

Natural for Mainframes

Unicode and Code Page Support

Version 4.2.6 for Mainframes

October 2009

This document applies to Natural Version 4.2.6 for Mainframes and to all subsequent releases.

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

Copyright © Software AG 1979-2009. All rights reserved.

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

Table of Contents









| | |
|--|----|
| 1 Unicode and Code Page Support | 1 |
| 2 Introduction | 3 |
| About Code Pages and Unicode | 4 |
| About Unicode and Code Page Support in Natural | 5 |
| ICU on Mainframe Platforms | 6 |
| 3 Unicode and Code Page Support in the Natural Programming Language | 7 |
| Natural Data Format U for Unicode-Based Data | 8 |
| Statements | 9 |
| Logical Condition Criteria | 13 |
| System Variables | 14 |
| Large and Dynamic Variables | 14 |
| Session Parameters | 15 |
| Sample Programs | 17 |
| 4 Configuration and Administration of the Unicode/Code Page Environment | 19 |
| ICU Library | 20 |
| Customizing the ICU Data Library for Mainframe Platforms | 21 |
| Profile Parameters | 22 |
| Encoding Information | 24 |
| Deploying Natural Objects with Encoding Information | 25 |
| 5 Development Environment | 27 |
| Development Environment | 28 |
| Customizing Your Environment | 29 |
| Editors | 30 |
| Code Page Support for Editors, System Commands and Utilities on the Mainframe | 32 |
| 6 Unicode Input/Output Handling in Natural Applications | 35 |
| Displaying and Entering Unicode Data | 36 |
| Natural Web I/O Interface Client | 37 |
| 7 Unicode Data Storage | 41 |
| Unicode Data/Parameter Access | 42 |
| Database Management System Interfaces | 42 |
| Work Files and Print Files on Windows, UNIX and OpenVMS Platforms | 43 |
| Work Files and Print Files on Mainframe Platforms | 47 |
| 8 Platform Differences | 49 |
| Windows, UNIX and OpenVMS Platforms | 50 |
| Mainframe Platforms | 51 |
| 9 Migrating Existing Applications | 61 |
| Impact of Unicode on Existing Applications | 62 |
| Migrating Existing Objects on Windows, UNIX and OpenVMS Platforms | 62 |
| Migrating Existing Objects on Mainframe Platforms | 63 |
| Adding Unicode Support to Existing Applications | 65 |
| Migrating Natural Remote Procedure Calls (RPC) | 66 |
| 10 Special Considerations and Limitations | 67 |





| | |
|---|----|
| Windows, UNIX and OpenVMS Platforms | 68 |
| Mainframe Platforms | 68 |
| 11 Bidirectional Language Support | 69 |
| 12 Double-Byte Character Support | 75 |
| 13 Frequently Asked Questions | 77 |
| Why do I get the startup error "Invalid code page specified"? | 78 |
| What is the "default code page"? | 78 |
| What default code page is used? | 78 |
| Should I save all Natural sources in UTF-8 format? | 78 |
| How can I handle UTF-8 encoding with Natural code? | 79 |
| Why are some characters not displayed correctly? | 79 |
| Why do I get an error when I want to edit a Natural source? | 79 |
| Why do I get an error when I want to save a Natural source? | 79 |
| How can I find out the encoding of a Natural source? | 80 |
| How can I change the encoding of a Natural source? | 80 |
| How can I convert an existing Natural source into UTF-8 format (Windows, UNIX and OpenVMS only)? | 80 |
| Which substitution character is used if a character cannot be converted? | 81 |
| Can I use Natural 4.2 sources with previous Natural versions? | 81 |
| Can I use UTF-8 sources with previous Natural versions? | 81 |
| Why do I get a conversion error when cataloging a source which has UTF-8 format? | 81 |
| Why do I get garbage On UNIX or OpenVMS when displaying U format via a terminal emulation? | 82 |
| Can I work with a current SPoD client and an older SPoD server? | 82 |
| Can I work with a current SPoD server and an older SPoD client? | 82 |
| Index | 83 |

1 Unicode and Code Page Support

This documentation describes how Natural supports Unicode and code pages on Windows, UNIX, OpenVMS and mainframe platforms. It also describes how Natural supports bidirectional languages and double-byte characters.

This documentation is organized under the following headings:

| | | |
|---|--|--|
|  | Introduction | General information on code pages and the Unicode Standard, and on how Unicode and code pages are supported in Natural. |
|  | Unicode and Code Page Support in the Natural Programming Language | Information on the U format and on statements, logical condition criteria, system variables, large and dynamic variables, and session parameters which provide Unicode and code page support. |
|  | Configuration and Administration of the Unicode/Code Page Environment | Information on the ICU library, on profile parameters which provide Unicode and code page support, and on the encoding of code page data. |
|  | Development Environment | How to customize your environment and how Unicode is handled by the Natural editors. Information on code page support for Natural sources on the mainframe (editors, system commands and utilities). |
|  | Unicode Input/Output Handling in Natural Applications | How to display and enter Unicode data. Information on the Natural Web I/O Interface client which is used in SPoD and runtime environments. |
|  | Unicode Data Storage | Information on database access, and on the work file types and print files which provide Unicode and code page support. |
|  | Platform Differences | Handling differences on Windows, UNIX, OpenVMS and mainframe platforms. |
|  | Migrating Existing Applications | About the impact of Unicode on existing applications. How to migrate existing objects, add Unicode support to existing applications, and how to migrate Natural remote procedure calls (RPC). |

| | | |
|---|---|--|
|  | Special Considerations and Limitations | Important information and restrictions on the different platforms. |
|  | Bidirectional Language Support | How Natural supports bidirectional languages. |
|  | Double-Byte Character Support | How Natural supports double-byte characters. |
|  | Frequently Asked Questions | Answers to frequently asked questions. |

2 Introduction

| | |
|--|---|
| ■ About Code Pages and Unicode | 4 |
| ■ About Unicode and Code Page Support in Natural | 5 |
| ■ ICU on Mainframe Platforms | 6 |

About Code Pages and Unicode

A traditional code page is a list of selected character codes, arranged in a certain order, that support specific languages or groups of languages that share common scripts. A code page can contain a maximum of 256 character codes. For character sets which contain more than 256 characters (for example, Chinese or Japanese), double-byte code unit handling (DBCS) is used: DBCS code pages are actually multi-byte encodings, a mix of 1-byte and 2-byte code points.

Code pages have the inherent disadvantage of not being able to be used to store different languages in the same data stream. Unicode was designed to remove this restriction by providing a standard encoding for all character sets which is independent of the platform, program, or language used to access the data. With Unicode, a unique number is provided for every character.

A single number is assigned to each code element defined by the Unicode Standard. Each of these numbers is called a “code point” and, when referred to in text, is listed in hexadecimal form following the prefix “U”. For example, the code point “U+0041” is the hexadecimal number “0041” (equal to the decimal number “65”). It represents the character “A” in the Unicode Standard which is named “LATIN CAPITAL LETTER A”.

The Unicode Standard defines three encoding forms that allow the same data to be transmitted in a byte, word or double word oriented format. A “code unit” is the minimal bit combination that can represent a character in a specific encoding. The Unicode Standard uses 8-bit code units in the UTF-8 encoding form, 16-bit code units in the UTF-16 encoding form, and 32-bit code units in the UTF-32 encoding form. All three encoding forms encode the *same* common character repertoire and can be efficiently transformed into one another without loss of data.

In the context of Natural, we are concerned with two of these encoding forms: UTF-16 and UTF-8. Natural uses UTF-16 for the coding of Unicode strings at runtime and UTF-8 for the coding of Unicode data in files. UTF-16 is an endian-dependant 2-byte encoding; the endian format that will be used depends on the platform. UTF-8 is a 1-byte encoding.

