality, such as versioning repository access, and to make use of the already existing build scripts.

Using the Deployment Wizard for Java Applications

The deployment wizard creates a Java deployment file in your project root. This is an Ant script. You can create one or more Java deployment files for a project, and you can also load an existing Java deployment file and modify the current settings.

To use the deployment wizard

1. In the **Navigator** view or in the **Natural Navigator** view, select the Java project for which you want to create the deployment file.

   Or:

   If you want to load the settings of an existing Java deployment file, select this file in the **Navigator** view or in the **Natural Navigator** view.
2. From the **File** menu or from the context menu, choose **New > Other**.
3. In the resulting **New** dialog box, expand the **Natural** node, select **Deploy Natural Java Ant** and then choose the **Next** button.

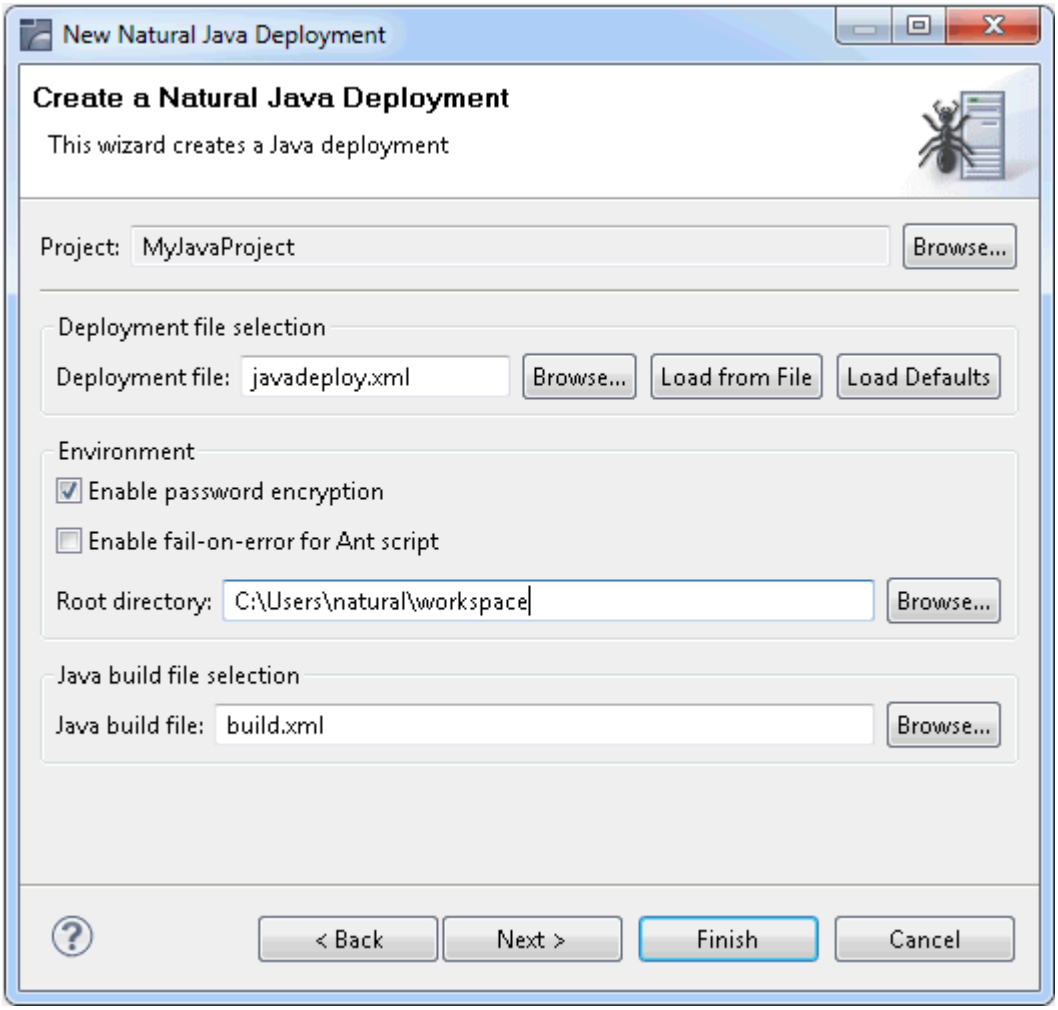
   The first page of the wizard appears (see below).
4. Specify all required information as described in the topics below. Use the **Next** button repeatedly to proceed from the first page of the wizard to the last page.
5. When all required information has been provided, choose the **Finish** button.

The different pages of the deployment wizard are described in the following topics:

- General Settings
- Repository

General Settings

On the first page of the wizard, you define general settings for the deployment.



Deployment file

The default name for the Java deployment file is *javadeploy.xml*. This name is shown in this text box when an existing Java deployment file was not selected while invoking the wizard. However, when an existing Java deployment file was selected, the name of the selected file is shown and the settings from this file are automatically loaded.

You can enter any other name for your new deployment file. It is recommended that your new deployment file also has the extension ".xml".

Note: If you keep the name *javadeploy.xml*, the settings from an existing Java deployment file with the same name are loaded the next time you select the project and invoke the wizard.

If you want to load an existing Java deployment file, choose the **Browse** button. A dialog appears, providing for selection all Java deployment files in the current project. Next, you have to choose the **Load from File** button. Otherwise, the settings in this file are not shown in the wizard and may thus be overwritten unintentionally.

If you want to return to the default settings of the deployment wizard (this also includes the information that can be specified on the other pages of the wizard), choose the **Load Defaults** button.

Enable password encryption
When enabled, all passwords that are used in the deployment file are stored in an encrypted format.

Enable fail-on-error for Ant script
When enabled, the Ant script reports errors and terminates in the case of a build failure.

When disabled, the Ant script still reports errors but build failures are only triggered in severe situations (see also ***Status Code Handling***).

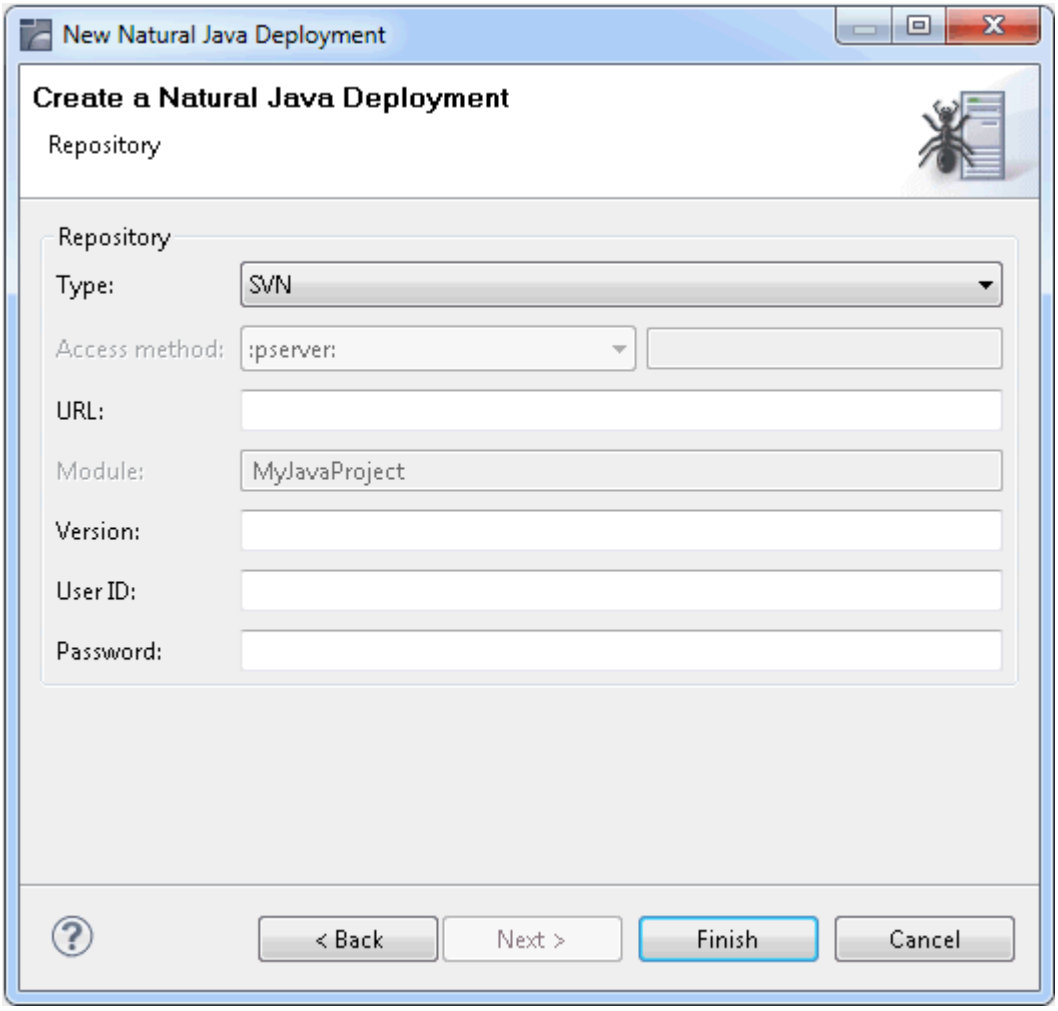
Root directory
This path should only be changed when you intend to start the deployment from the command line (that is, when the deployment is not to be started from Eclipse).

Specify the directory in which the selected project is to be checked out and where the processing takes place. When the deployment is supposed to run on a different machine, you can insert the desired root path via copy-and-paste.

Java build file
Specify the name of the Ant build file that is to be used for the deployment of the selected Java project. This build script must have been generated with some other tool in advance. You can also choose the **Browse** button to select the build file from a dialog box; in this case, all existing build files in the current Java project are provided for selection.

Repository

On the second page of the wizard, you define all settings related to the versioning repository. This can be either Subversion (SVN), GIT or CVS.



From the **Type** drop-down list box, select the type of versioning repository that you are using, and then specify all required information. The names of the text boxes and their availability changes according to the selected type.

The wizard usually collects a set of default information as given for the selected project. In most cases, only minor corrections have to be made to the defaults, for example, user ID and password may have to be provided.

Starting the Deployment from Eclipse

When you start the deployment process from Eclipse, it is not possible to execute the `checkout` and `update` targets of the deployment file since these targets would access the versioning repository, and this is not feasible from within an Eclipse environment. If you want to check out a specific revision from the versioning repository or if you want to update your project with sources from the versioning repository, you have to start the deployment from the command line as described below.

For testing purposes, for example, it is helpful to start the deployment process from Eclipse. Since the Java deployment file is an Ant script, the built-in Eclipse functionality of starting Ant scripts is used here. The `build` target of the Ant script will then be executed.

To start the deployment from Eclipse

- In the **Navigator** view or in the **Natural Navigator** view, select your Java deployment file, invoke the context menu and choose **Run As > Ant Build**.

  The deployment process is started, and the output of the deployment file is written to the **Console** view.

  Note: If you want to change the limit for the console output, you can do this in the general Eclipse preferences under **Run/Debug > Console**.

Starting the Deployment from the Command Line

You can start the deployment process from a Windows command line such as the Command Prompt (*cmd.exe*) or from a shell command line on a Linux system. When you start the deployment from the command line, special requirements must be met.

The following topics are covered below:

- Prerequisites

- Starting the Deployment

Prerequisites

The prerequisites for deploying a Java application are the same as for deploying Natural applications. See ***Prerequisites*** in *Deploying Natural Applications*. However, you have to copy the Java deployment file to the root directory (instead of the Natural deployment file).

In addition, all necessary tools which are referred to in the Ant build file (specified on the first page of the wizard) must be available and accessible on the processing platform.

Starting the Deployment

When all prerequisites are in place, the deployment can be started by issuing specific Ant calls. This section just provides some examples (where the default name *javadeploy.xml* is used).

- Print the help screen of the Ant script:

```
ant -lib path-to-mylib -f javadeploy.xml help
```

- Perform an initial checkout of the project sources from the versioning repository:

```
ant -lib path-to-mylib -f javadeploy.xml checkout
```

- Perform an update of the project sources from the versioning repository:

```
ant -lib path-to-mylib -f javadeploy.xml update
```

- Deploy the sources using the Ant build file which was specified on the first page of the wizard:

```
ant -lib path-to-mylib -f javadeploy.xml build
```

- With a single call, perform an update of the project sources from the versioning repository first, and then deploy the sources:

```
ant -lib path-to-mylib -f javadeploy.xml update build
```

- In the above examples, the logging information is written to standard output. If logging information is to be written to a file, use a call such as the following:

```
ant -lib path-to-mylib -f javadeploy.xml update build -logfile mylogfile.txt
```

Status Code Handling

The Ant deployment script for Java can run in two status code modes. The mode can be toggled by specifying the command line parameter `-Dnatural.ant.java.failonerror` as described in the following table.

Command Line Option	Description
`-Dnatural.ant.java.failonerror=no`	Only severe errors such as missing project directories will lead to a build failure with a status code other than 0. This is the default mode.
`-Dnatural.ant.java.failonerror=yes`	In addition to the severe errors described above, errors occurring during checkout or update will also lead to a status code other than 0 and hence will lead to a build failure.

Note: The default mode can also be changed on the **first page** of the deployment wizard.

When the additional status code handling has been enabled, the Ant tool as well as the internally used tools such as SVN, GIT or CVS clients may issue specific status codes. In case the status codes are unclear, refer to the documentation of these tools.

35 Using a Master Deployment

- General Information 500
- Using the Deployment Wizard for a Master Deployment 500
- Starting the Deployment from Eclipse 505
- Starting the Deployment from the Command Line 505

General Information

With a master deployment, you can combine the different types of deployment which are available with NaturalONE into a single common process. A master deployment is an Ant script that simply runs any existing project-specific Ant scripts in a configurable manner. Prerequisites:

- The deployment scripts of the projects that you want to include in the master deployment must have been generated previously.
- All prerequisites that apply for each of the selected projects also apply for the master deployment. See the corresponding deployment chapters in this NaturalONE documentation for further information.

The wizard for a master deployment collects information about the deployments to be processed and creates a full deployment kit, containing all necessary Ant scripts as well as the necessary JAR files used for deployment.

Note: With a master deployment, it is no longer required to manually copy the JAR files and the files that are created by the different deployment wizards. When you specify the corresponding options for the creation of the external deployment package, these files are automatically added to the deployment package. You can also specify to create a zip archive which then contains the required JAR files.

Using the Deployment Wizard for a Master Deployment

The deployment wizard creates the following files:

- The master deployment file. This is an Ant script for starting the deployment process. It is created in the root directory of the selected project. You can create one or more master deployment files, and you can also load an existing master deployment file and modify the current settings.
- Zip archive containing the deployment files for all selected projects and, if the corresponding option was selected, the JAR files necessary for running the deployment outside Eclipse. This zip archive is only created when the corresponding option is enabled in the deployment wizard. It is created in the package directory that has been specified in the wizard.

  Note: In the zip archive, the deployment files, which are taken from the different projects, are renamed so that they include the project name in the file name.

To use the deployment wizard

1 In the **Navigator** view or in the **Natural Navigator** view, select the project in which you want to create the master deployment file.

   Any type of project can be used here. It is not required that you select a Natural or Java project. For example, this may be a project that you have created for the sole purpose of holding the master deployment script.

   Or:

   If you want to load the settings of an existing master deployment file, select this file in the **Navigator** view or in the **Natural Navigator** view.

2 From the **File** menu or from the context menu, choose **New > Other**.

3 In the resulting **New** dialog box, expand the **Natural** node, select **Deploy Natural Master Ant** and then choose the **Next** button.

   The first page of the wizard appears (see below).

4 Specify all required information as described in the topics below. Use the **Next** button repeatedly to proceed from the first page of the wizard to the last page.

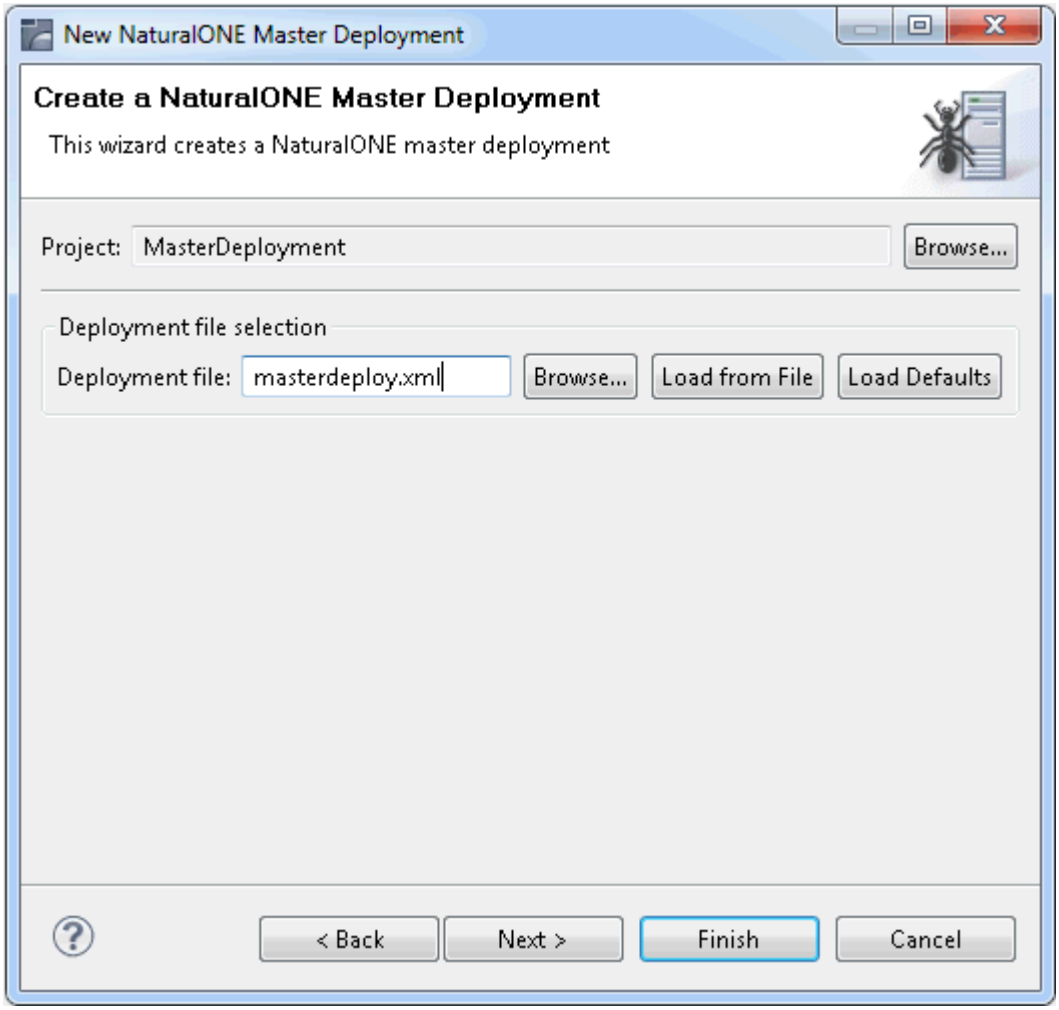
5 When all required information has been provided, choose the **Finish** button.

The different pages of the deployment wizard are described in the following topics:

- General Settings
- Application Selection
- External Package Creation

General Settings

On the first page of the wizard, you can define a different name for the master deployment file.