For a complete description of Unicode, see the Unicode consortium web site at <http://www.unicode.org/>.



Note: For obtaining information on Unicode code points, you can use the SYSCP utility which is available with Natural for Windows.

About Unicode and Code Page Support in Natural

In previous versions, Natural supported only code page characters. Starting with Natural versions 4.2 (mainframe), 6.2 (Windows and UNIX) and 6.3 (OpenVMS) Unicode is supported.

For Unicode support, the Natural data format U has been introduced and there are specific statements, parameters, system variables, etc. For details, see the remainder of this documentation.

Currently, most existing data is available in code page format. When converting this data to Unicode, it is required that the correct code page is used. Natural provides the possibility to define the correct code page on several levels:

- The system code page is used if a default code page is not defined in Natural.

If no code page is defined on a mainframe (`CP=OFF`), a default code page is not defined. `CP=AUTO` is intended to adjust the Natural session to the code page of the current I/O device.

- The default code page is used when the Natural parameter `CP` is defined; this overwrites the operating system's code page.
- The object code page which is defined, for example, for a source overwrites the default code page for this object.

When using Unicode strings and code page strings in one application, Natural does implicit conversions where necessary (for example, when moving or comparing data). Explicit conversions can be performed with the statement `MOVE ENCODED`.

In most cases, existing applications which do not require Unicode support, will run unchanged. On Windows, UNIX and OpenVMS platforms, changes can be necessary if the existing sources are encoded in different code pages. For more information, see *Migrating Existing Applications* later in this documentation.

It is not possible to run an existing application and also support Unicode data without any changes to the application. The Natural data format U has to be introduced in the application and it will most probably not suffice to simply replace the A format definitions with U format definitions. All code which assumes a specific memory layout of strings (for example, `REDEFINE` from alphanumeric to numeric format) has to be adapted.

Unicode characters are not permitted within variable names, object names and library names.

Unicode-based data are supported for Adabas and DB2.

Natural uses the International Components for Unicode (ICU) library for Unicode collation and conversion. For more information, see <http://icu.sourceforge.net/userguide/>. See also *ICU Library* later in this documentation.

ICU on Mainframe Platforms

Information on the currently used ICU version and Unicode specification is provided in the main menu of the SYSCP utility. See *Invoking and Terminating SYSCP* in the *Utilities* documentation of the Natural for Mainframes documentation.

3 Unicode and Code Page Support in the Natural Programming Language

| | |
|--|----|
| ■ Natural Data Format U for Unicode-Based Data | 8 |
| ■ Statements | 9 |
| ■ Logical Condition Criteria | 13 |
| ■ System Variables | 14 |
| ■ Large and Dynamic Variables | 14 |
| ■ Session Parameters | 15 |
| ■ Sample Programs | 17 |

Natural Data Format U for Unicode-Based Data

In Natural, you can specify Unicode strings with the format U and U constants.

■ Format U

With format U, you can define data which holds Unicode strings. The Natural data format U is internally UTF-16.

See also *Format and Length of User-Defined Variables* in the *Programming Guide*.

■ U Constants

You can define Unicode constants with the prefix "U". For example:

```
U'Äpfel'
```

The prefix "UH" can be used for defining Unicode constants in hexadecimal format. Four hexadecimal digits represent one UTF-16 code unit as defined by the Unicode Standard. So the overall length must be a multiple of four. For example, if you need the hexadecimal form of

```
U'Äpfel'
```

you need the UTF-16 code units for "Ä", "p", "f", "e" and "l" (which are "U+00C4", "U+0070", "U+0066", "U+0065" and "U+006C") and you have to combine them to the following hexadecimal string:

```
UH'00C4007000660065006C'
```

See also *Unicode Constants* in the *Programming Guide*.

The data format U is endian-dependant. This has to be considered when moving between the formats B and U.

U versus A

The advantage of the U format (as compared with the A format) is, that it can hold any combinations of characters from different languages and that it does not depend on the default code page (value of the system variable *CODEPAGE). Moreover, the U format makes it easier to share data between different platforms; no more conversions (for example, from EBCDIC to ASCII) are necessary.

On the other hand, U format data consumes more memory than A format data. For languages in which most strings can be represented by single-byte encoding, U format will result in strings occupying twice the space that was previously required. However, for East Asian languages, the memory consumption will often not be higher.

Statements

Basically, U format can be used in most statements which allow A format. However, if a Natural object name is given as an operand of a statement (for example, in the `CALLNAT` statement), U cannot be used because Natural object names have A format. For information on a specific statement, see the *Statements* documentation.

Basically, A and U format can be used together in one statement, for example:

```
EXAMINE S FOR P WITH DELIMITER D REPLACE R
```

where S is U format, and P, D and R are A format.

In the above example, the variables P, D and R are temporarily converted into the target format U before the actual execution of the `EXAMINE` statement. The conversion from Unicode to code page or vice versa requires calling an ICU function. The conversion requires additional computing time and additional memory. This disadvantage is even greater with very large variables. To avoid frequent conversions, it is recommended that you use only one format within one statement. When all operands in the above example are specified in either U format or A format, a conversion is not necessary. However, if you choose to specify only U operands, this variant will be slower since (due to its nature) this operand type consumes more resources; one character is then coded with 2 bytes (instead of 1 byte which is used with A format).

With a conversion (especially from U format to A format), there is always the risk that characters cannot be represented in the target code page. For example, you want to convert the Unicode character "U+05D0" (Hebrew letter Alef) into the code page IBM01140 (English). Since this character is not contained in the code page IBM01140, either the substitution character for this code page is used, or the place holder which was specified when defining the code page in `NATCONFIG` (mainframe only). When the parameter `CPCVERR` is set to `ON`, an error message will be issued in this case, indicating a conversion error. In any case, the original information will be lost.

The following statements are particularly affected when using Unicode:

- `MOVE NORMALIZED`
- `MOVE ENCODED`
- `EXAMINE`
- `PARSE XML`
- `REQUEST DOCUMENT`
- `DEFINE PRINTER`

- **CALLNAT (RPC)**

MOVE NORMALIZED

Normalization in Unicode: A process of removing alternate representations of equivalent sequences from textual data in order to convert the data into a form that can be binary-compared for equivalence. The Unicode Standard defines different normalization forms. The normalization form that results from the canonical decomposition of a Unicode string, followed by the replacement of all decomposed sequences by primary composites where possible, is called “Normalization Form Composed” (NFC).

Natural assumes that all Unicode data is in NFC format to assure that string operations can be performed without partial truncation of a Unicode character. Natural conversion operations assure that the resulting Unicode string is in NFC. If Unicode data is received from outside of Natural and it is not guaranteed that the data has NFC format, the `MOVE NORMALIZED` statement can be applied.

Example:

| Character Sequence | NFC |
|-------------------------|------------|
| ê (U+00EA) | ê (U+00EA) |
| e (U+0065) + ^ (U+0302) | ê (U+00EA) |



Note: Concatenating two or more strings in NFC format can result in not-NFC format.

MOVE ENCODED

An implicit conversion between Unicode and the default code page (value of the system variable `*CODEPAGE`) is performed when moving strings from U to A or vice versa with the `MOVE` statement.

Furthermore, the `MOVE ENCODED` statement can be used for conversion between different code pages or from any available code page to Unicode and vice versa. This can be helpful if data is coming from outside of Natural and this data is coded in a code page which differs from the default code page. But even for conversions between the default code page and Unicode, this statement can be used if you want to obtain a potential conversion error with the `GIVING` clause; if `CPCVERR` is set to `ON`, the `MOVE` statement will stop with a runtime error in this case.

If a character cannot be converted, it depends on the setting of the `CPCVERR` parameter whether a substitution character is used for this character or whether the conversion fails. On Windows, UNIX and OpenVMS platforms, the default substitution character (defined by ICU) for the conversion from Unicode to the default code page (CP) can be changed with the profile parameter `SUBCHAR`.