Deployment file

The default name for the master deployment file is *masterdeploy.xml*. This name is shown in this text box when an existing master deployment file was not selected while invoking the wizard. However, when an existing master deployment file was selected, the name of the selected file is shown and the settings from this file are automatically loaded.

You can enter any other name for your new deployment file. It is recommended that your new deployment file also has the extension ".xml".

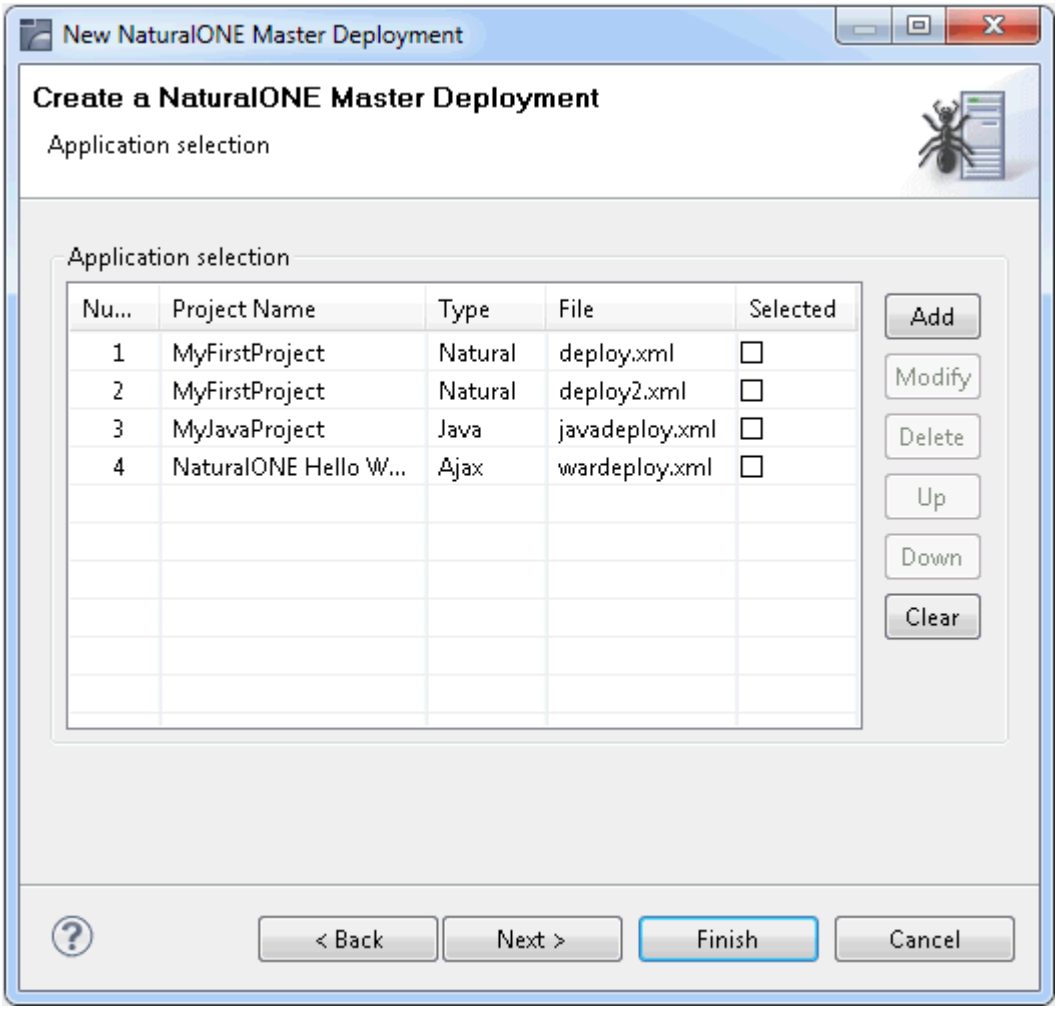
Note: If you keep the name *masterdeploy.xml*, the settings from an existing master deployment file with the same name are loaded the next time you select the project and invoke the wizard.

If you want to load an existing master deployment file, choose the **Browse** button. A dialog appears, providing for selection all master deployment files in the current project. Next, you have to choose the **Load from File** button. Otherwise, the settings in this file are not shown in the wizard and may thus be overwritten unintentionally.

If you want to return to the default settings of the deployment wizard, choose the **Load Defaults** button.

Application Selection

On the second page of the wizard, you select the deployment files that are to be included in the master deployment (by selecting the check boxes in the **Selected** column). By default, all deployment files that can be found in your workspace are listed on this page.

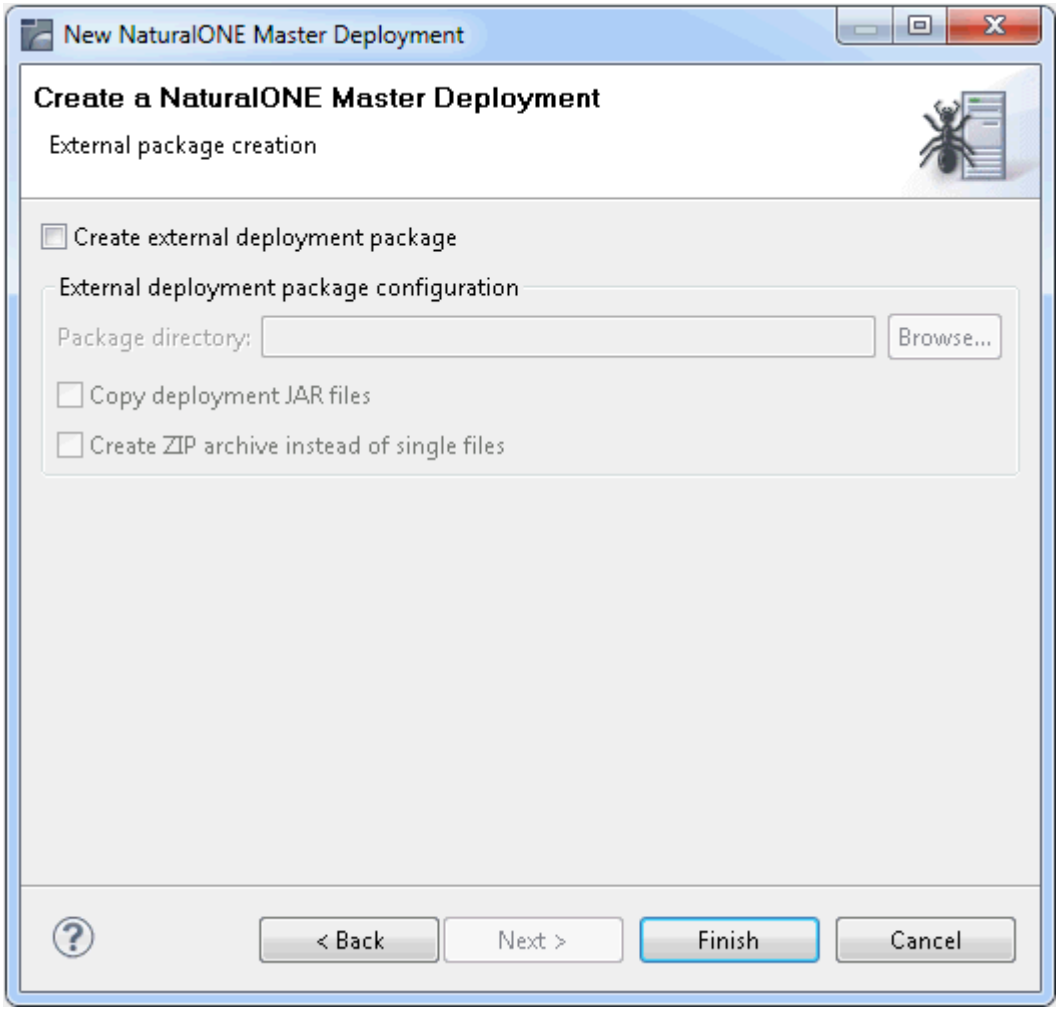


Using the buttons to the right of the list of deployment files, you can change the sequence of the deployment files, you can delete deployment files and you can add them again. You can also modify the name of a deployment file.

External Package Creation

On the third page of the wizard, you specify whether an external deployment package is to be created. If you want to start the deployment outside of Eclipse, you have to create such a package.

An external deployment package consists of all necessary Ant deployment scripts (as determined by the selected projects on the previous wizard page) and optionally the set of necessary JAR files.



Create external deployment package

When selected, all other options on this page are enabled. It is important that you specify the package directory.

When not selected, only the master deployment file will be created in the root directory of the selected project. In this case, the deployment can only be started from within Eclipse.

Package directory

The path to the directory to which the deployment files are to be copied.

Copy deployment JAR files
When enabled, the JAR files which are listed under ***Prerequisites*** in the section *Deploying Natural Applications* are added to the deployment package.

Create ZIP archive instead of single files
When enabled, a zip archive is created instead of single files. The zip archive has the same name as the master deployment file, but with the extension ".zip".

Starting the Deployment from Eclipse

When you start the deployment process from Eclipse, it is not possible to execute the `checkout` and `update` targets of the deployment file since these targets would access the versioning repository, and this is not feasible from within an Eclipse environment. If you want to check out a specific revision from the versioning repository or if you want to update your project with sources from the versioning repository, you have to start the deployment from the command line as described below.

For testing purposes, for example, it is helpful to start the deployment process from Eclipse. Since the master deployment file is an Ant script, the built-in Eclipse functionality of starting Ant scripts is used here. The `build` target of the Ant script will then be executed.

To start the deployment from Eclipse

- In the **Navigator** view or in the **Natural Navigator** view, select your master deployment file, invoke the context menu and choose **Run As > Ant Build**.

  The deployment process is started, and the output of the deployment script is written to the **Console** view.

  Note: If you want to change the limit for the console output, you can do this in the general Eclipse preferences under **Run/Debug > Console**.

Starting the Deployment from the Command Line

You can start the deployment process from a Windows command line such as the Command Prompt (*cmd.exe*) or from a shell command line on a Linux system. When you start the deployment from the command line, special requirements must be met.

The following topics are covered below:

- Prerequisites

- Starting the Deployment

Prerequisites

The prerequisites for a master deployment are the same as for deploying Natural applications. See ***Prerequisites*** in *Deploying Natural Applications*. However, you do not have to copy any files manually. This is automatically done by the master deployment.

Make sure that all necessary tools which are referred to in the project-specific build scripts are available and accessible on the processing platform.

Make sure that the files from the **external package** are in place. If you have chosen to create a zip archive, you have to unpack this file.

Starting the Deployment

When all prerequisites are in place, the deployment can be started by issuing specific Ant calls. This section just provides some examples (where the default name *masterdeploy.xml* is used). It is assumed that the JAR files have been placed in the directory in which you are currently working.

- Print the help screen of the Ant script:

```
ant -lib . -f masterdeploy.xml help
```

- Perform an initial checkout of all sources from all selected projects from the versioning repositories:

```
ant -lib . -f masterdeploy.xml checkout
```

- Perform an update of all sources from all selected projects from the versioning repositories:

```
ant -lib . -f masterdeploy.xml update
```

- Call the `build` targets of the selected project-specific deployment scripts:

```
ant -lib . -f masterdeploy.xml build
```

- With a single call, perform an update of all sources from all selected projects from the versioning repositories, and then call the `build` targets of the selected project-specific deployment scripts:

```
ant -lib . -f masterdeploy.xml update build
```

- In the above examples, the logging information is written to standard output. If logging information is to be written to a file, use a call such as the following:

```
ant -lib . -f masterdeploy.xml update build -logfile mylogfile.txt
```

Note: When using the `checkout`, `update` and `build` targets of the master deployment file, the master deployment file actually makes calls into the corresponding project-specific deployment files for Natural, Java and/or Ajax.

XVII Setting the Preferences

36 Setting the Preferences

- Showing the Natural-Specific Preferences 512
- Natural 513
- Appearance 528
- Label Decorations 528
- Data Browser 532
- Debug 532
- Debug Attach Settings 533
- Display Options 533
- Editors 534
- DDM Editor 534
- Map Editor 535
- Object Templates 539
- Source Editor 540
- NATstyle 548
- Natural Navigator 550
- Parser 551
- Profiler 555
- Regional Settings 559
- Runtime Execution 562
- File Transfer 565
- Natural I/O 566
- SSL/TLS 568
- Tomcat 570
- XML Toolkit 570

Showing the Natural-Specific Preferences

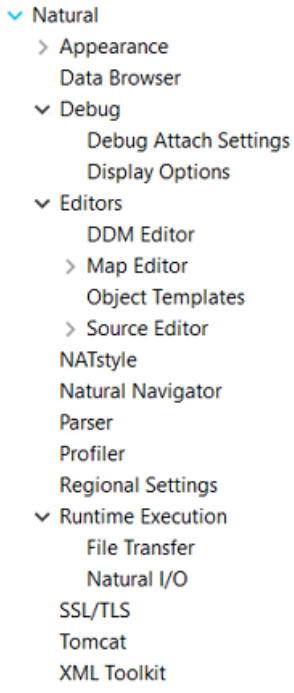
The Natural-specific preferences are set in the **Preferences** dialog box of Eclipse.

To show the Natural-specific preferences

1 From the **Window** menu, choose **Preferences**.

2 In the tree of the resulting dialog box, expand the **Software AG** node and then the **Natural** node.

Different pages are provided for setting different types of preferences.



Note: When optional components of NaturalONE have been installed, additional nodes may be shown when you expand the **Software AG** node, in parallel to the **Natural** node. For example, when **Service Development** has been selected in the installer, a **Business Services** node and a **Code Generation** node are shown. For detailed information on the preferences which are available from such a node, see the documentation for the corresponding optional component.

3 Select one of the pages in the tree and set the required options as described in the topics below.

Note: When you choose the **Restore Defaults** button on a page which contains several tabs, the defaults are restored for all tabs which belong to this page (not only for the currently visible tab).

4 Choose the **OK** button to save your changes and to close the dialog box.

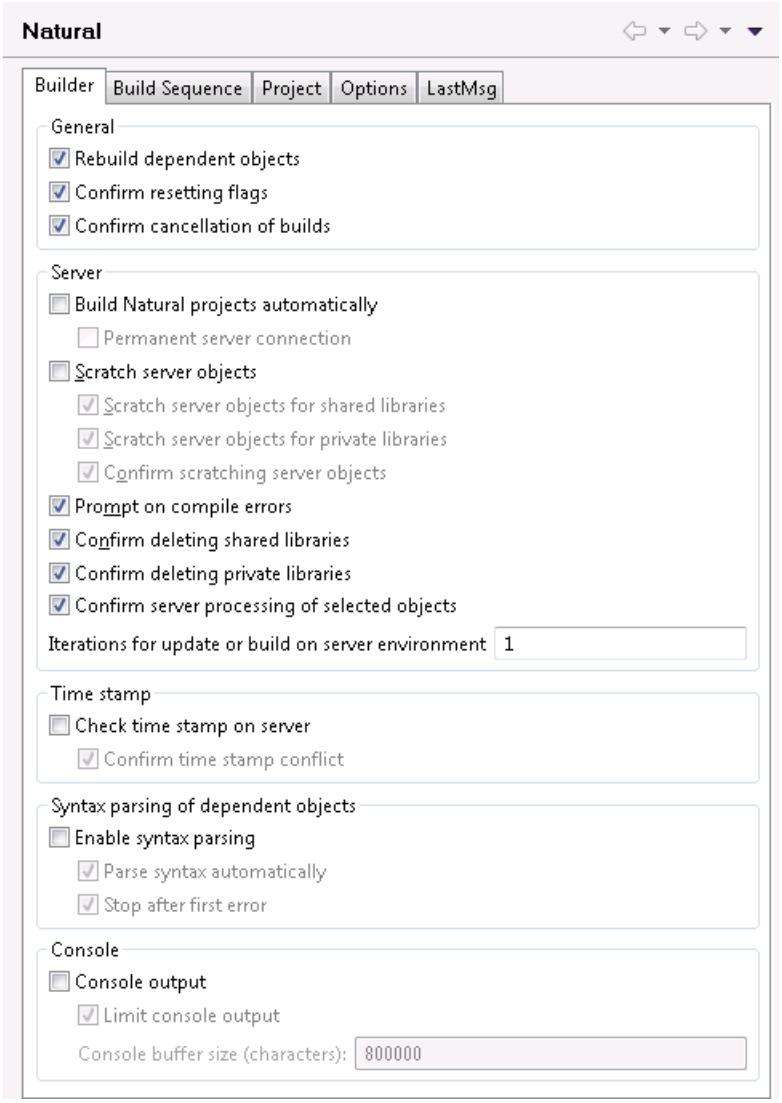
Natural

Several tabs are provided on the **Natural** page:

- Builder
- Build Sequence
- Project
- Options

- LastMsg

Builder



Rebuild dependent objects

When selected (default), only referencing Natural sources that require the modified copycode, data area or DDM at catalog time are cataloged in the Natural environment after the copycode, data area or DDM has been changed and uploaded to the Natural environment. Furthermore, if the `PCHECK` profile parameter is set in the **parser options** of the project properties, all referencing Natural sources are cataloged.

Confirm resetting flags

When selected (default), a dialog box appears when you **reset the flags**, which are part of the label decorations, in the Eclipse workspace. This dialog box asks whether you really want to reset the flags.

The dialog box contains the **Always reset without prompt** check box. When you select this check box in the dialog box, **Confirm resetting flags** is automatically deselected in the preferences.

Confirm cancelation of builds

When selected (default), a dialog box appears when you are about to cancel a build. This dialog box asks whether you really want to cancel the current build of the project.

The dialog box contains the **Always cancel the build without prompt** check box. When you select this check box in the dialog box, **Confirm cancelation of builds** is automatically deselected in the preferences.

For further information, see *Canceling a Build*.

Build Natural projects automatically

When selected, the appropriate Natural environment is automatically updated each time you save a source or add a new source. The source is uploaded to the Natural environment and is stowed there. The library into which a source is written and stowed is determined by the mode you have defined for the project (either shared mode or private mode). See *Steplibs* in *Changing the Project Properties* for further information.

When deselected (default), you have to update the Natural environment manually. See *Updating the Objects in the Natural Environment* for further information.

Permanent server connection

Only available when **Build Natural projects automatically** is selected.

When selected, a permanent connection to the Natural environment is established.

When deselected (default), a connection to the Natural environment is only established when information needs to be uploaded or downloaded.

Scratch server objects

Depending on which of the following options is selected (**Scratch server objects for shared libraries** or **Scratch server objects for private libraries** or both), this description applies to either private mode libraries or to shared mode libraries of a Natural environment or to both.

When selected, each object that is **renamed** or **deleted** in the Eclipse workspace is also renamed or deleted in the Natural environment when this environment is updated. The source and (if available) the generated object are then deleted in the Natural environment. This applies when the **Build Natural projects automatically** option is set (see above) or when the **Build Natural Project** command is used (see *Updating the Objects in the Natural Environment*).

When you delete objects, this option has only effect when you select single objects in the Eclipse workspace. When you select one of the container nodes (the node for a project, library or subfolder), the objects are only deleted in the Eclipse workspace, but not in the Natural environment. This is important if you want to remove projects or libraries from the Eclipse workspace, but do not want to remove the corresponding libraries in the Natural environment. If required, you have to delete the objects in the **Natural Server** view manually.