This statement can also be used for conversion from U data into UTF-8 format.



Note: If you convert data to a code page which differs from the default code page, it is recommended not to use this data in I/O. I/O is only meaningful with the default code page.

EXAMINE

A “grapheme” is what a user normally thinks of as a character. In most cases, a Unicode code point is a grapheme, however, a grapheme can also consist of several Unicode code points. For example, a sequence of one base character and one or more combining characters is a grapheme.

Example: "a" (U+0061) + "." (U+002E) + "^" (U+005E) defines one grapheme which is displayed as follows:

²

Note: If a base/combining character sequence is normalized, this does not mean that the sequence is always replaced by a pre-composed character, because not all characters are available in a pre-composed format.

A “supplementary code point” is a Unicode code point between “U+10000” and “U+10FFFF”. A supplementary code point is in UTF-16, represented by a surrogate pair which consists of two code units where the first value of the pair is a “high-surrogate code unit”, and the second is a “low-surrogate code unit”. Such characters are generally rare, but some are used, for example, as part of Chinese and Japanese personal names, and therefore support for these characters is commonly required for government applications in East Asian countries.

The string handling statements such as `EXAMINE` and its `SUBSTRING` option work on UTF-16 code units. It is the user's responsibility that the code does not separate graphemes or surrogate pairs.

However, the clauses `CHARPOSITION` and `CHARLENGTH` of the `EXAMINE` statement (see *Syntax 3 - EXAMINE for Unicode Graphemes*) can be used to ask for the start and length (in UTF-16 code units) of graphemes. The result values can be used for `SUBSTRING` calls. With these clauses, it is possible to scan a string grapheme by grapheme.

Example:

```

DEFINE DATA LOCAL
1 #UNICODE-STRING      (U15)
1 #CODE-UNIT-INDEX     (N4)
1 #CODE-UNIT-LEN       (N4)
1 #GRAPHEME-NUMBER     (N4)
END-DEFINE

MOVE U'Unicode' TO #UNICODE-STRING
#GRAPHEME-NUMBER := 1

```

```
REPEAT
EXAMINE
    FULL VALUE OF #UNICODE-STRING
    FOR CHARPOSITION #GRAPHEME-NUMBER
    GIVING POSITION IN #CODE-UNIT-INDEX
    GIVING LENGTH IN #CODE-UNIT-LEN

    DISPLAY #UNICODE-STRING #GRAPHEME-NUMBER #CODE-UNIT-INDEX #CODE-UNIT-LEN

    #GRAPHEME-NUMBER := #GRAPHEME-NUMBER + 1
WHILE #CODE-UNIT-INDEX NE 0
END-REPEAT

END
```

The above example program provides the following output:

| | | | |
|-----------------|------------------|------------------|----------------|
| Page | 1 | 05-12-15 | 09:33:49 |
| #UNICODE-STRING | #GRAPHEME-NUMBER | #CODE-UNIT-INDEX | #CODE-UNIT-LEN |
| ----- | ----- | ----- | ----- |
| aᳵcᳵbᳵcᳵd | 1 | 1 | 1 |
| aᳵcᳵbᳵcᳵd | 2 | 2 | 2 |
| aᳵcᳵbᳵcᳵd | 3 | 4 | 1 |
| aᳵcᳵbᳵcᳵd | 4 | 5 | 3 |
| aᳵcᳵbᳵcᳵd | 5 | 8 | 1 |
| aᳵcᳵbᳵcᳵd | 6 | 9 | 3 |
| aᳵcᳵbᳵcᳵd | 7 | 12 | 1 |
| aᳵcᳵbᳵcᳵd | 8 | 13 | 3 |
| aᳵcᳵbᳵcᳵd | 9 | 0 | 0 |

PARSE XML

XML documents can contain information within the XML document header about the encoding of the document (for example, <?xml version="1.0" encoding="UTF-8" ?>). If an XML document contains this information, the parsing of the XML document on Windows, UNIX and OpenVMS platforms always includes a conversion of the code page given within the XML document header to the default code page of Natural (value of the system variable *CODEPAGE), if the receiving field is not of format U.

On mainframe platforms, the document to be parsed is always internally converted to UTF-16 (if the document is not already encoded in UTF-16).

See the description of the PARSE XML statement for further information.

See also *Statements for Internet and XML Access* in the *Programming Guide*.

REQUEST DOCUMENT

Data transfer with the `REQUEST DOCUMENT` statement normally does not involve any code page conversion. If you want to have the outgoing and/or incoming data encoded in a specific code page, you can use the `DATA ALL` clause and/or the `RETURN PAGE` clause of the `REQUEST DOCUMENT` statement to specify this.

See the description of the `REQUEST DOCUMENT` statement for further information.

See also *Statements for Internet and XML Access* in the *Programming Guide*.

DEFINE PRINTER

On mainframe platforms, the `DEFINE PRINTER` statement provides a `CODEPAGE` clause to provide for conversion of print report data into a code page different from the default code page (value of the system variable `*CODEPAGE`). On Windows, UNIX and OpenVMS platforms, the `DEFINE PRINTER` statement does not provide such a clause; if the `CODEPAGE` clause is defined, it is ignored on Windows, UNIX and OpenVMS platforms.

CALLNAT (RPC)

Data exchange in Unicode format via RPC is supported. See the description of the `CALLNAT` statement.

If U data is sent from a platform with big endian encoding to a platform with little endian encoding or vice versa, the encoding is adapted so that it conforms with the encoding on the receiving platform. For example, when U data in little endian encoding arrives on a big endian platform, this data is converted to big endian encoding before it is handed over to the program. When this data is sent back, it is converted back to little endian encoding.

Logical Condition Criteria

In a logical condition criterion, Unicode operands can be used together with alphanumeric and binary operands. If not all operands are Unicode operands (format U), the second and all following operands are converted to the format of the first operand. If a binary operand (format B) is specified as the second or a following operand, the length of the binary operand must be even; the binary operand is assumed to contain Unicode code points.

If the first operand is a Unicode operand (format U) and the comparison is therefore performed as a Unicode comparison, the ICU collation algorithm is used. The ICU algorithm does not perform a plain binary comparison. For example,

- some code points such as "U+0000" are ignored during the comparison process,

- combined characters are considered as being equal to the equivalent single code point (for example, the German character "ä" represented by "U+00E4" and the combination of the code points "U+0061" and "U+0308" are considered as being equal by ICU).



Note: Comparing an alphanumeric and a Unicode operand can deliver different results, depending on the sequence of the fields.

See also *Logical Condition Criteria* in the *Programming Guide*.

System Variables

This section covers the following topics:

- `*CODEPAGE`
- `*LOCALE`

`*CODEPAGE`

The system variable `*CODEPAGE` is used to return the IANA name of the default code page, that is, the code page used for conversions between Unicode and code page format.

`*LOCALE`

The system variable `*LOCALE` contains the language and country of the current locale.

Large and Dynamic Variables

U format can be used for large and dynamic variables. For dynamic U variables, `*LENGTH` returns the number of UTF-16 code units.

See also *Introduction to Dynamic Variables and Fields* in the *Programming Guide*.

Session Parameters

The following session parameters are available:

| Parameter | Description |
|-----------|---|
| DL | Specifies the display length for a field of format A or U. See also <i>Display Length for Output - DL Parameter</i> in the <i>Programming Guide</i> . |
| EMU * | Edit mask in Unicode. |
| ICU * | Insertion character in Unicode. |
| LCU * | Leading characters in Unicode. |
| TCU * | Trailing characters in Unicode. |

Session parameters marked with an asterisk (*) in the above table are only available on Windows, UNIX and OpenVMS platforms.