When deselected (default), the objects in the Natural environment are not affected when the environment is updated. The objects that have been renamed in the Eclipse workspace are then available in the Natural environment, in addition to the original objects. The objects that have been deleted in the Eclipse workspace are still available in the Natural environment. Be aware of the fact that the objects which are still available in the Natural environment may still be referenced by other programs; this may cause problems.

Scratch server objects for shared libraries

Only available when **Scratch server objects** is selected.

The Natural objects will be deleted or renamed according to the description above in the shared libraries (when working in shared mode). When working in private mode, the shared library objects are not deleted or renamed.

Scratch server objects for private libraries

Only available when **Scratch server objects** is selected.

The Natural objects will be deleted or renamed according to the description above in the private libraries (when working in private mode).

Confirm scratching server objects

Only available when **Scratch server objects** is selected.

When selected, a dialog box appears when you use the **Clean** command from the **Project** menu, or when the Natural environment is updated after you have **renamed** or **deleted** objects in the Eclipse workspace (that is, when the **Build Natural projects automatically option** is set or when the **Build Natural Project** command is used). In this dialog box, you are asked whether you really want to scratch (that is, delete) the corresponding object(s) from the Natural environment. Up to 10 affected objects are listed in this dialog box.

The dialog box contains the **Always scratch without prompt** check box. When you select this check box in the dialog box, **Confirm scratching server objects** is automatically deselected in the preferences.

Prompt on compile errors

When selected, a message box appears when a compile error occurs during a **manual update** of the Natural environment (that is, during the execution of an **Upload**, **Update** or **Build Natural Project** command). This message box shows the corresponding Natural error message. You can choose one of the following command buttons:

- **Continue**
  When you choose this button, you confirm the current error. When further objects have been selected for update, the processing will continue.
- **Cancel**
  When you choose this button, the processing of further objects will be canceled.

The message box contains the **Do not show this message again** check box. When you select this check box in the message box, **Prompt on compile errors** is automatically deselected in the preferences.

This option only applies to manual updates. The message box does not appear for compile errors which occur during an **automatic update** (that is, when the **Build Natural projects automatically** option is enabled in the Natural preferences).

Confirm deleting server libraries

When selected, an additional dialog box appears when you rebuild a project in the Eclipse workspace and you select the **Delete the contents of the affected libraries on the server first** check box in the **Rebuild Natural Project** dialog box (see ***Rebuilding all Objects in the Natural Environment***). In the additional dialog box, you are then asked whether you really want to delete the entire contents of the affected libraries from the Natural environment. Up to 10 affected libraries are listed in this dialog box.

The additional dialog box contains the **Always delete without prompt** check box. When you select this check box in the dialog box, **Confirm deleting server libraries** is automatically deselected in the preferences.

Confirm deleting private libraries

When selected, an additional dialog appears when you change the development mode in the **Steplibs** property page of a project from private to shared mode or when you set a different runtime environment in the **Runtime** property page of the project. Please refer to ***Private Mode*** and *Private-mode Libraries* for more information.

The additional dialog box contains the **Always delete private libraries without prompt** check box. When you select this check box in the dialog box, the private libraries will automatically be deleted from the runtime environment and **Confirm deleting private libraries** is deselected in the preferences.

Confirm server processing of selected objects

When selected (default), a dialog box appears when you are about to **update** the objects in the Natural server environment using the **Update** or **Upload** command. This dialog box asks whether you really want to update/upload the selected objects and replace the corresponding objects on the server.

The dialog box only appears if a Natural project, library, folder or multiple Natural objects are selected. If only a single Natural object is selected, the dialog box does not appear and the object is processed immediately.

The dialog box contains the **Always process selected objects without prompt** check box. When you select this check box in the dialog box, **Confirm server processing of selected objects** is automatically deselected in the preferences.

Iterations for update or build on server environment

When the **Update** or **Build Natural Project** command is selected, you can specify the number of iterations (1 to 9). This can help you to catalog all sources on the server in the proper order in one pass (see also ***Build Sequence***).

When you specify a number greater than 1, the sources containing errors are re-cataloged until either no more errors occur or the specified number of retries has been reached. The default is 1.

See ***Updating the Objects in the Natural Environment*** for further information.

Check time stamp on server

When selected, it is checked whether the source that is to be uploaded has been changed between download and upload. This check is done by comparing the time stamp of the source that is to be uploaded with the time stamp of the source that is to be overwritten on the server. For further information, see ***Checking the Time Stamps in the Natural Environment*** in *Modifying Objects in the Natural Environment or in the Repository*.

The following applies for debugging: when selected, the time stamp of the source in the local workspace is compared with the time stamp of the corresponding source on the server. When a time stamp conflict is found, a dialog box appears, asking whether you want to update the source in the workspace with the source from the server. See also ***Starting the Debugger***. When **Check time stamp on server** is selected, the above mentioned dialog box is always shown, regardless of the setting of the **Confirm time stamp conflict** option.

The **Check time stamp on server** option applies only for shared mode. When working in private mode or when using a private-mode library, the setting of this option is ignored. For more information on the different development modes, see ***Steplibs*** in *Changing the Project Properties*.

Note: This option is not used for the deployment of Natural applications. If you want to enable time stamp checking for the deployment, you have to do this in the deployment wizard for Natural applications. For further information, see ***Checking the Time Stamps in the Natural Environment*** in *Deploying Natural Applications*.

Confirm time stamp conflict

Only available when **Check time stamp on server** is selected. Applies only when transferring objects from the workspace to the server.

When selected, a dialog box appears when the time stamp of the source that is to be uploaded differs from the time stamp of the source that is to be overwritten on the server. You can then decide whether to proceed with the upload or not.

The dialog box contains the **Always skip objects without prompt (No To All)** check box. When you select this check box in the dialog box, **Confirm time stamp conflict** is automatically deselected in the preferences.

For further information, see ***Checking the Time Stamps in the Natural Environment***.

Enable syntax parsing

When selected, the local NaturalONE parser also considers dependent objects when Natural sources are modified and saved. A dependent object is, for example, a program which uses

the fields from a data area. If the data area is changed, this change must also be considered in the dependent program.

Parse syntax automatically

Only available when **Enable syntax parsing** is selected.

When selected (default), all dependent objects are automatically parsed after a Natural source has been saved. If the modification of a source leads to errors in dependent objects, error markers will be shown in the **Navigator** view or in the **Natural Navigator** view. In addition, a corresponding error message is shown in the **Problems** view. For example, when you rename a field in a local data area and then save the local data area, all programs which use the renamed field from the local data area are automatically shown with an error marker.

When deselected, it is possible to parse the dependent objects manually, using the **Parse All** command. For further information, see the information on manual parsing in *Parsing Dependent Objects*.

When you change the setting of this option, the setting of the **Parse Syntax Automatically** command is adapted accordingly. For further information, see the information on automatic parsing in *Parsing Dependent Objects*.

Stop after first error

Only available when **Enable syntax parsing** is selected.

When selected (default), parsing of dependent objects is stopped when the first error occurs in the current source. This reduces the number of error markers and thus improves the performance of syntax parsing and reduces memory consumption. In this case, only the first error marker is shown. If the source contains more than one syntax error, an additional information marker is shown which has the description "I_0004: Source has more than one error".

When you double-click the information marker in the **Problems** view (or when you select the marker and then choose **Go to** from the context menu), the source containing the first error is opened. The vertical ruler on the left side of the source editor then shows the information marker in the first line, and all error markers in the lines which cause that error.

The information marker will disappear when the source has less than two errors, or when you deselect the option **Stop after first error** and start syntax parsing once more.

Console output

When selected, Natural builder information is shown in the **Console** view each time you unload a project from the Natural environment or you save a source to the Eclipse workspace.

For example, when **Build Natural projects automatically** is disabled, a corresponding message is shown informing you that the changed source is waiting to be uploaded and stowed. On the other hand, when **Build Natural projects automatically** is enabled, you are informed that the changed source is uploaded and stowed.

Limit console output

Only available when **Console output** is selected.

When selected, the value defined in the **Console buffer size (characters)** text box is used to define the limit for the console output of the Natural builder. When the console output surpasses the specified maximum size, the output is truncated from the beginning of the buffer.

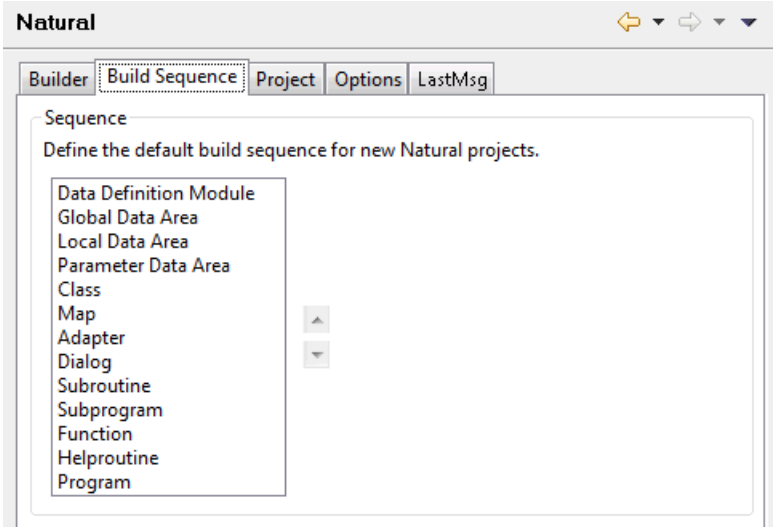
When deselected, the console output of the Natural builder is not limited.

Console buffer size (characters)

Only available when **Limit console output** is selected.

The value in this text box defines the maximum number of characters for the console output of the Natural builder. The default value is 800000.

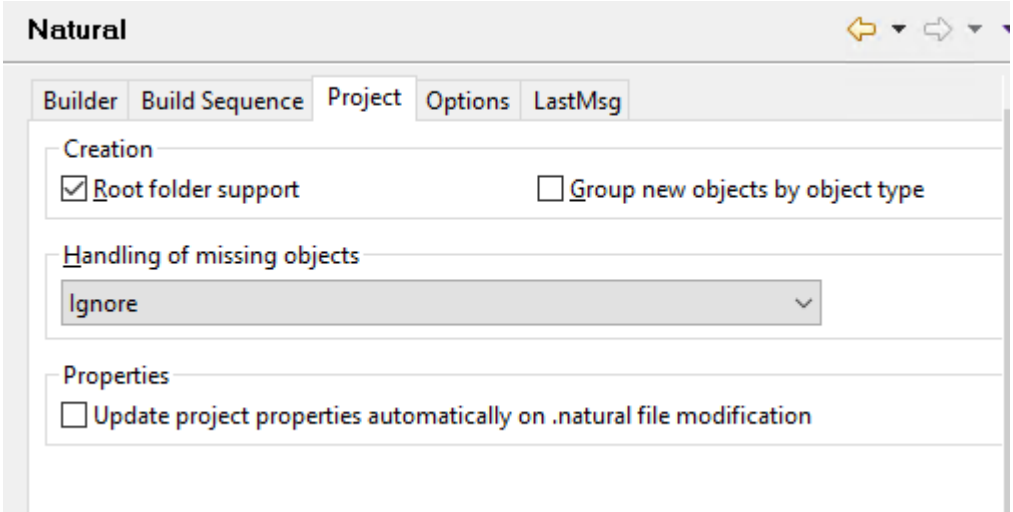
Build Sequence



If required, you can use the up-arrow and down-arrow buttons to change the build sequence for new Natural projects. This sequence is used by the **Build Natural Project** command to upload and stow objects in the Natural environment. See also ***Build Sequence***.

You can also define an individual build sequence for each Natural project in the project properties. See ***Builder*** in *Changing the Project Properties*.

Project



Root folder support

When selected (default), the additional option **Create the library root folder** is shown in the **New Natural Project** dialog box and in the **Add to New Natural Project** dialog box. When you select this additional option in the above mentioned dialog boxes, an additional folder with the name "Natural-Libraries" is created in the project. Any new libraries and objects that you create in this project are from now on always placed in the "Natural-Libraries" folder. Any libraries and objects that are not contained in this root folder are ignored when the project is built. See also *Libraries in a Natural Project*.

Other (optional) components of NaturalONE may also make use of root folders. For information on how such a component uses this option, see the documentation for this component.

Group new objects by object type

When selected, the **Group new objects by object type** option is automatically selected when you **create a Natural project** (either by using the wizard or by downloading a library or object from a Natural server into a new project).

When not selected (default), this option is not automatically selected when you create a Natural project.

For further information, see *Group Folders*.

Handling of missing objects

Pertains to the source editor, map editor, the **Dependencies** view and for debugging.

When your source code references, for example, a data area, copycode or DDM which the parser cannot find in your workspace, an error marker is shown in the source editor, and the corresponding message is shown in the **Problems** view. Missing objects are listed as "<Unknown>" in the **Dependencies** view.

Note: This option relates only to objects that are required by the parser (copycodes, data areas and DDMs). Other objects that are called by the program (such as subprograms, subroutines or maps) are not affected. In addition, the map editor uses this option to determine whether layout maps that are only available on the server should be automatically read from the server or downloaded into the workspace.

When you select an option other than **Ignore** from this drop-down list box, NaturalONE tries to establish a connection to the Natural server whenever the parser detects a missing object (using the connection properties stored in the associated Natural project). When the connection can be established, any missing object that can be found on the Natural server is either loaded into memory or downloaded into the workspace, depending on the option you have selected (see below). The **steplib** information defined for the current project or library is used for locating the missing objects.

This drop-down list box provides the following options:

- **Ignore**
  Default. No action is taken. In this case, you can manually download a missing object using the **Open** command in the **Dependencies** view (see *Dependencies View* in *Using the Source Editor*).
- **Read from server**
  The object is loaded into memory and can thus be found by the parser and in the **Dependencies** view. The object remains in memory as long as you do not close the source which references this object. When you close the source, the object is removed from memory.

  Note: By selecting this option, the parser error markers are removed, however, you will not be able to use the loaded object in the same way as a downloaded object. For example, the object cannot be used if you want to import data fields.

- **Download from server**
  The object is physically downloaded into the workspace. It is placed into the current project, into a library which has the same name as the library on the Natural server.

  Exceptions:

  - If Natural Security does not allow downloading the object physically, the **Download from server** option is handled as **Read from server**.
  - When you are editing objects directly on a Natural server, missing objects are never downloaded into the temporary project which is automatically created in your workspace. They can only be loaded into memory. Therefore, the **Download from server** option is handled as **Read from server**.

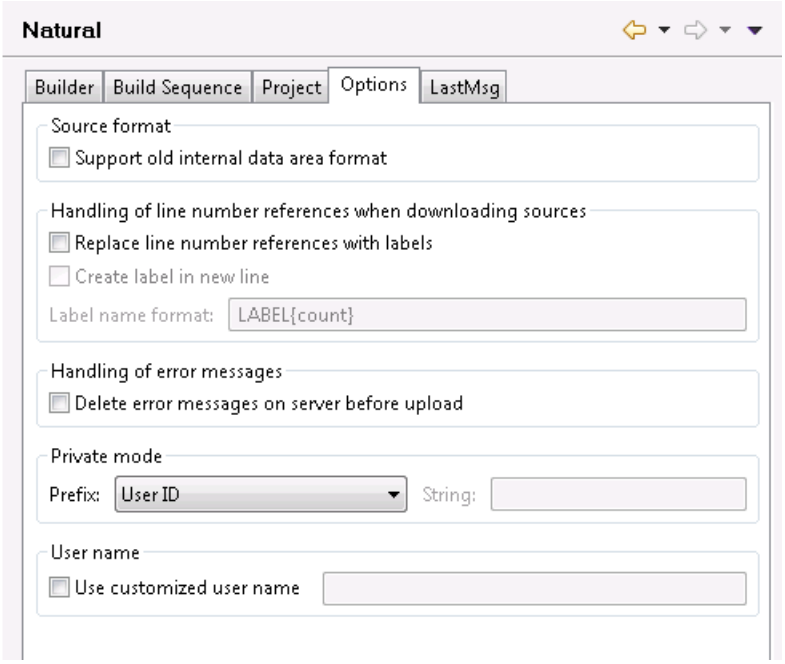
Properties

The **Update project properties automatically on .natural file modification** option controls whether or not an automatic update of the Natural project properties is made on a *.natural* file modification.

When selected, an automatic update of the Natural project properties is made when the content of the *.natural* file was changed externally, typically when a repository update has been made on this file. A manual update of the project properties is no longer necessary.

When not selected, no automatic update of the Natural project properties is made. A manual update of the project properties must be performed using the function **Restore from Defaults** in the **Project > Properties > Natural** dialog.

Options



Support old internal data area format

The setting of this option is used as the default value in the **project properties** when you **create** a new Natural project (either by using the wizard or by downloading from a Natural server).

This option applies to the internal format of data areas in a Natural for Windows, UNIX and OpenVMS environment.

With Natural Version 6.1 for Windows and UNIX and Natural Version 6.3 for OpenVMS, a new internal format was introduced for data areas which supports, for example, dynamic and large variables.

When data areas are uploaded to the Natural environment, the new internal data area format is used by default. It is strongly recommended that you keep this default (that is, do not select this check box).

Data areas with the new format are not downward-compatible. Therefore, it is not possible to use them with Version 5.1 and below. Select this check box only, if you require data areas in the old format that are to be used with Natural Version 5.1 or below.

Replace line number references with labels

When selected, the line number references in the source code are permanently replaced with labels when sources are downloaded to the Eclipse workspace. When the sources are later uploaded to the Natural environment, the labels remain in the source code. See also ***Line Numbers***.

Create label in new line

Only available when **Replace line number references with labels** is selected.

When not selected (default), each label is inserted at the beginning of the line which is referred to by the line number reference.

When selected, each label is created in a new line, directly above the line which is referred to by the line number reference.

Label name format

Only available when **Replace line number references with labels** is selected.

By default, the label name format "LABEL{count}" is used, which results in the label names "LABEL1", "LABEL2", etc. You can specify any other valid label name containing the parameter "{count}". If you do not specify "{count}", the label count is nevertheless appended at the end of the label name.

Delete error messages on server before upload

The setting of this option is used as the default value in the **project properties** when you **create** a new Natural project (either by using the wizard or by downloading from a Natural server).

When selected, all error messages are deleted in the appropriate library on the server before the error messages from a project are uploaded.

When not selected (default), all error messages are uploaded to the server. Any error messages which are no longer available in the project are not deleted on the server. This may cause inconsistencies.

See also ***Creating Application-Specific Messages***.

Private mode

The setting of the **Prefix** drop-down list box determines the names of the private-mode user libraries and private-mode steplibs that are automatically created when private mode is active for a project (when Natural Security is not active) or library (when Natural Security is active). For further information on private mode, see ***Steplibs*** in *Changing the Project Properties*.

Each name consists of a prefix which is up to six characters long and two digits which are automatically incremented (from 01 to ZZ). By default, the first 6 characters of the user ID are used as the prefix. For example, when the user ID is "CHARLES" (7 characters), the name for the first library is "CHARLE01", the name for the second library is "CHARLE02", and so on. When a user ID has less than 6 characters, the library name is filled with additional zeros (for example, "JIM00001").

The **Prefix** drop-down list box provides the following options:

- **<customize>**
  You define your own prefix, which can be up to 6 characters long, in the **String** text box. Due to the Natural naming conventions for libraries, the prefix must start with an alphabetical character.
- **User ID**
  The first 6 characters of the user ID are used as the prefix. This is the default.
- **Project name**
  The first 6 characters of the project name, as defined in the **Navigator** view or **Natural Navigator** view, are used as the prefix. Embedded blanks are removed and lowercase characters are converted to uppercase.
- **Library name**
  The first 6 characters of the original library name are used as the prefix.