DL **versus** AL

As long as Natural was not Unicode-enabled, the length of an alphanumeric field was always identical to the number of columns needed for displaying the field (called number of display columns). This was even true for the East Asian languages which use DBCS code pages: an A format field can hold only half the characters (for example, A10 results in A5).

Example:

```

DEFINE DATA LOCAL
1  #A8 (A8)
END-DEFINE
#A8 := 'computer'
WRITE #A8
#A8 := '電腦系統'
WRITE #A8
END

```

The above code results in the following output:

```

Page      1 ...
computer
電腦系統

```

With U format fields, the length of a field and the number of display columns is no longer identical. U characters can have narrow width (for example, Latin characters), wide width (for example, Chinese characters) or no width (for example, combining characters). Therefore, it is

totally unknown how many display columns a U field needs; this depends on the contents of the field. Natural cannot automatically decide how many columns are to be reserved on the screen: if the maximum size is assumed, Latin output will have large gaps, and if the minimum size is assumed, Chinese output cannot be displayed totally. Therefore, the Natural programmer has to define the display width of a field; this is done with the `DL` parameter. The `AL` parameter cannot be used for this purpose, because it cuts away the part of the field which exceeds the defined length. But we do not want to cut any characters from the U field; we only want to define the start position of the following field.

Example:

```
DEFINE DATA LOCAL
1  #U8 (U8)
1  #U4 (U4)
END-DEFINE
#U8 := 'computer'
WRITE #U8
#U4 := U'電腦系統'
WRITE #U4 (DL=8)
END
```

The above code results in the same output as above:

```
Page      1  ...

computer
電腦系統
```

On Windows, either locally with the output window or in a remote development environment with the Natural Web I/O Interface client, it is possible to scroll in a field where the defined value for the `DL` parameter is smaller than the real display width of the field.

EMU, ICU, LCU, TCU **versus** EM, IC, LC, TC

The parameters `EMU`, `ICU`, `LCU` and `TCU` (which are only available on Windows, UNIX and OpenVMS platforms) allow using characters which are not included in the default code page. They are stored in Unicode format in the generated program. These parameters can be used with all field formats.

The parameters `EM`, `IC`, `LC` and `TC` can also be used with U format fields. These parameters may also be useful if characters which are contained in the default code page have different encodings in other code pages. For example, the Euro sign (€) has the code point "0x80" in the "windows-1252" (Latin 1) code page, but the code point "0x88" in the "windows-1251" (Cyrillic) code page. Thus, using a Unicode parameter will assure that the Euro sign is always displayed correctly, no matter what code page is installed on the PC.

Example for EMU:

```

DEFINE DATA
LOCAL
  01 EMPLOYEES-VIEW VIEW OF EMPLOYEES
    02 FIRST-NAME
    02 NAME
    02 SALARY (1)
END-DEFINE
*
  READ (6) EMPLOYEES-VIEW
    DISPLAY NAME FIRST-NAME SALARY(1) (EMU=999,999€)
  END-READ
*
END

```

The above code results in the following output:

| Page | 1 | | 05-12-15 11:45:36 |
|----------|------------|------------------|-------------------|
| NAME | FIRST-NAME | ANNUAL SALARY | |
| ADAM | SIMONE | 159,980€ | |
| MORENO | HUMBERTO | 165,810€ | |
| BLOND | ALEXANDRE | 172,000€ | |
| MAIZIERE | ELISABETH | 166,900€ | |
| CAUDAL | ALBERT | 167,350€ | |
| VERDIE | BERNARD | 170,100€ | |

Sample Programs

The library SYSEXPB contains sample programs for Unicode and code page support in Natural:

- UNICOX01 lists all Unicode characters.
- UNICOX02 converts Unicode characters to code points and vice versa.
- CODEPX01 lists all code pages, whether the code page is supported in Natural and which encoding it uses. For all supported code pages, it offers services to list the characters of the code page and to convert a string from the code page into its hexadecimal representation and vice versa.
- CODEPXL1 lists all characters of any 1-byte code page.
- CODEPXL2 lists all characters of any 2-byte code page.
- CODEPXC1 converts a string from any code page into its hexadecimal representation and vice versa.

4 Configuration and Administration of the Unicode/Code Page Environment

| | |
|--|----|
| ■ ICU Library | 20 |
| ■ Customizing the ICU Data Library for Mainframe Platforms | 21 |
| ■ Profile Parameters | 22 |
| ■ Encoding Information | 24 |
| ■ Deploying Natural Objects with Encoding Information | 25 |

ICU Library

Windows, UNIX and OpenVMS Platforms

The ICU libraries are always installed with the full set of ICU conversion and collation data. The settings in the configuration file *NATCONV.INI* apply to the A format. For the U format, the corresponding checks (for example, when a character is translated to upper case) are done via the ICU library.



Note: For obtaining information on the ICU version and the supported code pages, you can use the SYSCP utility which is available with Natural for Windows.

Mainframe Platforms

The relevant modules can be linked statically to the shared nucleus or loaded dynamically by means of the RCA technique. See *NATICU Modules for Different Purposes*.

For running applications without Unicode and without code page support, that is, with the profile parameter settings *CFICU=OFF* and *CP=OFF*, none of the supplied ICU modules needs to be linked to the Natural nucleus.



Note: Information on the currently used ICU version and Unicode specification is provided in the main menu of the SYSCP utility. See *Invoking and Terminating SYSCP* in the *Utilities* documentation of the Natural for Mainframes documentation.

Three different load modules are offered:

| Load Module | Description |
|-------------|--|
| NATICU | Contains code page and Unicode conversion functionality as well as collation services. The first is needed for conversion from one code page to another code page or to Unicode and vice versa. The latter is used for string comparison of Unicode strings with consideration of locale ID. NATICU contains the most popular code pages and locales. The code pages are already declared in NATCONFIG. |
| NATICUCV | Same as NATICU, but without collation services. Therefore, it is smaller. |
| NATICUXL | Same as NATICU, but it contains all possible converters and locales offered by the currently supported ICU version. It supports about 230 different code pages (predominantly EBCDIC code pages) and 238 locales. Therefore, the module size is huge. If NATICUXL is linked to the Natural nucleus, the desired code pages have to be declared in the configuration file NATCONFIG. |

| Load Module | Description |
|-------------|--|
| | <p>NATCUXL supports all code pages and locale IDs which are supported by the currently supported ICU version (see http://demo.icu-project.org/icu-bin/convexp).</p> <p>Note: Due to technical restrictions, NATCUXL is not delivered for z/VSE.</p> |

See also [NATICU Modules for Different Purposes](#).

Note for z/VSE:

If you receive an error during linkage editing when you try to link one of the ICU load modules statically to the shared nucleus because the size of the resulting phase is too large for your z/VSE partition, proceed as follows:

- Check whether NATICUCV can be used instead of NATICU (if you tried to link NATICU).
- Load the relevant module dynamically by means of the RCA technique. See [NATICU Modules for Different Purposes](#). Modules NATICU and NATICUCV are also delivered as a phase to make use of the RCA technique more convenient.
- Ask your system administrator to configure your partition so that the storage available for the linkage editor is sufficient to hold the resulting phase.

Additional tables:

| Table | Description |
|----------|--|
| NATSCTU | Required scanner table for Unicode characters. It maps the properties of Unicode characters of the currently supported Unicode version to be used by the Natural nucleus. This table must never be changed. |
| NATCPTAB | Optional single-byte code page conversion accelerator tables. If the table is present, conversion from one code page to another code page will be faster since it is performed via this table rather than by calling ICU functions. For more information, see <i>Translation Tables</i> in the <i>Operations</i> documentation for Natural for Mainframes. |

Customizing the ICU Data Library for Mainframe Platforms

ICU makes use of a wide variety of data tables to provide many of its services. Examples include converter mapping tables, collation rules, transliteration rules, break iterator rules and dictionaries, and other locale data. The ICU data library for Natural is provided as a package that contains the desired data items. The usage of packages instead of single data item files increases the performance since there is only one file access during the initialization to load the package. However, it is not so flexible since it requires a rebuild of a package if data items need to be added.