User name

The setting of **User name** determines the name which is to be used as user ID in new NaturalONE projects and new Natural objects.

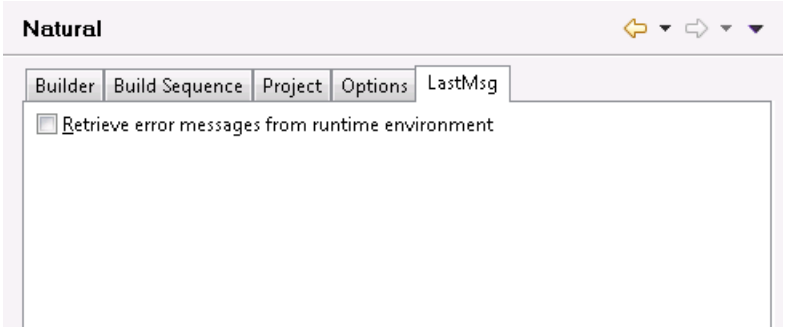
When **Use customized user name** is selected, the user name must not be blank and must be valid according to the rules for valid Natural library names. When **Use customized user name** is not selected, the user name is identical to the login user name of the operating system.

The user name is taken:

- as user ID in new Natural projects created using the New Project Wizard.
- as user ID in projects checked out from the repository.
- as user ID in imported projects.
- as author in objects created using the New Object Wizard.

Setting or modifying the user name has no effect for already existing Natural projects and Natural objects.

LastMsg



NaturalONE uses the **LastMsg** view to display additional information about the error situation which has occurred last. This functionality can be activated or deactivated by setting the preference **Retrieve error messages from runtime environment**.

Any time a Natural command is issued, the potentially created error messages in the runtime environment are transferred to NaturalONE and displayed inside this view; for further information, see *Using the LastMsg View*.

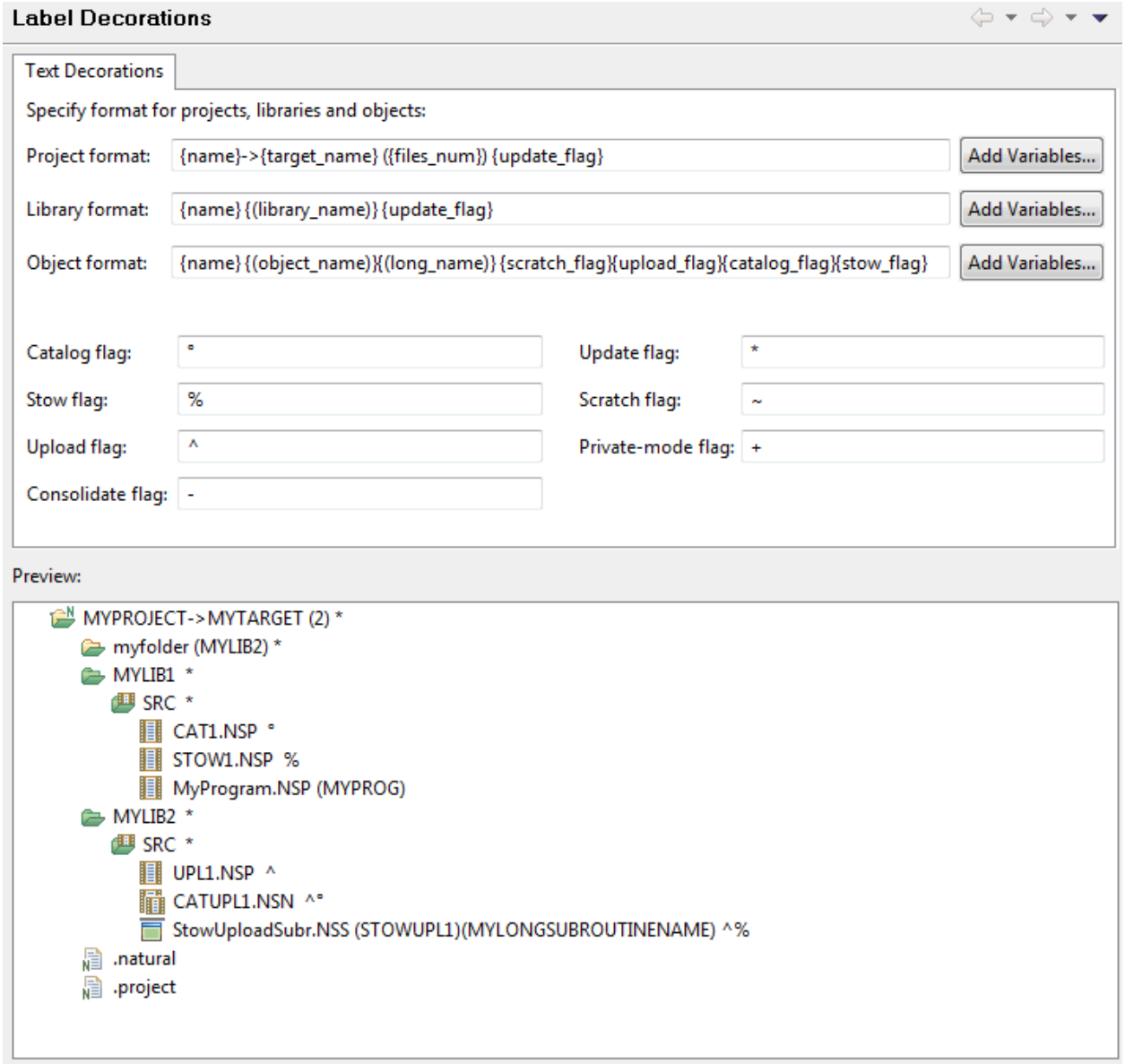
Appearance

This page does not contain any options. When you expand the corresponding node in the tree, you can see the **Label Decorations** subnode (see the description below).

Label Decorations

This page is available when you expand the **Appearance** node in the tree.

You can specify label decorations for the project nodes, library nodes, library subnodes and objects that are shown in the **Navigator** view or in the **Natural Navigator** view. By default, specific variables are set for each of these elements.



If you want to use different variables, for example, for the object format, you can choose the corresponding **Add Variables** button. Before you choose this command button, make sure to place the cursor at the position where you want to insert the variables (do not place the cursor within an existing variable). In the resulting dialog box, you can then select the required variables (a brief description of each variable is provided in this dialog box).

The following table describes the variables which can be defined for the different formats (in alphabetical order):

Variable	Description	Format		
		Project	Library	Object
`catalog_flag`	Only shown in the label decoration when the object needs to be cataloged in the Natural environment. See also ***Flags in a Label Decoration***.			X
`code_page`	The code page that is defined for the Natural object.			X
`consolidate_flag`	Only shown in the label decoration when objects in a private-mode library are obsolete and need to be consolidated. See also ***Flags in a Label Decoration*** and ***Consolidating Objects in Private-Mode Libraries***.	X	X	X
`ddm_dbid_fnr`	The database ID (DBID) and file number (FNR) of a DDM. Only shown in the label decoration when the object is a DDM.			X
`files_num`	Number of Natural libraries in the project (project format) or number of Natural objects in the library (library format).	X	X	
`library_name`	Name of the associated Natural library. Only shown in the label decoration when the name is considered as a library folder name.		X	
`long_name`	Long name of the Natural object. Only shown in the label decoration when the object has a long name. **Important:** Do not confuse the long names of Natural objects (for example, of subroutines or DDMs) with the "long" file names which can optionally be defined for Natural objects. These are two different things.			X
`name`	Name of the current project, library or object. With the object format, this can be either the Natural object name or a file name. See also ***File Names***.	X	X	X
`object_name`	Natural object name. Only shown in the label decoration when a file name has been defined. See also ***File Names***.			X
`private_mode_flag`	Only shown in the label decoration when the object is available in a private-mode library. See also ***Flags in a Label Decoration***.	X	X	X
`private_name`	Name of the associated private-mode library on the Natural server. Only shown in the label decoration when a private-mode name exists. See also ***Steplibs*** in *Changing the Project Properties*.		X	
`size`	Size of the Natural object in bytes as it will be stored on the Natural server (that is, without the source header). This is different from the object properties where the source header is included in the number of bytes.			X
`SM_mode`	Programming mode of the Natural object. This can either be "S" for structured mode or "R" for reporting mode.			X

Variable	Description	Format		
		Project	Library	Object
`scratch_flag`	Only shown in the label decoration when the objects needs to be scratched in the Natural environment. See also ***Flags in a Label Decoration***.			X
`src_lines`	Number of source code lines that will be stored on the Natural server (that is, without the source header). This is different from the source editor. When line numbers are shown, the source header lines are also counted.			X
`stow_flag`	Only shown in the label decoration when the object needs to be stowed in the Natural environment. See also ***Flags in a Label Decoration***.			X
`target_name`	Name of the assigned Natural environment. For a Natural server, host name and port number are shown. For a local Natural runtime, the string "natural-runtime" is shown.	X		
`type`	Type of the Natural object (such as "PROGRAM" or "DDM").			X
`update_flag`	Only shown in the label decoration when objects need to be updated in the Natural environment. See also ***Flags in a Label Decoration***.	X	X	
`upload_flag`	Only shown in the label decoration when the object needs to be uploaded to the Natural environment. See also ***Flags in a Label Decoration***.			X

In the text boxes for the different formats, you can enter any characters (for example, parentheses, minus signs or blanks) or even words between the variables in order to improve the readability in the **Navigator** view or **Natural Navigator** view. The preview area always shows how the current definition would be rendered in the **Navigator** view.

When a specific value is to be shown in parentheses, you simply define it as shown in the following example:

```
({files_num})
```

However, if a value is not always available (for example, the long name of a Natural object), you can also use the following syntax in order to avoid empty parentheses in the **Navigator** view or **Natural Navigator** view (you can use this syntax with all variables and all types of brackets):

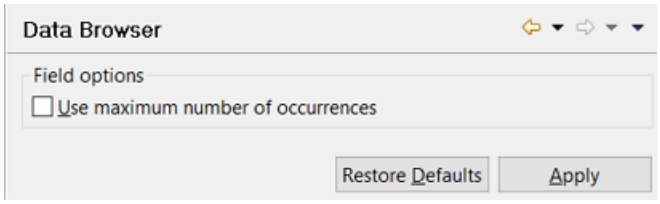
```
{(long_name)}
```

The representation of the different flags (catalog, stow, upload, update, scratch, consolidate, private mode) depends on the characters that are defined on this page. These flags are shown in the **Navigator** view or in the **Natural Navigator** view. See the flag descriptions in *Flags in a Label Decoration*. If you want, you can define different flags to be shown in the **Navigator** view.

Note: The Natural-specific label decorations that are shown in the **Navigator** view or in the **Natural Navigator** view are controlled by the preferences under **General > Appearance > Label Decorations**. By default, all Natural-specific label decorations are shown. If you do not want to have label decorations, just go to the above mentioned preference page and deselect **Natural Objects**.

Data Browser

The following settings can be made for the data browser in the Natural preferences:



Use maximum number of occurrences

The number of occurrences for multiple fields and periodic groups given in the Predict file for the Adabas DDM are used. If no Predict file is accessible, always the maximum number of occurrences is used. The maximum number of occurrences is 191 for a multiple field and 91 for a periodic group.

See also *Displaying the Properties for a Field* in *Using the Data Browser*.

Debug

This page does not contain any options. When you expand the corresponding node in the tree, you can see the **Debug Attach Settings** and **Display Options** subnodes (see the descriptions below).

Debug Attach Settings

The debug attach settings that are defined in the Natural preferences are used when you debug Natural RPC applications, Natural for Ajax workplace applications or external Natural applications. See also *Using a Debug Attach Server*.

Use debug attach server
When selected, the debug attach server is used.

Host name
Only available when the **Use debug attach server** check box is selected.

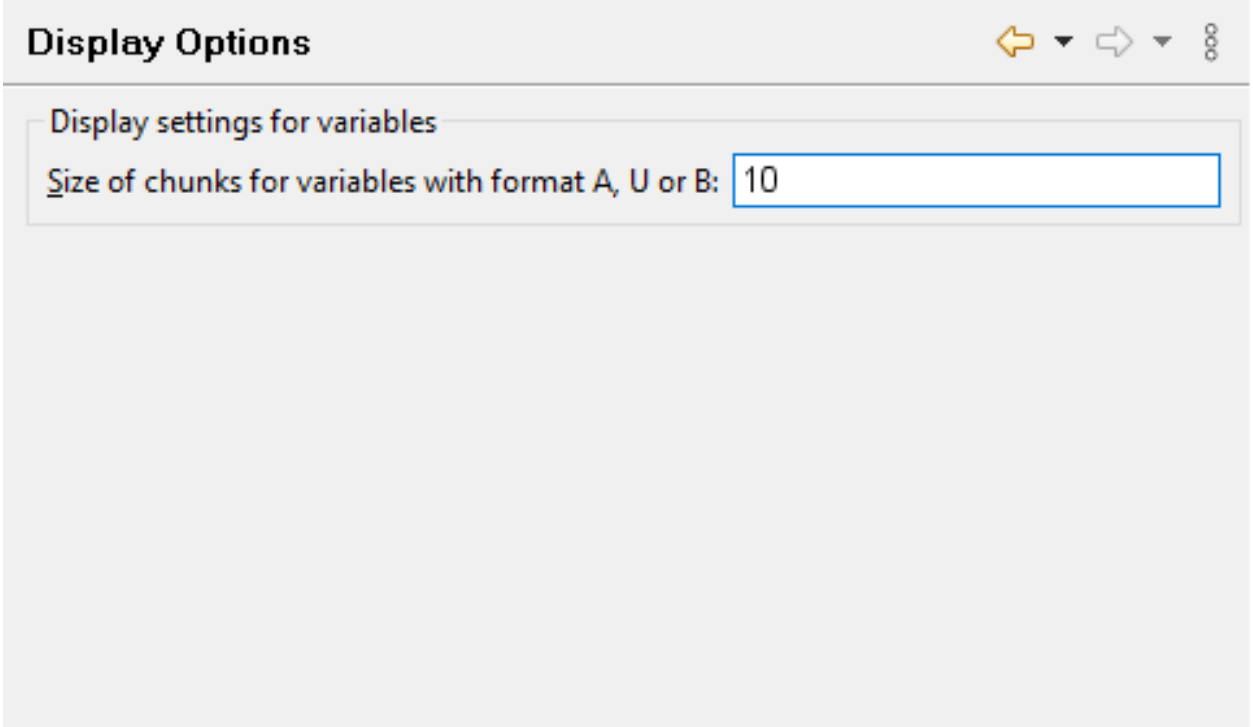
The name of the host (or IP address) on which the debug attach server is running. The name of your local server is automatically provided. Do not use "localhost" as the host name.

Port number
Only available when the **Use debug attach server** check box is selected.

The number of the port to which the debug attach server is listening. The port number used on your local server is automatically provided. The default port is 2500.

Display Options

The display settings for variables. See also the description of the **Variables** view.



Size of chunks for variables with format A, U or B

Defines the size of the chunks which are displayed in the Debug perspective's **Variables** view for variables of format A, U or B format. The default chunk size is 10. The value has to be in the range from 1 - 253.

If the chunk size preference is changed during debugging, the new chunk size will be used after the next debugger **Step** or **Resume** action.

Editors

This page does not contain any options. When you expand the corresponding node in the tree, you can see subnodes for the different types of editors (see the descriptions below).

DDM Editor

This page is available when you expand the **Editors** node in the tree.

You can set preferences for various DDM editor options. These settings are taken as default values each time you start the DDM editor. See ***Using the DDM Editor*** for further information.

You can set the following options:

Insert location

You can specify where fields are to be inserted. Select one of the following option buttons:

- **Insert before**
  When selected, a field is always inserted before the currently selected field.
- **Insert after**
  When selected, a field is always inserted after the currently selected field.

Best fit

When selected, the width of each visible column is adjusted so that the column header and the content of a column are always completely visible.

Auto fit

Only available when **Best fit** is selected.

When selected, each edited column is automatically readjusted to the optimum width (as described above) when you leave the column.

When deselected, the width of the edited column is not readjusted when you leave the column.

Map Editor

This page is available when you expand the **Editors** node in the tree.

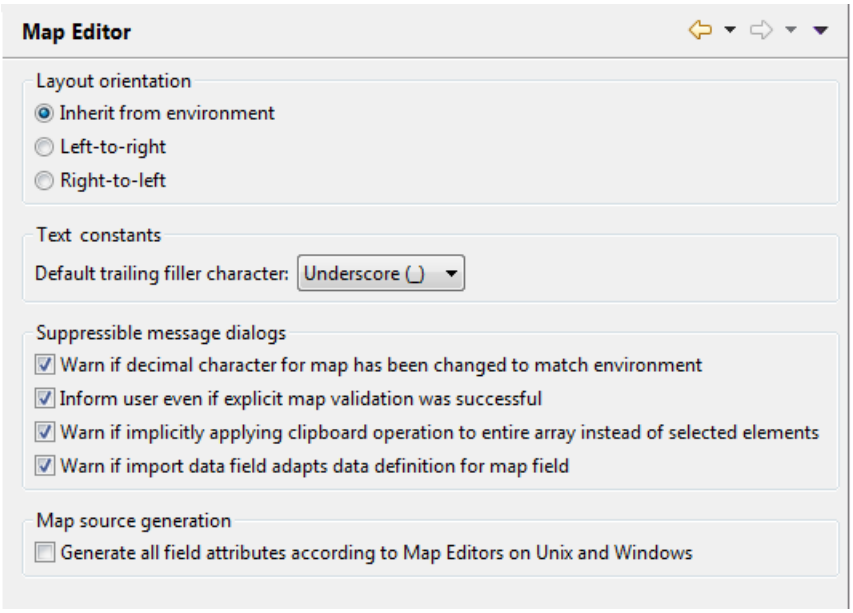
You can set the preferences for various map editor options. These settings are taken as default values each time you start the map editor. See ***Using the Map Editor*** for further information.

General preference settings and a subnode are provided for the map editor:

- General Map Editor Settings

- Coloring

General Map Editor Settings



You can set the following options:

Layout orientation

Here you can specify the default display direction. For further information refer to ***Editor*** in section *Changing the Project Properties*.

Text constants

Here you can set the default trailing filler character for text constant labels in new maps. The value entered here is used to initialize the Label Filler property for the map. See ***Changing the Properties for the Map*** in the section *Using the Map Editor*.

Suppressible message dialogs

You can suppress some message dialogs in the map editor. The following options determine whether or not the corresponding message dialogs will be shown. These message dialogs can also be directly suppressed via the **Do not show this message in the future** check box within the message dialog itself. When you select this check box in the message dialog, the corresponding option is automatically deselected in the preferences.

- **Warn if decimal character for map has been changed to match environment**
  When selected, a message dialog appears after the decimal character used by the map has been changed to match the value used by the corresponding project or (secured) library.

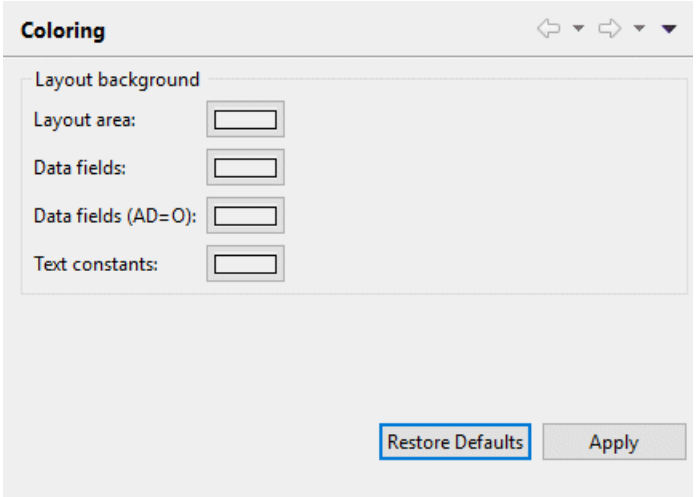
- **Inform user even if explicit map validation was successful**
  When selected, a message dialog appears after performing an explicit validation of the map. See ***Validating Maps*** in the section *Using the Map Editor* for more information.
- **Warn if implicitly applying clipboard operation to entire array instead of selected elements**
  When selected, a message dialog appears if a clipboard operation (**Cut**, **Copy**, **Delete**) is attempted on individual array elements. Because these operations are not supported for array elements, the operation is instead implicitly applied to the entire array in such cases. See also ***Selecting Controls in the Map*** in the section *Using the Map Editor*.
- **Warn if import data field adapts data definition for map field**
  When selected, a message dialog appears if the selected field from ***Importing Data Fields*** exists on the map and the data definition can be adapted. See also ***Adding Controls to the Map*** for more information.

Map source generation

You can influence the generation of a map source when you save the map in the Map Editor. This may be useful to achieve full run time compatibility when a map object is executed within an application on a specific operating system.

- **Generate all field attributes according to Map Editors on Unix and Windows**
  When selected, all field attributes which are defined for the map fields are generated. When deselected, only the field attributes which do not match the default setting are generated. This is compatible to the Map Editor on mainframe systems.

Coloring



You can change the background colors for the different elements in the map editor layout page. The coloring for each element can be adapted individually.

To change the color for a specific syntax element

1 Click on the **Color** button of the map layout element you want to change and then choose the new color in the resulting dialog box.

   Color buttons are available for the following elements:

   - **Layout area**
   - **Data fields**
   - **Data fields (AD = 0)**
   - **Text constants**

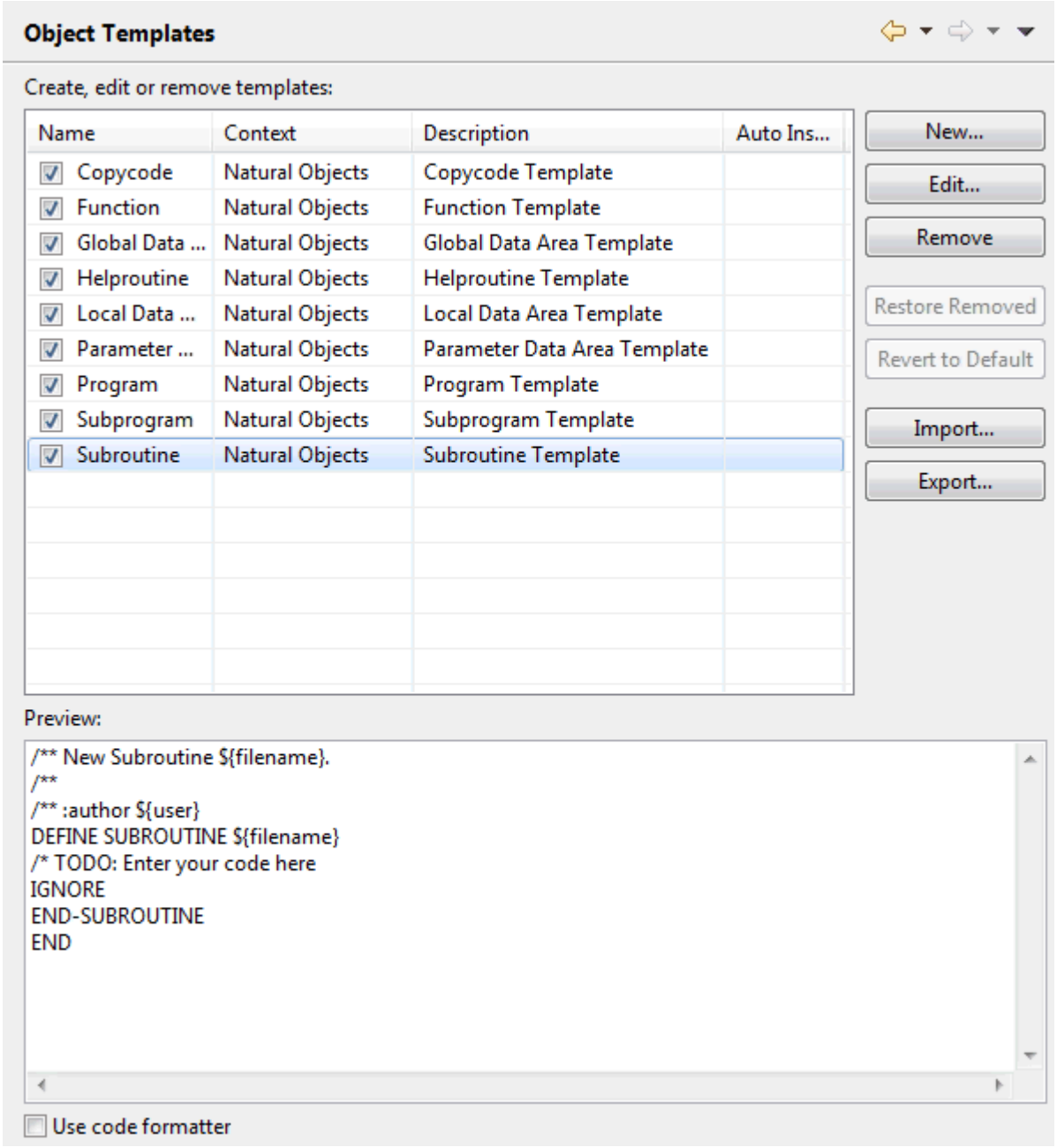
2 Click the **Apply** button in order to make your color selection permanent.

   As a result, the background of the corresponding map editor layout element is visualized with the set background color.

Object Templates

This page is available when you expand the **Editors** node in the tree.

When you create a new Natural object which uses the source editor, a skeleton which is typical for this type of object is automatically provided in the source editor. The skeletons for the different types of objects are defined using the templates on this preference page. You can edit the existing templates so that they meet your specific requirements, and you can also create new templates.



Note: The layout of this preference page is the same as that, for example, of the Java editor templates. For information on how to use this page, see the Eclipse online help.

For example, when you create a new subroutine with the name "MYSUB1", the following skeleton is automatically provided in the source editor:

```
/** New Subroutine MYSUB1.
/**
/** :author natural
DEFINE SUBROUTINE MYSUB1
/* TODO: Enter your code here
IGNORE
END-SUBROUTINE
END
```

The file name that is shown in the above example is controlled by the variable `${filename}` which is defined in the template. When you edit a template, you can define the following variables:

Variable Name	Description
`${date}`	Current date.
`${filename}`	Name of the current file.
`${time}`	Current time.
`${user}`	Name of the current user.
`${year}`	Current year.

Source Editor

This page is available when you expand the **Editors** node in the tree.

You can set preferences for various source editor options. These settings are taken as default values each time you start the source editor. See ***Using the Source Editor*** for further information.

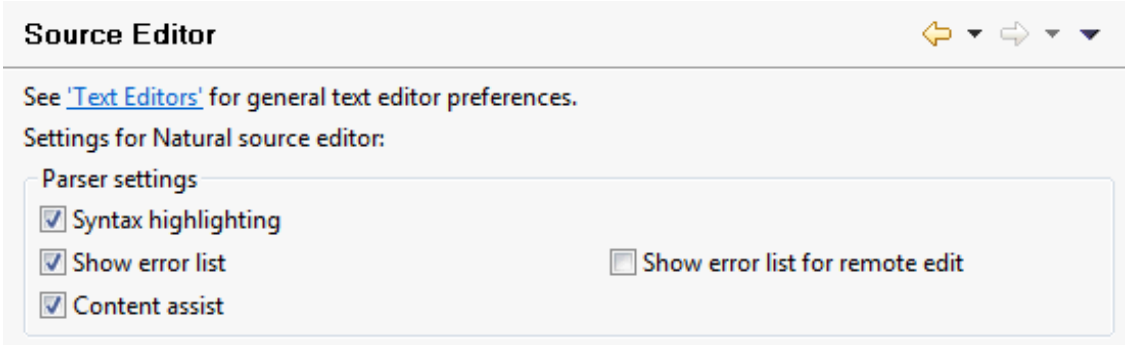
Several preference pages (subnodes) are provided for the source editor:

- Source Editor
- Case Translation
- Folding
- Mark Occurrences
- Protection
- Struct
- Syntax Coloring

- Templates

Source Editor

NaturalONE uses the general text editor preferences of Eclipse for its source editor, but it also provides special options for the source editor.



The following options are available:

Syntax highlighting
When selected, syntax highlighting is used in the source editor. See also *Syntax Coloring*.

Show error list
When selected, the errors in the source code are shown persistently in the **Problems** view after a program containing errors has been saved.

Show error list for remote edit
The **Natural Server** view provides direct access to the Natural objects stored on a server. When you edit a source directly on the server, a temporary project is created in your workspace and the source is downloaded into this project. After the download, the Natural parser performs a syntax check.

When this option is selected and external objects (such as data areas or copycodes) are defined in the downloaded source, many errors are displayed in the editor. The reason for this is that no data definitions are available to the parser.

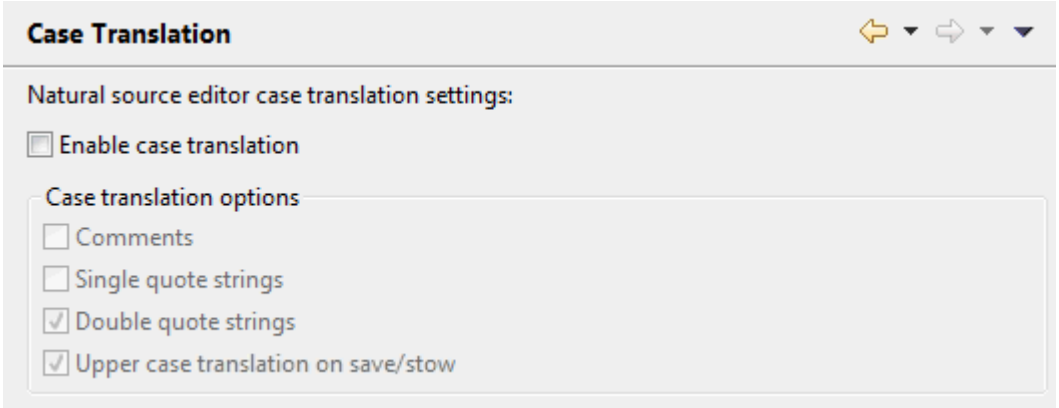
When this option is not selected, the misleading errors are not shown in the editor.

To benefit from the capabilities of the Natural parser, however, it is recommended that you create a Natural project in the local Eclipse workspace and then download the entire application into this project. Additional functionality such as versioning of the sources and deployment to the production environment is then available.

Content assist
When selected, content assist is available in the source editor.

Case Translation

In the source editor, you can translate source code from upper case to lower case or from lower case to upper case.



Enable case translation

When selected, the commands **Upper Case** and **Lower Case** are enabled in the source editor. See *Translating to Upper Case or Lower Case*.

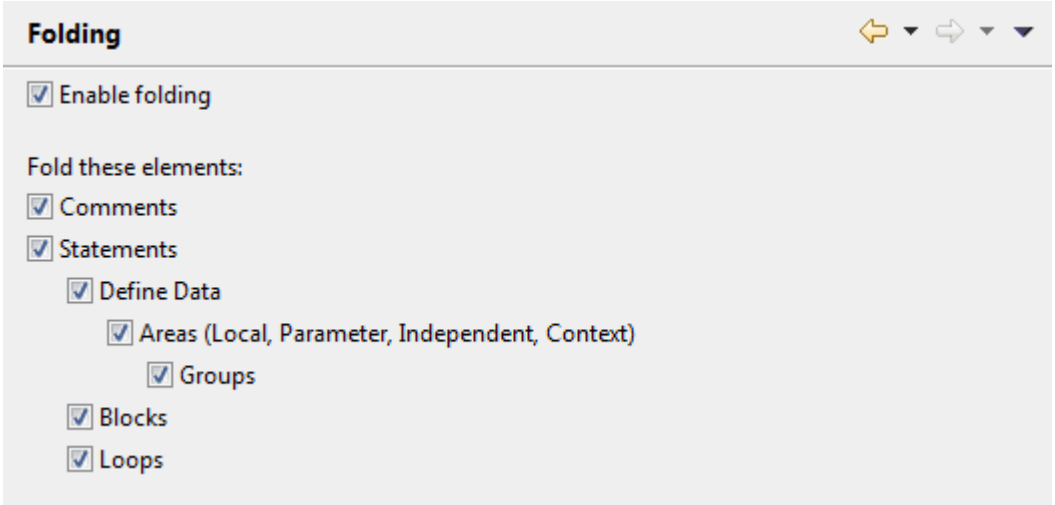
When selected, the remaining check boxes on this preference page are also enabled and you can specify the case translation options. When you then use one of the above mentioned commands in the source editor, the following applies:

- **Comments**
  When selected, the case for strings within a comment is changed.
- **Single quote strings**
  When selected, the case for strings within single quotes is changed.
- **Double quote strings**
  When selected, the case for strings within double quotes is changed.
- **Upper case translation on save/stow**
  When selected, the entire source code is automatically translated to upper case whenever you save or stow the source code, depending on the settings of the above options.

  This is useful with Natural servers on mainframes where the `LOWSRCE` (allow lower-case source) compilation option is set to `OFF`. For details, see the description of the system command `COMPOPT` in the Natural for Mainframes documentation.

Folding

Folding means that different elements in the source editor can be collapsed and expanded. This improves the readability and maintainability of objects with complex code structures.



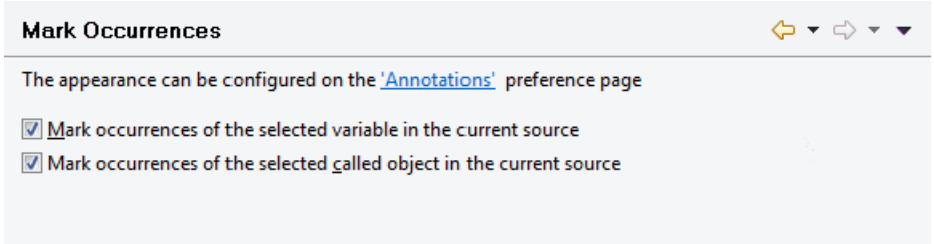
Enable folding

When selected, folding is enabled in the source editor. By selecting the remaining check boxes on this preference page, you can then specify the elements that are to be folded.

Elements that can be folded are, for example, `DEFINE DATA` blocks, `REPEAT` blocks, `IF THEN ELSE` blocks, `READ` blocks, and blocks of two or more consecutive comment lines.

Mark Occurrences

In the source editor, it is possible to mark all occurrences of a selected variable.



Mark occurrences of the selected variable in the current source

When selected, all references to the selected variable are marked in the current source.

The marker may be, for example, text which is highlighted with a specific color. This depends on the settings on the **Annotations** page in the general text editor preferences of Eclipse. The corresponding annotation type is **Occurrences**.

Note: The annotation type **Write Occurrences** is currently not supported.

Mark occurrences of the selected called object in the in the current source

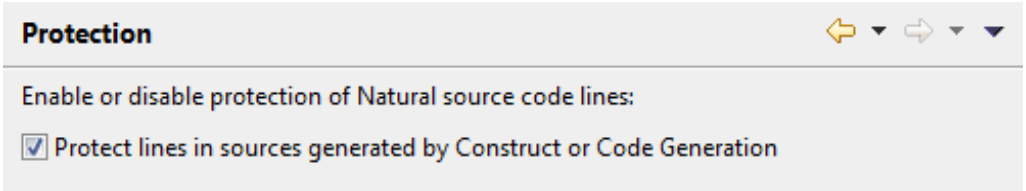
When selected, all references to a selected called object are marked in the current source.

Called objects are subprograms, subroutines, functions, adapters, maps, helproutines and programs.

If you open an object from the passive cross-references of the *Dependencies view*, the cursor is positioned to the first occurrence of the corresponding called object and all references are marked.

Protection

In the source editor, you can enable or disable the protection of source code lines.



Protect lines in sources generated by Construct or Code Generation

When selected (default), the protected lines in sources generated by Construct or by the Code Generation component of NaturalONE cannot be edited. These are the sources where the first line of the code starts with `**SAG GENERATOR`.

When deselected, all lines in the generated sources can be edited. You should disable the protection only if you no longer intend to regenerate the sources, but intend to maintain and further develop the code using the source editor of NaturalONE instead.

See also *Protected Lines in Sources Generated by Construct or Code Generation*.

Struct

When you use the **Struct** command in the source editor, the source code lines are indented according to the settings that are defined on this page. See *Indenting the Source Code Lines*.

Indentation size

You can enter the number of positions (from 1 to 9) by which source code lines are to be indented. By default, indentation is by 2 positions.

Indentation/alignment of comments

This drop-down list box provides the following options:

- **Indent comment lines**
  Each comment line will be indented as far as the statement line above it; except comment lines which begin at the beginning of a line, these will be not be indented.
- **Do not indent comment lines**
  Comment lines will not be indented.
- **Align comment lines left-justified**
  Comment lines will be aligned left-justified.