The ICU data library may be customized in order to add existing or new converter mapping tables or to add other data items such as collation rules, break iterator rules and other locale data.

The customization tool for the ICU data library is available from the Component Downloads area in ServLine24 (<http://servline24.softwareag.com/public/>). Detailed information on how to customize the ICU data library is provided in the *readme.txt* file which is part of the downloaded zip file.

Profile Parameters

This section lists the profile parameters which are used in conjunction with Unicode and code page support. Not all profile parameters are available on all platforms.

- [All Platforms](#)
- [Windows, UNIX and OpenVMS Platforms](#)
- [Mainframe Platforms](#)
- [Natural Development Server for Mainframes](#)

All Platforms

The following profile parameters are available on all platforms:

| Parameter | Description |
|-----------|---|
| CP | <p>Defines the default code page for Natural. This code page is used for the runtime and development environment if not superposed with a code page defined for a single object (for example, for a Natural source).</p> <p>Only platform-suitable code pages can be used. This means, for example, that no EBCDIC code page can be defined for a Windows, UNIX or OpenVMS platform, or that no ASCII code page can be defined for a mainframe platform. On mainframes, an initialization error message occurs if a wrong code page is used.</p> <p>Note: As of Natural Version 6.2 for Windows and UNIX, Natural Version 6.3 for OpenVMS and Natural Version 4.2 for Mainframes, the existing CP parameter (used with Natural RPC) has been renamed to CPRPC.</p> |
| CPCVERR | <p>Specifies whether a conversion error that occurs when converting from Unicode to code page or from code page to Unicode or from one code page to another code page results in a Natural error or not.</p> <p>This parameter is not regarded for the conversion of Natural sources when loading them into the source area or when cataloging them.</p> <p>On mainframe platforms, it is not regarded whether a Unicode field is converted into the code page before an I/O on a terminal emulation. In this case, the substitution character defined by ICU is replaced by the place holder character which is defined in NATCONFIG.</p> |
| CPOBJIN | <p>Specifies the code page in which the batch input file for data is encoded. This file is defined with the Natural profile parameter CMOBJIN (Windows, UNIX and OpenVMS) or in the dataset CMOBJIN (mainframe).</p> |

| Parameter | Description |
|-----------|---|
| CPPRINT | Specifies the code page in which the batch output file shall be encoded. This file is defined with the Natural profile parameter CMPRINT (Windows, UNIX and OpenVMS) or in the dataset CMPRINT (mainframe). |
| CPSYNIN | Specifies the code page in which the batch input file for commands is encoded. This file is defined with the Natural profile parameter CMSYNIN (Windows, UNIX and OpenVMS) or in the dataset CMSYNIN (mainframe). |
| SRETAIN | Specifies that all existing sources have to be saved in their original encoding format. See also Customizing Your Environment . |

See also:

- *Code Pages for the Input and Output Files* in the section *Natural in Batch Mode* of the *Operations* documentation for Natural for Windows, Natural for UNIX and Natural for OpenVMS.
- *Natural in Batch Mode* in the *Operations* documentation for Natural for Mainframes.
- For valid code pages, see <http://www.iana.org/assignments/character-sets>.

Windows, UNIX and OpenVMS Platforms

The following profile parameters are only available on Windows, UNIX and OpenVMS platforms:

| Parameter | Description |
|-----------|---|
| SUTF8 | Specifies the default format to be used when Natural sources are saved. Note: On UNIX and OpenVMS, this parameter can only be used in a SPoD environment. |
| SUBCHAR | Specifies the substitution character for the conversion from Unicode to the default code page. If SUBCHAR is OFF, the default substitution character defined by ICU will be used. Note: SUBCHAR does not influence conversions from code page to Unicode or from Unicode to code pages which differ from the default code page. |
| WEBIO | Specifies whether the Natural Web I/O Interface client (which supports Unicode) or the terminal emulation window (which is not Unicode-enabled) is used for input and output. In a local Windows environment, the output window (which is Unicode-enabled) is used. In a remote Windows environment, the Natural Web I/O Interface client is always used, regardless of the setting of this parameter. Note: For mainframe platforms, the NDV configuration parameter <code>TERMINAL_EMULATION</code> is used instead. See below. |

Mainframe Platforms

The following profile parameters and/or macros are only available on mainframe platforms:

| Parameter | Macro | Description |
|----------------|--|--|
| CFICU | NTCFICU | Enables Unicode support for various Unicode settings. |
| Not available. | NTCPAGE | In the NATCONFIG module, this macro defines a code page and all related information, such as replacement characters, locale ID and collation tables. |
| PRINT | CP keyword subparameter of NTPRINT macro | Defines the code page for a report. |
| CMPO | CPAGE keyword subparameter of NTCMPO macro | Generates code page-sensitive Natural programs. |
| OPRB | NTOPRB | Set the ACODE and/or WCODE option to define the user encoding if the used Adabas database is enabled for UES (universal encoding support). |

Natural Development Server for Mainframes

The following NDV configuration parameter is only available with the Natural Development Server for mainframe platforms:

| Settings | Description |
|--------------------------|--|
| TERMINAL_EMULATION=WEBIO | Specifies that the Natural Web I/O Interface client (which supports Unicode) is used for input and output. |

Encoding Information

The encoding of code page data can be specified on different levels:

- [Level 1 - Default Code Page](#)

- **Level 2 - Code Page for a Single Object**

Level 1 - Default Code Page

The default code page can be defined with the `CP` parameter. On Windows, UNIX and OpenVMS platforms, it overwrites the system code page and is valid for all code page data.

Level 2 - Code Page for a Single Object

A code page can be defined for Natural sources, batch input (`CPOBJIN`, `CPSYNIN`) and output files (`CPPRINT`).

In addition, on Windows, UNIX and OpenVMS platforms, a code page can be defined for work files of type ASCII, ASCII compressed, Unformatted and CSV; see *Work File Assignments* in the *Configuration Utility* documentation.

If a code page is defined at object level, this overwrites the default code page.



Note: On Windows, UNIX and OpenVMS platforms, it is important that the correct code page is defined for every object. For more information, see [Migrating Existing Applications](#).

Deploying Natural Objects with Encoding Information

Windows, UNIX and OpenVMS Platforms

If you want to deploy Natural objects for which encoding information has already been defined, you have to keep in mind that the encoding information is stored in the file `FILEDIR.SAG` and that it is lost if you deploy only the object file from outside of Natural.

When deploying Natural objects, you have the following possibilities for keeping the encoding information:

- You can copy the entire library. The copy of the library can then be distributed to all Windows, UNIX and OpenVMS platforms. In this case, the original code page is kept. If a library is copied from Windows to UNIX or OpenVMS, you have to keep in mind that it may be possible that the objects cannot be opened with a native Natural for UNIX or Natural for OpenVMS editor because these editors can only open objects with the default code page.
- You can use the Object Handler which keeps the encoding information. In this case, the original code page is kept. If a Windows library is unloaded on UNIX or OpenVMS, you have to keep in mind that it may be possible that the objects cannot be opened with a native Natural for UNIX or Natural for OpenVMS editor because these editors can only open objects with the default code page.

- You can copy and paste objects with Natural Studio. In a SPoD environment, if the target environment is located on a platform different from the source environment, Natural tries to save the object with the default code page of the target environment. If this is not possible, the object is stored in UTF-8 format. For UNIX and OpenVMS targets, this assures that the object can be opened with the native Natural for UNIX or Natural for OpenVMS editors, if all characters of the source are available in the default code page of the UNIX or OpenVMS server.