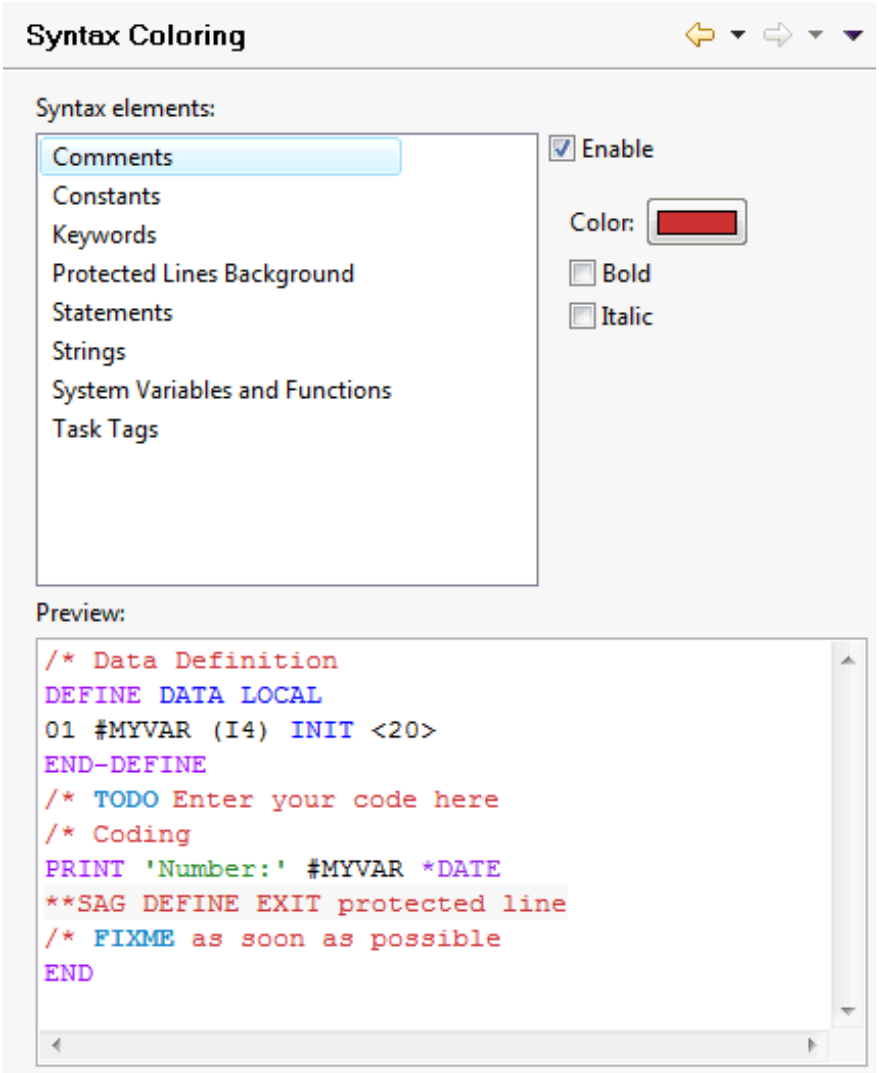
Maximum line length

You can enter a number which defines the maximum line length. A line can be up to 245 characters long. Default: 72.

Syntax Coloring

You can change the colors for the different syntax elements in the source editor. The current colors are shown in the preview area at the bottom of the page. The coloring for each syntax element can be enabled/disabled individually.

Note: To see the colors in the source editor, syntax highlighting must be enabled on the **Source Editor** page.



To change the color for a specific syntax element

1 Select the syntax element in the list box at the top.

2 Make sure that the **Enable** check box is selected for this syntax element.

The color that is currently used for this syntax element is shown on the **Color** button.

3 Choose the **Color** button, and then choose the new color in the resulting dialog box.

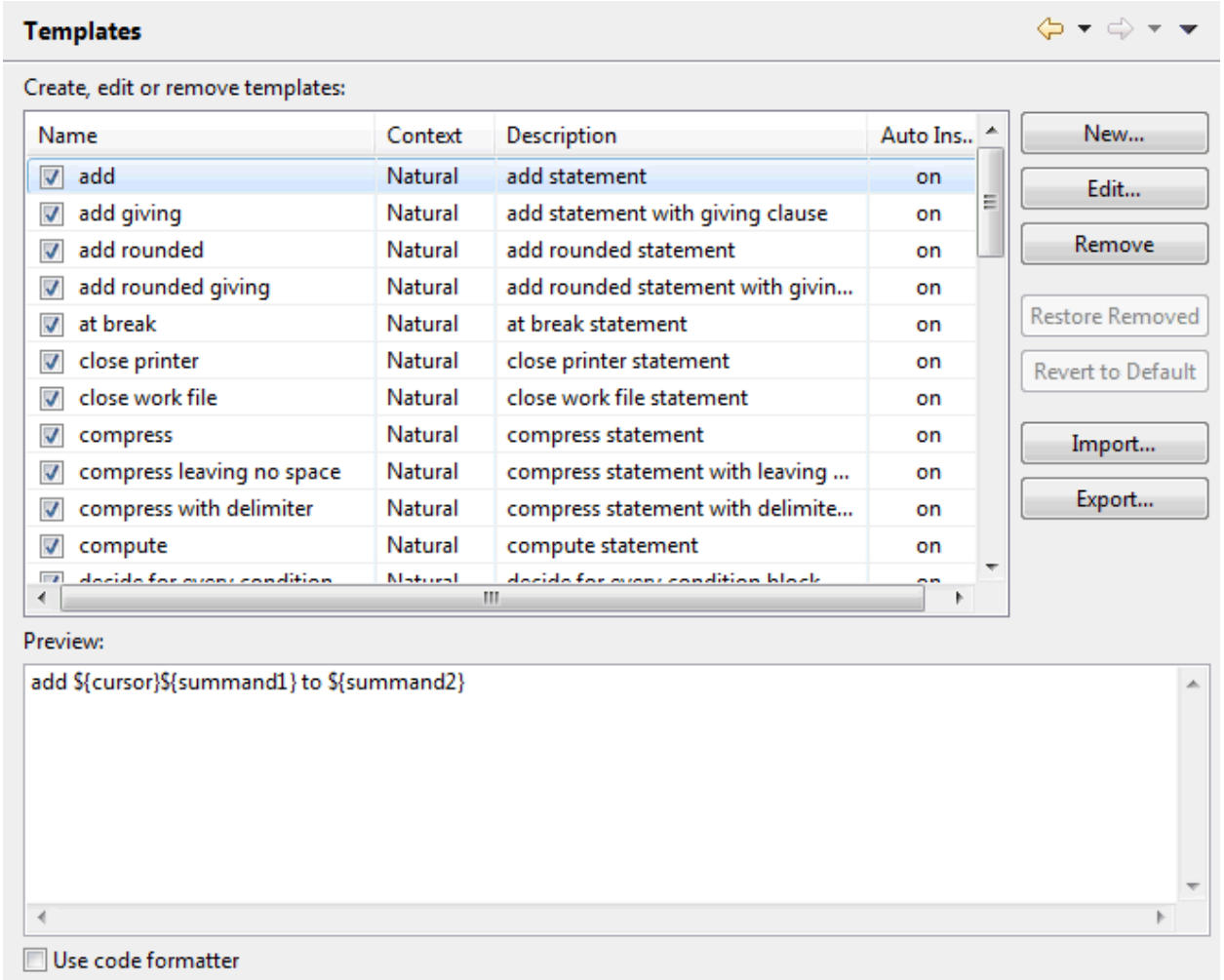
If you want to define bold and/or italic text for the selected syntax element, select the corresponding check box(es).

Each change is shown in the preview area at the bottom of the page.

Templates

Templates are used to insert recurring coding patterns into your source code. They are inserted using content assist, which is a standard feature in Eclipse. See also *Using Content Assist*.

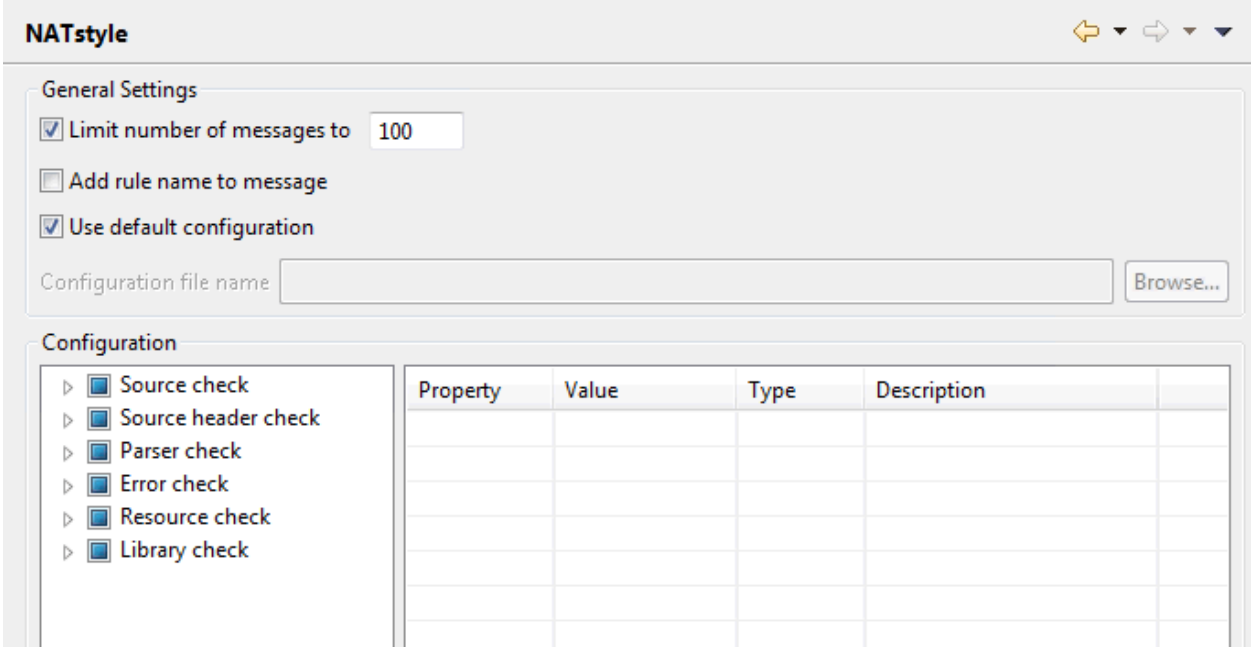
A number of templates is automatically provided with NaturalONE. You can edit them so that they meet your specific requirements, and you can also create new templates. The name of a template must always start with the name of a Natural statement; otherwise the template is not offered for selection by the content assist feature.



Note: The layout for this preference page is the same as that, for example, of the Java editor templates. For information on how to use this page, see the Eclipse online help.

NATstyle

The NATstyle settings that are defined in the Natural preferences are used when you check your Natural code with NATstyle. See also *Checking Natural Code with NATstyle*.



Limit number of messages to

When selected, the number of entries in the **Problems** view and the number of NATstyle markers in the Natural editors is limited to the number you specify in the text box.

Add rule name to message

When selected, the name of the rule is added to the end of the message. It is shown in brackets. For example:

```
TODO: Enter your code here [TODO comment]
```

Use default configuration

When selected, the default configuration is used for checking the Natural code. The default configuration cannot be modified.

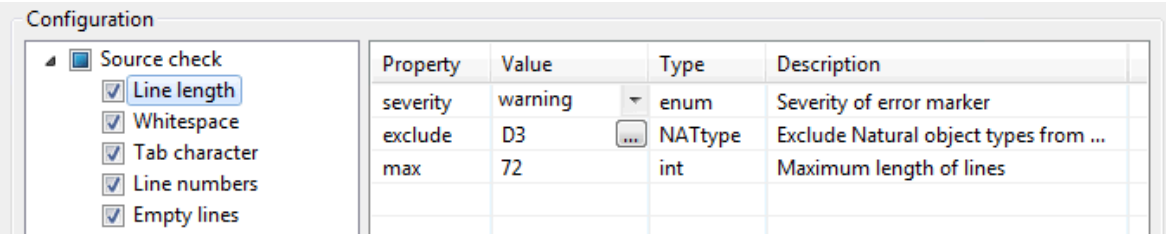
When not selected, you can specify your own configuration file (see below). In this case, the **Value** column in the **Configuration** group box can be edited.

Configuration file name

Only available when the **Use default configuration** check box is not selected. When you use the **Browse** button to select an existing configuration file, this file is loaded and the settings defined in this file are shown in the **Configuration** group box.

To create a new configuration file, proceed as follows:

1. In the **Configuration** group box, expand the tree on the left.
2. In the tree, select the rule that you want to modify (for example, select **Line length** which is visible when you expand **Source check**).



3. Specify the appropriate information in the **Value** column. Different properties can be set for a rule. Depending on the property, you can specify the value using a drop-down list box, a dialog box or you simply type the value in a text box. See also the description of the property types below.
4. In the tree, make sure to select the check boxes for all rules that are to be used.
5. Choose the **Apply** button.

   When a configuration file has not yet been defined, a dialog box appears, asking whether you want to save your changes to a new configuration file.
6. Choose the **Yes** button.

   A dialog box appears, providing the file name *NATstyle.xml* as a proposal.
7. Specify a file name and choose the **Save** button.

   The path and name of this file is now shown in the **Configuration file name** text box.

It is possible to export the NATstyle preferences from the currently used configuration file using the standard Eclipse functionality (**File > Export > General > Preferences**). This is helpful, for example, if you want to import these settings later into a different Eclipse workspace. A **NaturalONE NATstyle Preferences** entry is then available on the **Export Preferences** page. This entry is not available if the default configuration is currently used.

If you want to return to the default configuration, either choose the **Restore Defaults** button or select the **Use default configuration** check box. In both cases, a dialog box appears, asking whether you want to reload the default configuration. Any changes to your own configuration file which have not yet been applied will be lost.

Configuration

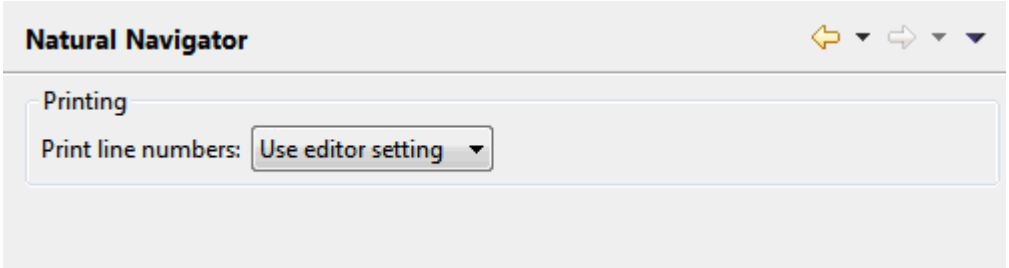
For detailed information on each rule that is shown in this group box, see ***Overview of NATstyle Rules, Error Messages and Solutions*** in the section *Checking Natural Code with NATstyle*.

Each property in a rule can belong to one of the following types:

Property Type	Value	Modification via
String	A standard Java string.	Text box.
String[]	An array of standard Java strings.	Dialog box in which you can add, modify, delete or clear string values.
int	A standard Java integer value.	Text box.
long	A standard Java long value.	Text box.
enum	One of a number of values.	Drop-down list box.
boolean	Either "true" or "false".	Drop-down list box.
regex	A regular expression.	Dialog box in which you enter a regular expression and a test string.
NATtype	An unsorted list of identifier characters for the Natural object types.	Dialog box in which you select the Natural object types that are to be checked.

Natural Navigator

The Natural Navigator settings that are defined in the Natural preferences are used when you work in the **Natural Navigator** view. See also *Using the Natural Navigator View*.



Print line numbers

Determines whether the printout of an object source contains line numbers when it is printed from the **Natural Navigator** view. See also *Printing Objects*.

This drop-down list box provides the following options:

- **Use editor setting**
  Default. The current setting of the **Show line numbers** option of the general Eclipse preferences is used (**Preferences > General > Editors > Text Editors**).
- **Yes**
  If selected, line numbers are always printed. The setting of the **Show line numbers** option of the general Eclipse preferences is ignored in this case.

- **No**
  If selected, line numbers are never printed. The setting of the **Show line numbers** option of the general Eclipse preferences is ignored in this case.

Parser

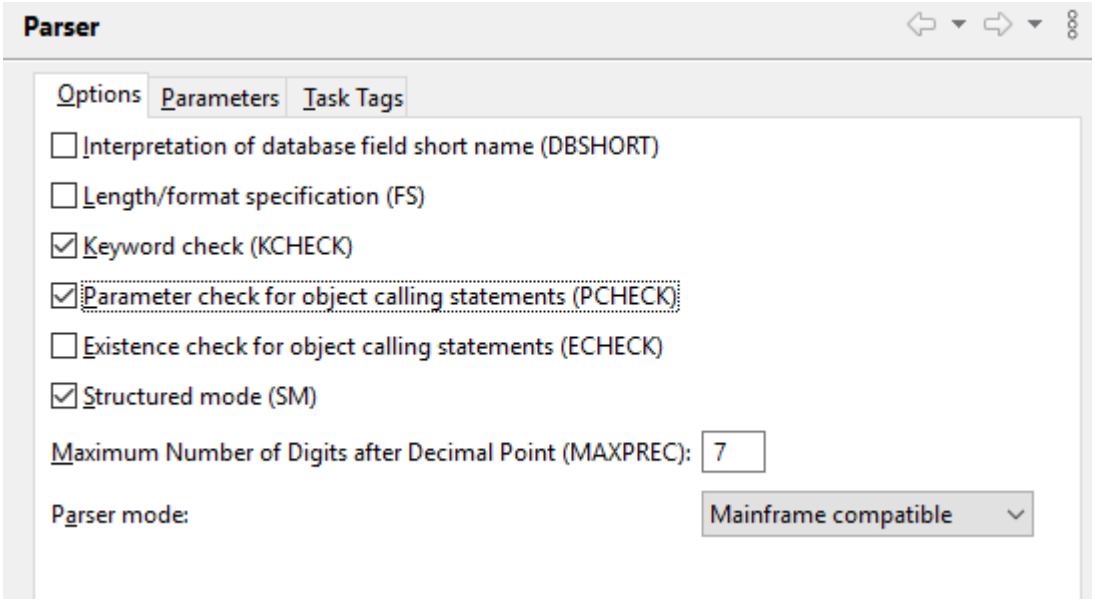
The parser settings (Natural profile parameters) that are defined in the Natural preferences are used as the default values when you create a new Natural project from scratch. See also ***Creating a New Project Using a Wizard***.

The following tabs are provided:

- Options
- Parameters

- Task Tags

Options



On this tab, you can specify the default values for the following Natural profile parameters:

Option	Corresponding Natural Profile Parameter
Interpretation of database field short names	DBSHORT
Length/format specification	FS
Keyword check	KCHECK
Parameter check for object calling statements	PCHECK
Existence check for object calling statements	ECHECK
Structured mode	SM
Maximum Number of Digits after Decimal Point	MAXPREC

For detailed information on these profile parameters, see the Natural documentation for the appropriate platform.

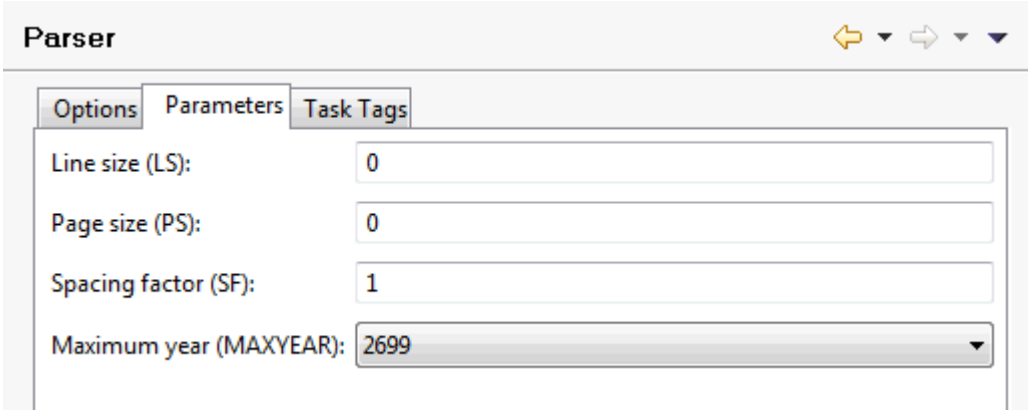
The **Parser mode** drop-down list box, which is also provided on this tab, allows you to define the platform for which the Natural language syntax is to be checked. You can select one of the following options:

- **Mainframe compatible**
  Natural sources are checked according to the Natural syntax for mainframe platforms.
- **Open systems compatible**
  Natural sources are checked according to the Natural syntax for Windows, UNIX and OpenVMS platforms.
- **Error tolerant**
  When the Natural sources are checked, all valid Natural syntax is accepted, no matter for which platform you are currently developing. An error will occur only if invalid Natural syntax is found which does not apply to any of the supported platforms.

  For example, if you are developing for UNIX platforms and your code contains special syntax which is only valid for mainframe platforms (such as the SQL extended set for DB2 databases), the parser will *not* consider this as an error.
- **Platform compatible**
  When the Natural sources are checked, not all Natural syntax is accepted. The parser only accepts syntax which can be used on *all* supported platforms. An error will occur if platform-specific syntax is found (even if this is the correct syntax for a specific platform).

  For example, if your code contains special syntax which is only valid for mainframe platforms (such as the SQL extended set for DB2 databases), the parser will consider this as an error.

Parameters

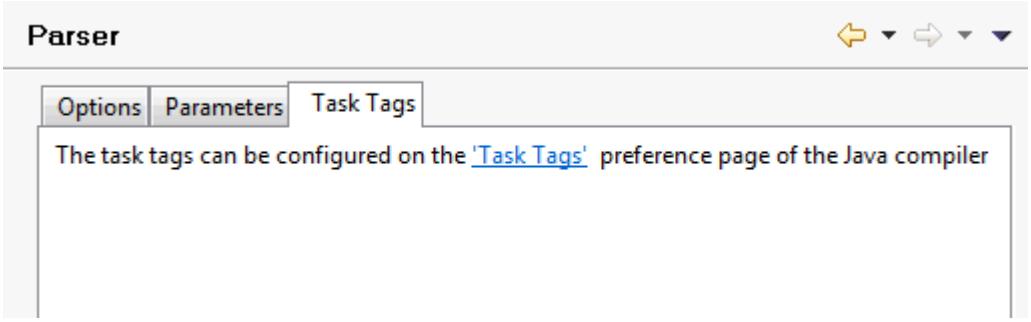


On this tab, you can specify the default values for the following Natural profile parameters:

Option	Corresponding Natural Profile Parameter
Line size	LS
Page size	PS
Spacing factor	SF
Maximum year	MAXYEAR

For detailed information on these profile parameters, see the Natural documentation for the appropriate platform.

Task Tags



A task tag can be used to mark a section in a Natural source that needs further refining. See also *Working with Tasks*. NaturalONE uses the task tags that are defined for the Java compiler. By default, these are `TODO` and `FIXME`. To go to the corresponding Java preference page, you can simply click the link on this tab. For more information, see the Eclipse online help.

Note: Other than the Java compiler, NaturalONE does not support project-specific configurations.

Profiler

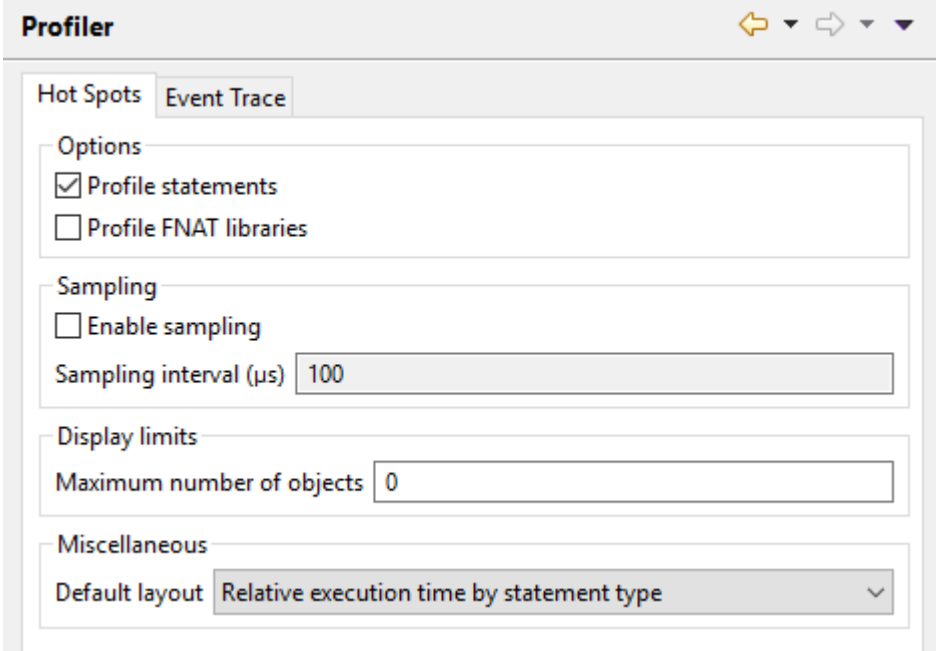
The profiler settings that are defined in the Natural preferences are used as the defaults when you start a profiler session. See also *Using the Natural Profiler*.

The following tabs are provided:

- Hot Spots
- Event Trace

Hot Spots

On this tab, you specify the defaults for the output that is to be shown on the **Hot Spots** page of the profiler.



Profile statements

When selected (default), the **Hot Spots** page of the profiler provides information on the used Natural statements. In this case, expandable nodes are shown for all involved Natural objects.

When not selected, only the information on the involved Natural objects is shown. Expandable nodes are not available.

Profile FNAT libraries

When selected, both system library (FNAT) and user library (FUSER) objects are profiled. For released versions of the system libraries, statement information is not available.

When not selected (default), only user library objects are profiled.

Enable sampling

When selected, a sampling technique is used which reduces the number of events sent by the trace session.

When sampling is enabled, the hit count corresponds to the number of recorded events (that is, the number of events that survive the sampling filter) and is not proportional to the number of times the event occurred (for example, the number of times a particular statement was executed). Instead, the hit count is weighted towards the events that take longer to execute, because they have a correspondingly greater chance of crossing a sampling interval. However, the displayed execution times (in particular the CPU times) are more-or-less independent of the sampling interval (except for statistical averaging effects, which increase with increasing sampling intervals and which may become significant). The higher the sampling interval, the less precise the results, but the lower the amount of generated and transferred event data (and thus the better the performance).

Sampling interval

Only enabled when the **Enable sampling** check box is selected.

Specify the CPU time interval in microseconds to be used for sampling. Only the events that cross this sampling boundary are recorded.

Natural servers on Windows and UNIX platforms only support a limited set of discrete values, and may therefore implicitly round the entered value up or down to the next supported value. The set of supported sampling intervals is: 1, 2, 4, 5, 8, 10, 16, 20, 25, 40, 50, 80, 100, 125, 200, 250, 400, 500, 625, 1000, 1250, 2000, 2500, 5000, 10000 . This is not relevant for the mainframe.

Maximum number of objects

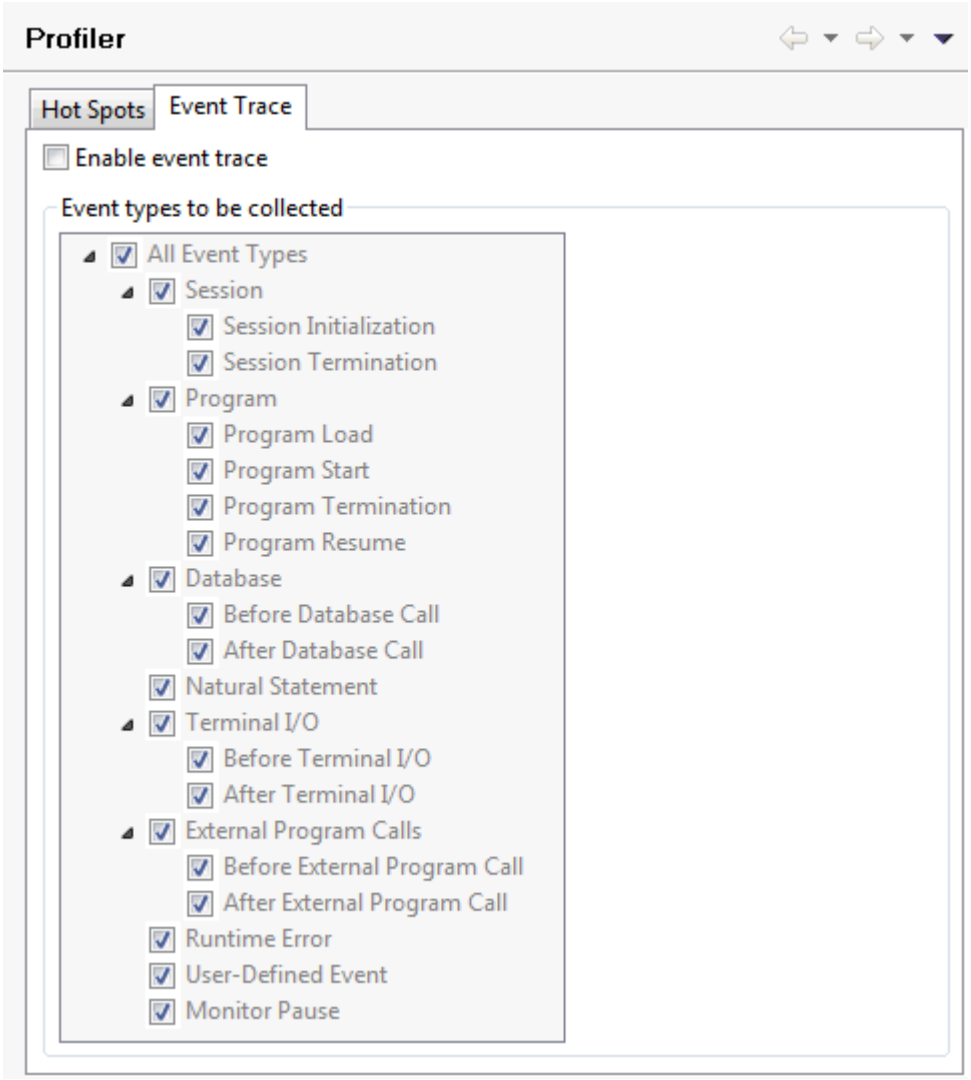
The maximum number of objects to be shown on the **Hot Spots** page of the profiler. The value 0 means that all objects are to be shown (default).

Default layout

The layout initially used for displaying the data on the **Hot Spots** page of the profiler. For further information, see **Viewing the Profiler Output > Hot Spots** in *Using the Natural Profiler*.

Event Trace

On this tab, you specify the defaults for the output that is to be shown on the **Event Trace** page of the profiler.



Enable event trace

If selected, trace records for the events are collected on the server, and the **Event Trace** page is shown in the profiler.

Note: Selecting this option can result in a big increase in the amount of data transferred to and stored on the client.

Event types to be collected

Only enabled when the **Enable event trace** check box is selected.

Deselect the event types for which you do not want to collect trace data. Or select the event types for which you want to collect trace data. See *Overview of Event Types* for a description of each event type.

Notes:

1. When the event type **Natural Statement** is selected, each executed Natural statement is traced, which may lead to a large amount of trace records.
2. If you intend to collect data for the event type **Natural Statement** and you are using a Natural server in a UNIX or Windows environment, the programs to be traced must have been cataloged with `GPGEN=(PROFILER=ON)`.

Regional Settings

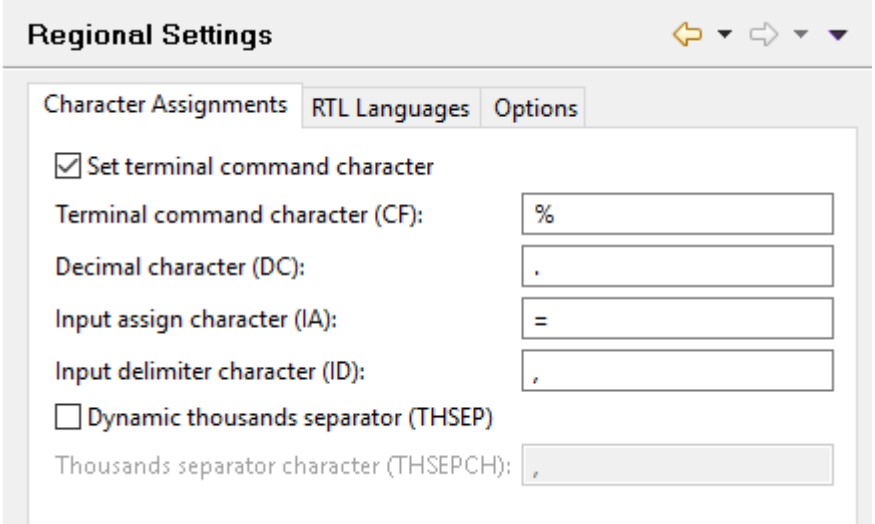
The regional settings (Natural profile parameters) that are defined in the Natural preferences are used as the default values when you create a new Natural project from scratch. See also ***Creating a New Project Using a Wizard***.

The following tabs are provided:

- Character Assignments
- RTL Languages

- Options

Character Assignments



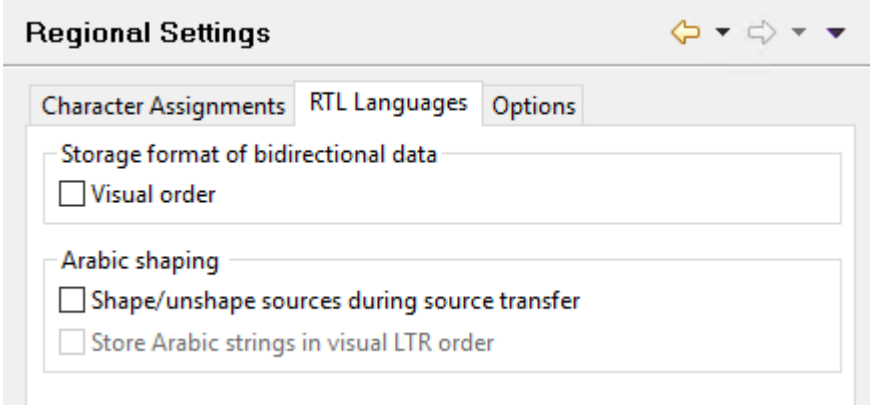
On this tab, you can specify the default values for the following Natural profile parameters:

Option	Corresponding Natural Profile Parameter
Set terminal command character	CF
Terminal command character	
Decimal character	DC
Input assign character	IA
Input delimiter character	ID
Dynamic thousands separator	THSEP
Thousands separator character	THSEPCH

For detailed information on these profile parameters, see the Natural documentation for the appropriate platform.

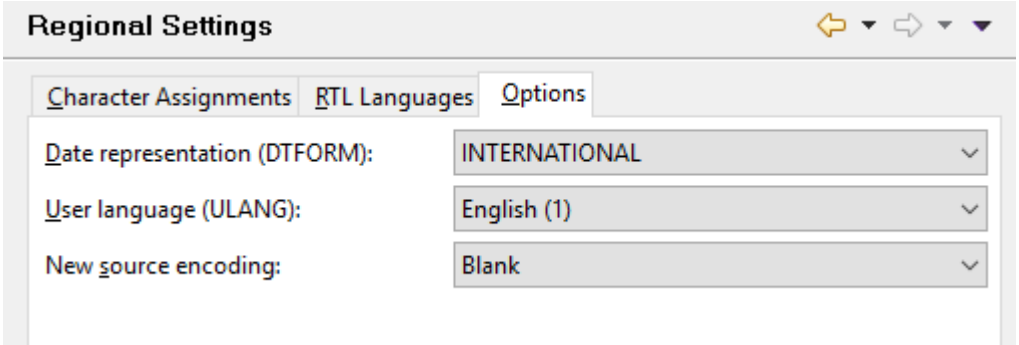
RTL Languages

On this tab, you can specify the default settings for languages that are written from right-to-left (RTL).



For further information see *Changing the Project Properties*.

Options



On this tab, you can specify the default values for the following Natural profile parameters:

Option	Corresponding Natural Profile Parameter
Date representation	DTFORM
User language	ULANG

For detailed information on these profile parameters, see the Natural documentation for the appropriate platform.

In addition, the **New source encoding** drop-down list box determines the initial setting used for the corresponding project property for newly-created projects. See *Changing the Project Properties* for further information.

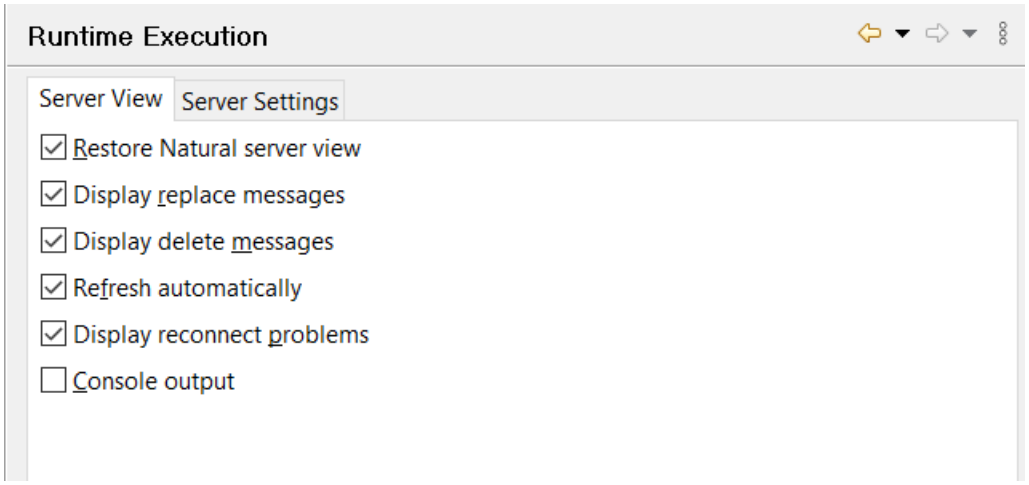
Runtime Execution

You can specify particular settings for the Natural server. Several tabs are provided on the **Runtime Execution** page:

- Server View

- Server Settings

Server View



Restore Natural server view

When selected, the same libraries that were expanded in the **Natural Server** view before you have closed NaturalONE are expanded when you start NaturalONE the next time. When not selected, server connections are not performed either with the implication that potential server problems will not be visualized via an error decorator.

Display replace messages

When selected, a message box appears each time an object in the **Natural Server** view is about to be overwritten with another object which has the same name. You are then asked whether you really want to overwrite the object.

Display delete messages

When selected, a dialog box appears each time an object in the **Natural Server** view is about to be deleted. You are then asked whether you really want to delete the object.

Refresh automatically

When selected, the display is refreshed automatically for every delete, copy-and-paste, move and rename operation in the **Natural Server** view and for every change of the filter settings in this view. It is recommended to use this setting as long as it does not cause any performance problems. See also *Refreshing the Display*.

The automatic refresh is performance-optimized and refreshes only the parent nodes of changed nodes. For this reason, redundant nodes are not always removed. For example, when you remove all programs of a library by selecting every single program and then choosing the **Delete** command from the context menu, the **Program** group node is not removed. However, when you remove all programs of a library by selecting the **Program** group node and then choosing the **Delete** command, the **Program** group node is also removed.

Note: This setting does not apply to upload operations (that is, when objects are copied to the Natural server).