Mainframe Platforms

For objects on mainframe platforms, there are no special considerations for keeping the code page information of the object since it is part of the object directory.

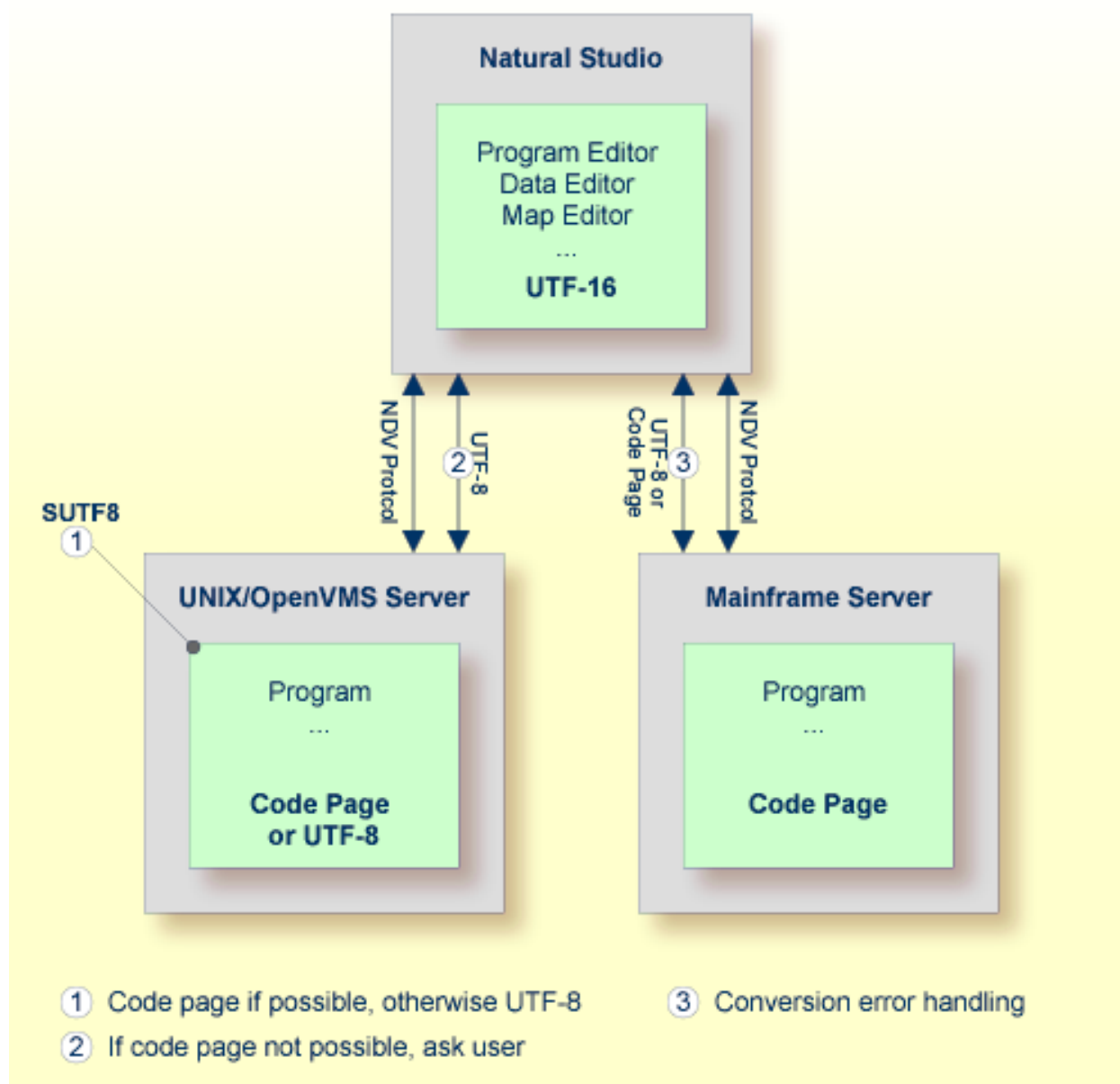
5

Development Environment

| | |
|---|----|
| ■ Development Environment | 28 |
| ■ Customizing Your Environment | 29 |
| ■ Editors | 30 |
| ■ Code Page Support for Editors, System Commands and Utilities on the Mainframe | 32 |

Development Environment

The development environment for Unicode applications is Natural Single Point of Development (SPoD).



In a SPoD environment, the Natural objects of a Unicode application which are located on a Natural Development Server (NDV) can be modified using Natural Studio. If supported by the server, the sources are exchanged between client and server in UTF-8 format.

On NDV servers for UNIX and OpenVMS, the setting of the profile parameter `SUTF8` determines the format that is used when storing the Natural object on the server. This is handled just like the local Windows case.

On NDV servers for mainframes, the objects are stored with the default or their original encoding, depending on the setting of the profile parameter `SRETAIN`.

Customizing Your Environment

On Windows, UNIX and OpenVMS platforms, it is important that you define the correct default code page for your environment before changing any Natural code. For more information, see [Migrating Existing Applications](#).

If you want to store characters from different languages in your sources, you have to save the sources in UTF-8 format on Windows, UNIX or OpenVMS platforms, or you have to use hexadecimal UH constants in the sources. With the profile parameters `SUTF8` and `SRETAIN` you can control in which format sources are saved. The following table lists some situations and the recommended settings.



Note: On UNIX and OpenVMS, the parameter `SUTF8` can only be used in a SPoD environment.

| Situation | Settings | Effect |
|---|---|---|
| Sources are located on Windows; U constants are needed. | <code>SUTF8=ON</code> , <code>SRETAIN=OFF</code> | All sources are saved in UTF-8 format when saving them with Natural 6.2 or above. New sources are created in UTF-8 format. All characters can be stored in a source. |
| Sources are located on Windows, UNIX and/or OpenVMS; U constants are needed and SPoD is used for development. | <code>SUTF8=ON</code> , <code>SRETAIN=ON</code> | All sources are saved in UTF-8 format when a conversion to the original code page is no longer possible; if it is possible, the code page of a source will not be changed. New sources are created in UTF-8 format. All characters can be stored in a source. A source with UTF-8 format can only be changed with SPoD; it can no longer be handled with the Natural for UNIX or Natural for OpenVMS editors. |
| Sources are located on Windows; UNIX and/or OpenVMS; no U constants are needed. | <code>SUTF8=OFF</code> , <code>SRETAIN=ON</code> | All sources are saved with the original code page. New sources are saved with the default code page (of server). Only characters from the source code page can be stored in a source. The sources can further be handled with the Natural for UNIX or Natural for OpenVMS editors. |

| Situation | Settings | Effect |
|--|---------------------------|---|
| Sources are located on Windows, UNIX, OpenVMS and/or mainframe; U constants are needed and SPoD is used for development. | SUTF8=OFF , SRETAIN=ON | All sources are saved with the original code page. New sources are saved with the default code page (of the server). Only characters from the source code page can be stored in a source. The sources can further be handled with the Natural for UNIX, Natural for OpenVMS and Natural for Mainframes editors. All Unicode constants have to be defined as hexadecimal constants (UH). |

If the parameter `SUTF8` is set to `OFF` and you store a source which contains characters from different character sets, but which was not yet saved in UTF-8 format, it is possible that the generated program is created, but that the source cannot be saved and thus remains unchanged. This happens if characters from different character sets are used in a comment or in a U constant. For this reason, it is recommended that you set the parameter `SUTF8` to `ON` if you want to create sources with characters from different character sets and if your sources do not need to be distributed to mainframe platforms.

If the parameter `SRETAIN` is set to `OFF`, all sources are saved with the default code page. You have to be careful with this setting because it may lead to improper code page information if you have sources which were created with an earlier Natural version. In this case, the encoding information of the source is unassigned and the source is always opened with the default code page (value of the system variable `*CODEPAGE`). This will often work even if the default code page is not the correct encoding of the source. Some language-specific characters will be displayed incorrectly in this case. If such a source is opened with the wrong code page and is saved with `SRETAIN` being set to `ON`, no encoding will be stored for the source; the source can later be opened correctly if Natural is started with the correct default code page. However, once you have saved the source with `SRETAIN` being set to `OFF`, the default code page will be saved as the encoding of the source; from this time on, the source will only be opened with this code page. For this reason, you should use this setting only if you are certain that all of your Natural sources are encoded in the default code page.

See also: *Regional Settings* in the *Configuration Utility* documentation.

Editors

The Natural for Windows editors are fully Unicode-enabled. Via SPoD they can also be used for mainframe, UNIX and OpenVMS sources. The editors provided with Natural for Mainframes, Natural for UNIX and Natural for OpenVMS are not Unicode-enabled.



Note: The editors provided with Natural for Mainframes provide code page support. See [Code Page Support for Editors, System Commands and Utilities on the Mainframe](#).

When a source is opened with an editor in Natural Studio (Natural for Windows), the content of the source will be converted from the corresponding code page to Unicode before it is loaded into

the editor. This will guarantee that all characters can be displayed correctly even if the source contains characters which are not included in the system code page. If the conversion from the source's code page to Unicode fails, an error will be displayed and the editor is not opened. In this case, the user has to define the correct encoding of the source. The source encoding can be changed in the **Properties** dialog box (see *Properties for the Nodes* in the *Using Natural Studio* documentation).

For Windows, UNIX and OpenVMS sources, the Natural for Windows editors allow saving sources which contain characters from different languages in UTF-8 format. On mainframes, it will not be possible to save UTF-8 sources.



Note: If you save a UNIX or OpenVMS source in UTF-8 format or with a code page which differs from the default code page, the source can no longer be opened with the native Natural for UNIX or Natural for OpenVMS editor. Mainframe sources can be saved with a different code page and can be edited with the native Natural for Mainframes editors.

Even if you do not want to use Unicode strings in your programs and sources, the Unicode-enabled editors have the advantage that you can write sources in all code pages, no matter which system code page is installed. For example, if you have installed the "windows-1252" (Latin 1) code page, you can write a program containing Cyrillic characters and save this program with the "windows-1251" (Cyrillic) code page. You only have to select code page "windows-1251" in the **Save As** dialog box (see *Saving an Object with a New Name* in the *Using Natural Studio* documentation).

Using the Natural for Windows program editor, you can convert text constants into their hexadecimal Unicode representations (see *Converting to Hexadecimal Format* in the *Program Editor* section of the *Natural for Windows Editors* documentation). If you are developing for a platform where UTF-8 sources are not preferred, you can thus enter all characters for a Unicode constant, select all the characters of the constant, convert them to their hexadecimal representation and then add the "UH" prefix for Unicode hexadecimal constants. Furthermore, when you hover the mouse pointer over a character or a selected character range of a text constant, a tool tip shows the corresponding hexadecimal Unicode representation.

A byte order mark (BOM) consists of the character code "U+FEFF" at the beginning of a data stream where it can be used as a signature defining the byte order and encoding form, primarily of unmarked plain-text files. On Windows, a byte order mark is used by some editors (for example, Notepad) to mark UTF-8 files. The Natural for Windows editors will recognize an UTF-8 byte order mark when reading an object. If the object has no other encoding defined so far, Natural will interpret it as UTF-8 and when the object is saved, UTF-8 will be stored as the encoding for the object. The byte order mark is removed in this case.

Code Page Support for Editors, System Commands and Utilities on the Mainframe

The following topics are covered below:

- [Editors](#)
- [System Commands and Utilities](#)

Editors

The program, map and data area editors provided with Natural for Mainframes are not Unicode-enabled. Instead the sources are stored with code page information. According to the setting of the profile parameter `SRETAIN`, Natural sources with code page information may be converted automatically from the current code page of the source into the default code page of the current Natural session (value of the system variable `*CODEPAGE`) if the source is loaded into the editor. If there are any characters that cannot be converted, a window displays a code point conversion error and asks for substitute values for those code points that cannot be converted. The display of this message is independent from the current setting of the parameter `CPCVERR`. In this case, the user can decide to open the editor with or without converting the source into the default code page. Saving or stowing a converted source will save the new code page information. Sources without code page information (for example, sources that have been saved or stowed with previous Natural versions) are loaded into the editors without any conversion. According to the setting of the profile parameter `SRETAIN`, the current code page information of the source will be retained.

Inserting sources with the `.I` command or the split screen function will also convert sources, if necessary, according to the setting of the profile parameter `SRETAIN`. If characters cannot be converted, the defined substitution character will be inserted instead.

The check and conversion of the source is performed when the editor is started, not when the program is loaded into the source area. If a program is executed via `RUN program-name`, a conversion is not performed. This causes different behavior, depending on whether `RUN program-name` is entered on the `NEXT` screen or on an editor screen. If `RUN program-name` is entered on the `NEXT` screen, no conversion follows; if it is entered on an editor screen, the editor is started right after the execution of the program and a conversion is performed.

See the table below for the code page that is assigned to an existing Natural source that is saved or stowed, depending on the values of the profile parameters `SRETAIN` and `CP`.

| Original Source Code Page Information | Setting of <code>SRETAIN</code> | Source Code Page Information after <code>SAVE</code> or <code>STOW</code> if <code>CP</code> is Set to a Value other than <code>OFF</code> | Source Code Page Information after <code>SAVE</code> or <code>STOW</code> if <code>CP</code> is set to <code>OFF</code> |
|---|---|--|---|
| Source without code page information | <code>SRETAIN=ON</code> <code>SRETAIN=(ON, EXCEPTNEW)</code> | No code page information | No code page information |
| Source without code page information | <code>SRETAIN=OFF</code> | Code page resulting from evaluation of <code>CP</code> | No code page information |
| Source is encoded in <i>code page 1</i> | <code>SRETAIN=ON</code> <code>SRETAIN=(ON, EXCEPTNEW)</code> | Original code page (<i>code page 1</i>) | Original code page (<i>code page 1</i>) |
| Source is encoded in <i>code page 1</i> | <code>SRETAIN=OFF</code> | Code page resulting from evaluation of <code>CP</code> | Original code page (<i>code page 1</i>) |

The table below shows the code page that is assigned to a new Natural source that is saved or stowed, depending on the values of the profile parameters `SRETAIN` and `CP`.

| Setting of <code>SRETAIN</code> | Source Code Page Information after <code>SAVE</code> or <code>STOW</code> if <code>CP</code> is Set to a Value other than <code>OFF</code> | Source Code Page Information after <code>SAVE</code> or <code>STOW</code> if <code>CP</code> is set to <code>OFF</code> |
|--------------------------------------|--|---|
| <code>SRETAIN=ON</code> | Code page resulting from evaluation of <code>CP</code> | No code page information |
| <code>SRETAIN=OFF</code> | Code page resulting from evaluation of <code>CP</code> | No code page information |
| <code>SRETAIN=(ON, EXCEPTNEW)</code> | No code page information | No code page information |

System Commands and Utilities

LIST

By default, the system command `LIST` displays sources as they are stored in the system file without any conversions.

The `CONVERTED` option of the `LIST` command converts the source into the default code page (value of the system variable `*CODEPAGE`) if the code page information of the source is provided. All non-convertible characters are then replaced by the defined substitution character.

LIST DIR

The system command `LIST DIR` shows the used code page information of a Natural source in the directory window.

SCAN

Similar to the editors, the system command `SCAN` converts the sources before executing the actual `SCAN` command.

Object Handler (SYSOBJH)

The Object Handler unloads and loads sources with different code page information and preserves the original code page information.