Display reconnect problems

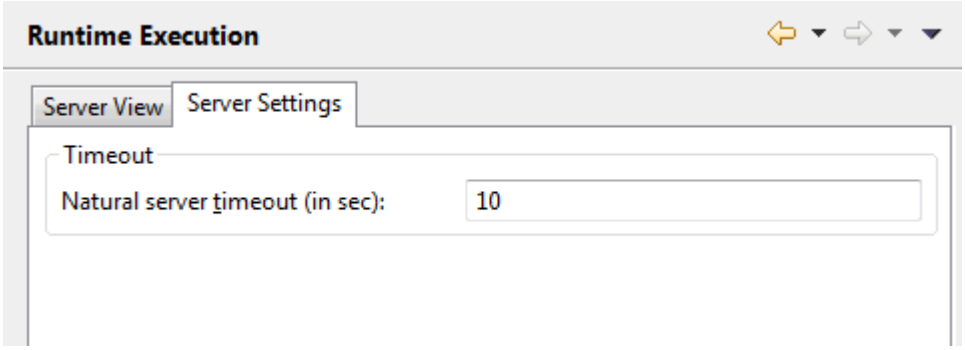
When selected, a message box appears each time you start NaturalONE and the connection to a mapped Natural server cannot be established. This is helpful, if the **Natural Server** view contains many nodes and a reconnect problem is therefore not immediately visible.

The message box contains the **Do not show this message any more** check box. When you select this check box in the message box, **Display reconnect problems** is automatically deselected in the preferences.

Console output

When selected, status information about success or failure is shown in the **Console** view each time you **check**, **stow** or **catalog** an object in the **Natural Server** view. In addition, information about the download process is shown each time you **download** objects from the **Natural Server** view into a new or existing project.

Server Settings

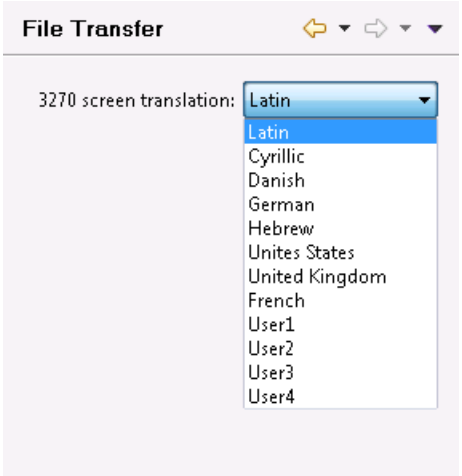


Natural server timeout (in sec)

The number of seconds that NaturalONE waits for an answer from the Natural server. The default is 10 seconds. Normally, you need not change this default value.

File Transfer

This information is required when an application using file transfer (`UPLOAD/DOWNLOAD PC`) is being executed or debugged.



3270 screen translation

Using the drop down box, you can select a character set to be used for Natural file transfer. The default character set is "Latin".

Choosing a different character set can be required if the application that is to be executed on the mainframe uses a different character set. For example, if an application with codepage 1140 is being executed on the mainframe, the 3270 screen translation should be set to "United States". Otherwise some characters might not get converted correctly. For example, if the screen

translation is still set to "Latin" (using codepage 1141) the backslash character \ will be converted to Ö which is especially fatal when using file paths.

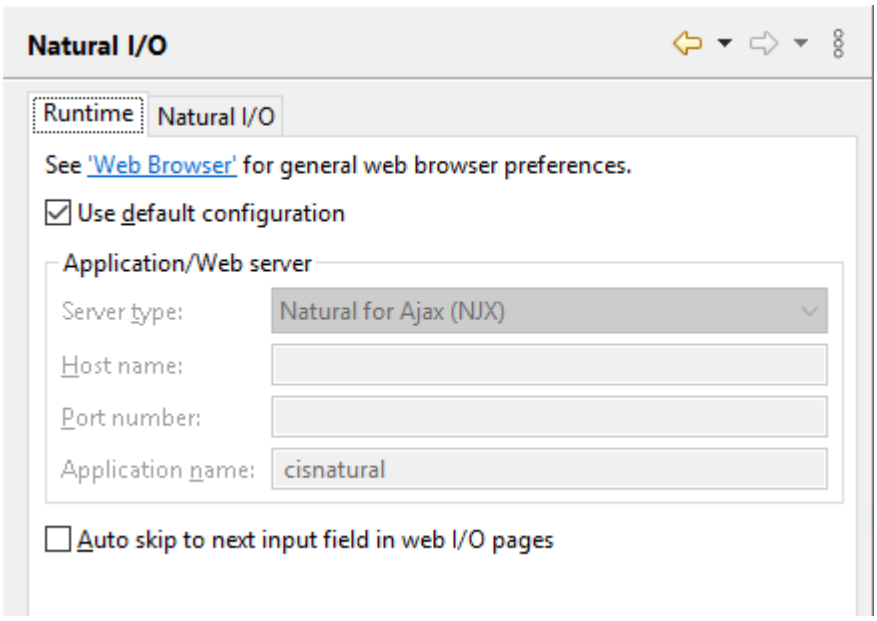
Natural I/O

This page is available when you expand the **Runtime Execution** node in the tree. It provides the following tabs:

- Runtime
- Natural I/O

Runtime

This information is required for executing and debugging Natural applications.



Use default configuration
When selected, the local Tomcat server is used to execute and debug applications. In this case, the group **Application/Web server** is disabled.

Server type
Only available when **Use default configuration** is not selected.

Select the server type that you want to use from the drop-down list box. The server type determines how your applications are displayed in the browser. Natural for Ajax (NJX) is used for displaying rich internet applications. The Natural Web I/O Interface client (NWO) is used for displaying character-based applications.

Host name

Only available when **Use default configuration** is not selected.

The name of the host on which the application server, servlet container or web server is running.

The application server, servlet container or web server is responsible for rendering the Natural output in the web browser and for passing the input to NaturalONE.

Unless the application server, servlet container or web server runs on the same PC as your Eclipse installation, see the **Natural I/O** preferences for limiting the number of open ports.

Port number

Only available when **Use default configuration** is not selected.

The number of the port on which the application server, servlet container or web server has been started.

Application name

Only available when **Use default configuration** is not selected and the server type Natural for Ajax (NJX) has been selected. Not available for other server types.

The name of the application on the application server, servlet container or web server that is running. The application server, servlet container or web server is capable of running multiple applications at once with the same host name and port number, but different application names.

Web Browser

NaturalONE displays the output of your program using the web browser specified ín the Eclipse **Web Browser** settings.

Auto skip to next input field in web I/O pages

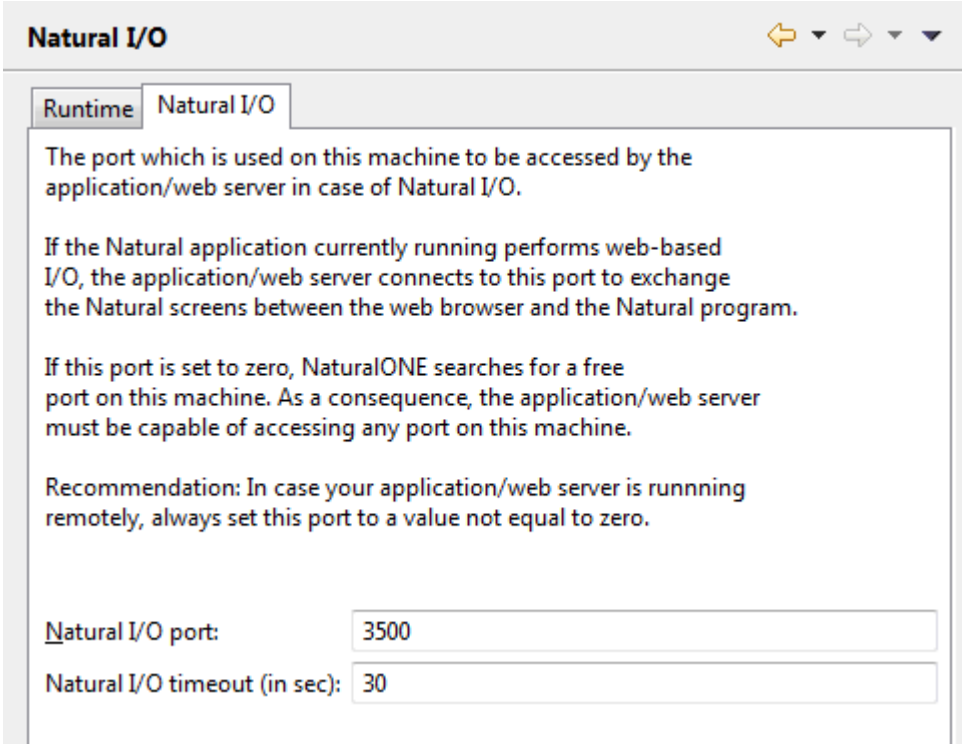
Only available when the default configuration is used or when the server type Natural for Ajax (NJX) has been selected. Not available when a server type other than Natural for Ajax is selected.

When selected, the cursor is automatically placed in the next input field when the last possible character has been entered in the current input field.

When not selected (default), the cursor remains in the current input field.

Natural I/O

This information is required when executing and debugging Natural applications.



Specify the following information:

Natural I/O port
The default port is 3500. If this port is already used on your PC, specify another port.

Natural I/O timeout (in sec)
The number of seconds that NaturalONE waits for an answer from the application/web server. The default is 30 seconds. Normally, you need not change this value.

SSL/TLS

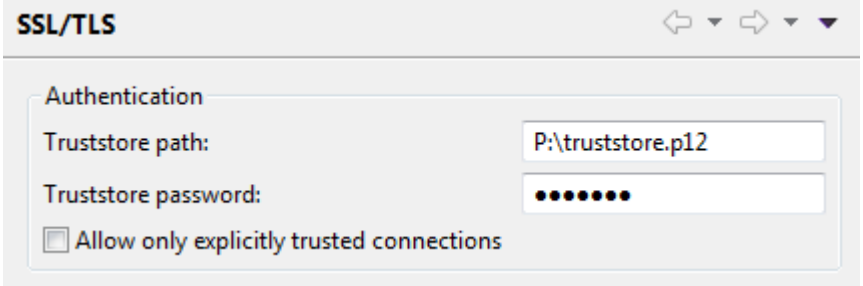
With this preference page you can setup SSL/TLS authentication. SSL/TLS authentication means checking the identity of the other party. In the case of NaturalONE, it will be checked whether the Natural Development (NDV) server to be connected to can be considered trustworthy. This is achieved via the usage of a so-called truststore containing the certificate(s) of the addressed NDV server(s).

To create your own truststore you can use, for example, Sun's *keytool* utility which can be found in the *bin* directory of the Java Runtime Environment (JRE):

```
keytool -import -alias mycert1 -file ndv.server.cert.crt -keystore truststore.p12 ↩
-storetype PKCS12
```

With the above command, the NDV server certificate named `ndv.server.cert.crt` was added to the key store `truststore.p12` with alias name `mycert1` in the certificate storage format `PKCS12`.

SSL/TLS authentication can be optional (see below) and is enabled per default.

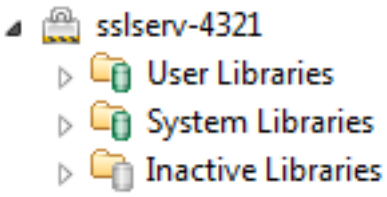


Truststore path
The path where the truststore can be found.

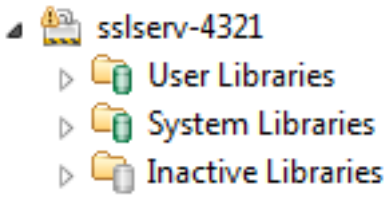
Truststore password
The password of the specified truststore. This can be empty if the truststore is not secured with a password.

Allow only explicitly trusted connections

- If enabled (default), connections to SSL/TLS-secured NDV servers are only possible if the authentication process with the provided truststore succeeds. In this case, the server's certificate must have been explicitly added to the specified truststore. The following shows a successful mapping where authentication succeeded:



- If disabled, unauthenticated connections to SSL/TLS-secured NDV servers on hosts with private IP addresses (see below) or on the same machine (localhost) will be allowed. The following shows a successful mapping where authentication failed (notice the warning decorator in the upper left corner of the server icon):

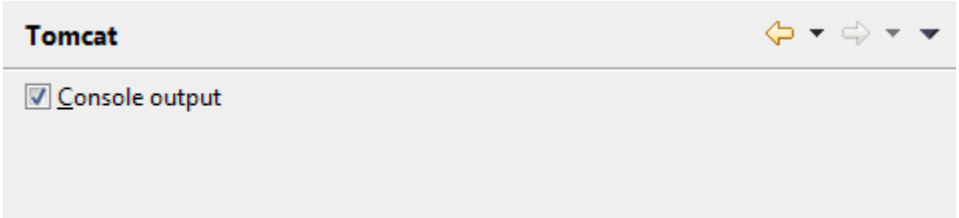


Private IP addresses are IP addresses reserved for hosts within a home or business network ("Intranet") that are not directly reachable externally. Furthermore, such hosts are invariably protected by a firewall. For these reasons, hosts with private IP addresses are much less vulnerable to external attack than those with public IP addresses.

By disabling this option, you are effectively implicitly trusting any SSL/TLS-secured NDV servers on such hosts.

Tomcat

NaturalONE includes an internal Apache Tomcat server. You can specify the following setting.



Console output

When selected (default), the console output of the internal Tomcat server is shown in the **Console** view. Such an output is shown, for example, when the Tomcat server is started after the startup of NaturalONE, or when the web context (cisnatural) in which Natural for Ajax is running has to be restarted.

When not selected, the console output is not shown in the **Console** view.

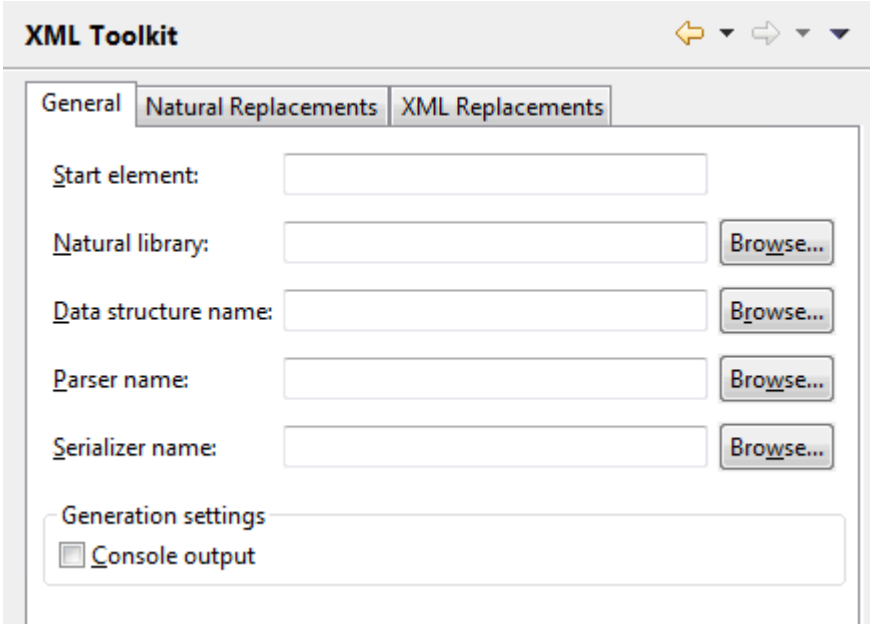
XML Toolkit

Several tabs are provided for the XML toolkit:

- General
- Natural Replacements
- XML Replacements

For further information, see ***Using the XML Toolkit***.

General



You can specify that certain items are automatically preselected in the XML toolkit wizard.

Start element
The element that is to be used as the basis for the generated output file.

Natural library
This can be a specific library.

Data structure name
This can be a Natural data area, an XML schema or a DTD from a specific library.

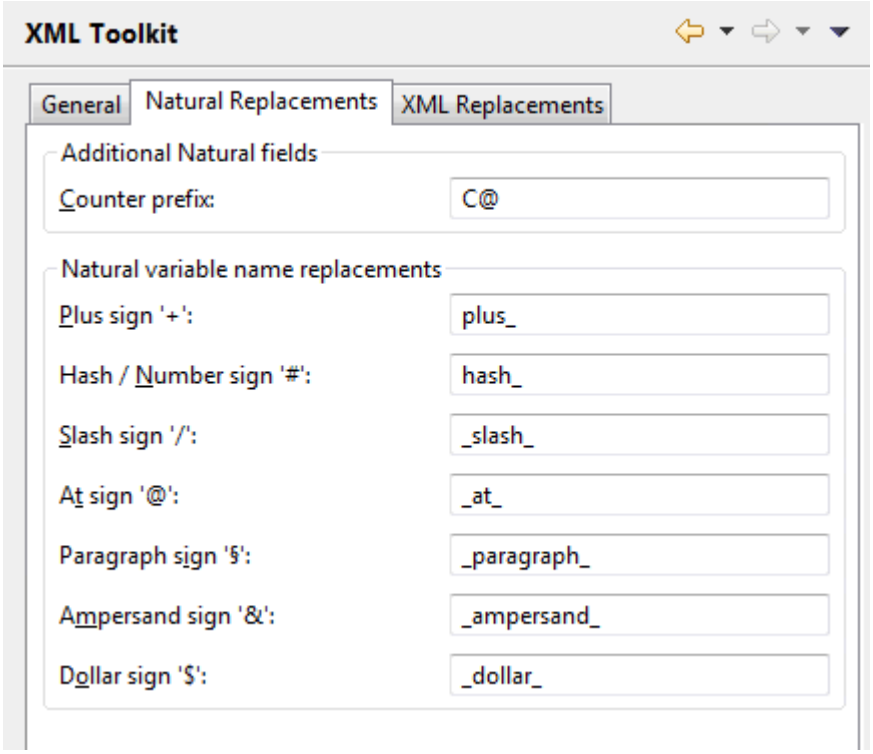
Parser name
This can be a subprogram or copycode from a specific library.

Serializer name
This can be a subprogram or copycode from a specific library.

Console output
When selected, trace information is shown in the **Console** view during the generation of the output files.

Natural Replacements

Special characters that are not valid in XML have to be converted into valid names. You can change the default conversion settings, if required.



You can change the following settings:

Additional Natural fields

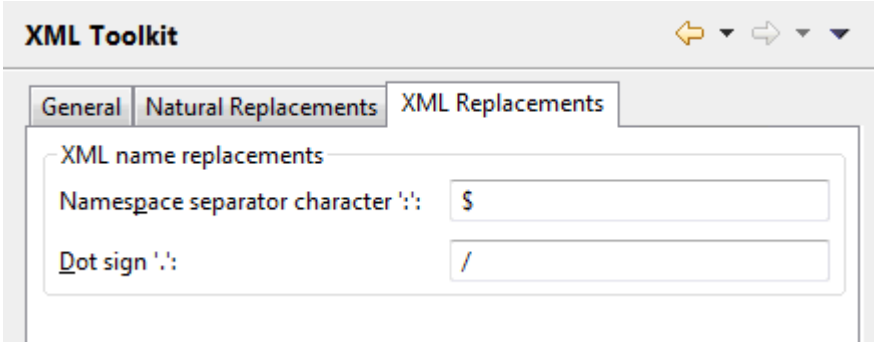
Character	Default Value
Counter prefix	C@

Natural variable name replacements

Character	Default Value
Plus sign '+'	plus_
Hash/Number sign '#'	hash_
Slash sign '/'	_slash_
At sign '@'	_at_
Paragraph sign '§'	_paragraph_
Ampersand sign '&'	_ampersand_
Dollar sign '$'	_dollar_

XML Replacements

Special characters that are not valid in XML have to be converted into valid names. You can change the default conversion settings, if required.



You can change the following settings:

XML name replacements

Character	Default Value
Namespace separator character ':'	$
Dot sign '.'	/