The transfer format option UTF-8 converts sources from any code page to UTF-8 format while unloading, and stores information about the original code page in the work file. The corresponding load function converts the source back to the original code page or to another code page, if specified. This option can also be used to provide code page information for sources which have been saved or stowed with previous Natural versions and which therefore do not contain any code page information.

Unload and load sources in internal format will keep the code page information, if available.

SYSCP Utility - Code Page Administration

The `SYSCP` utility can be used to obtain information on code pages and to check or change the code page assignment of a source.

6 Unicode Input/Output Handling in Natural Applications

| | |
|--|----|
| ■ Displaying and Entering Unicode Data | 36 |
| ■ Natural Web I/O Interface Client | 37 |

Displaying and Entering Unicode Data

If you want to display or enter Unicode data, the following possibilities exist:

- When working in the local development environment with Natural for Windows, all Unicode characters can be displayed and entered in the Natural output window.
- When working in a remote development environment with Natural for Windows (SPoD), the Natural Web I/O Interface client (see [below](#)) is necessary for displaying and entering all Unicode characters.
- When running applications with Natural for UNIX, Natural for OpenVMS, Natural for Mainframes or Natural for Windows, see [Natural Web I/O Interface Client](#) below.



Notes:

1. Even if you are working with a Unicode-enabled output interface on Windows, you will see only the Unicode characters which are supported by the currently selected font.
2. Unicode data cannot be displayed on 3270 terminals.

If you run Natural via a terminal emulation or a mainframe terminal like 3270/3279, the page will be converted to the default code page (value of the system variable `*CODEPAGE`) before displaying it, so that all characters which are not contained in the default code page are replaced with the substitution character. Equally, input is only possible in code page format and will be converted to Unicode format before assigning it to a U format field. You have to regard that the substitution character is defined by the ICU conversion tables. Depending on this character, it is possible that garbage is displayed with a terminal emulation. On UNIX and OpenVMS platforms, you can change this substitution character by setting the profile parameter `SUBCHAR`. However, it is strongly recommended that you use the Natural Web I/O Interface when displaying characters not contained in the default code page. When running a remote Windows session, the Natural Web I/O Interface will be used in any case.

On code page oriented mainframe terminals, it is important to select the suitable code page. The default code page of Natural, the code page of the terminal and even the font used by the terminal determine the capability of displaying certain characters correctly.

Natural Web I/O Interface Client

The Natural Web I/O Interface client is used to display non-GUI information which contains Unicode characters. It can be used in the following environments:

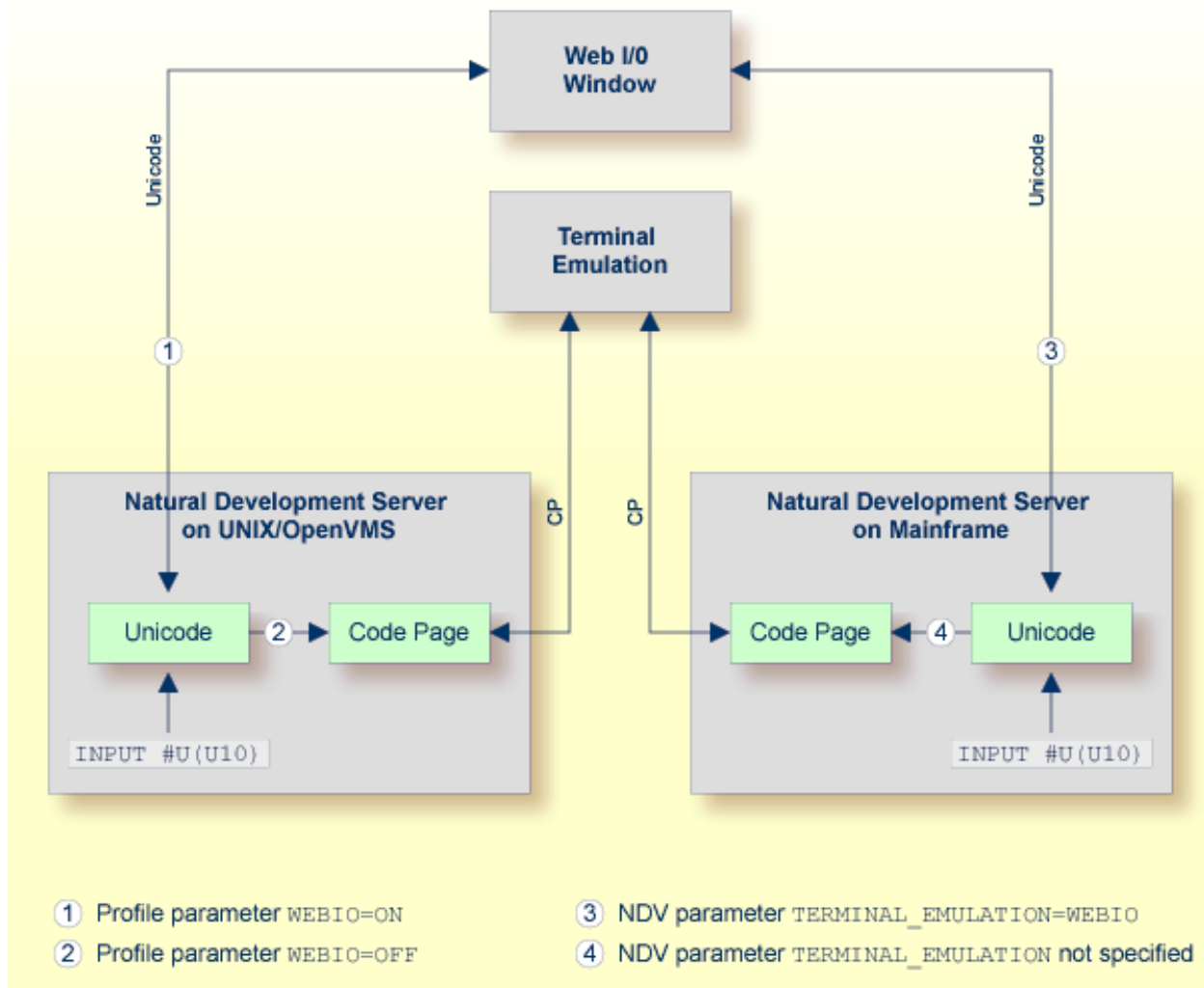
- [SPoD Environment](#)
- [Runtime Environment](#)

SPoD Environment

The Natural Web I/O Interface client can be invoked when you use Natural for Windows and you are working with Natural Studio in a remote development environment (SPoD); see *Natural Web I/O Interface Client* in *Remote Development Using SPoD* which is part of the Natural for Windows documentation.

When the Natural Web I/O Interface client is used, the Web I/O windows appears instead of the terminal emulation window which is not Unicode-enabled in remote UNIX, OpenVMS or mainframe environments, or instead of the output window in remote Windows environments.

The following graphic shows the SPoD environment for Unicode applications with Natural Development Servers (NDV) on UNIX, OpenVMS and mainframes:



So that the Natural Web I/O Interface client can be invoked, the Natural Development Server has to be configured as follows:

■ UNIX and OpenVMS

If you want to use the Natural Web I/O Interface client in a remote UNIX or OpenVMS environment, the profile parameter `WEBIO` must be set to `ON` on the NDV server. See *Configuration Utility* in the Natural for UNIX or Natural for OpenVMS documentation.

■ Mainframe

If you want to use the Natural Web I/O Interface client in a remote mainframe environment, the NDV configuration parameter `TERMINAL_EMULATION` must be set to `WEBIO` on the NDV server. See *NDV Configuration Parameters* in the Natural Development Server documentation. Moreover, the Web I/O terminal converter module `NATWEB` must be linked to the Natural nucleus. The Natural profile parameter `TMODEL` can be used to determine the user screen size.

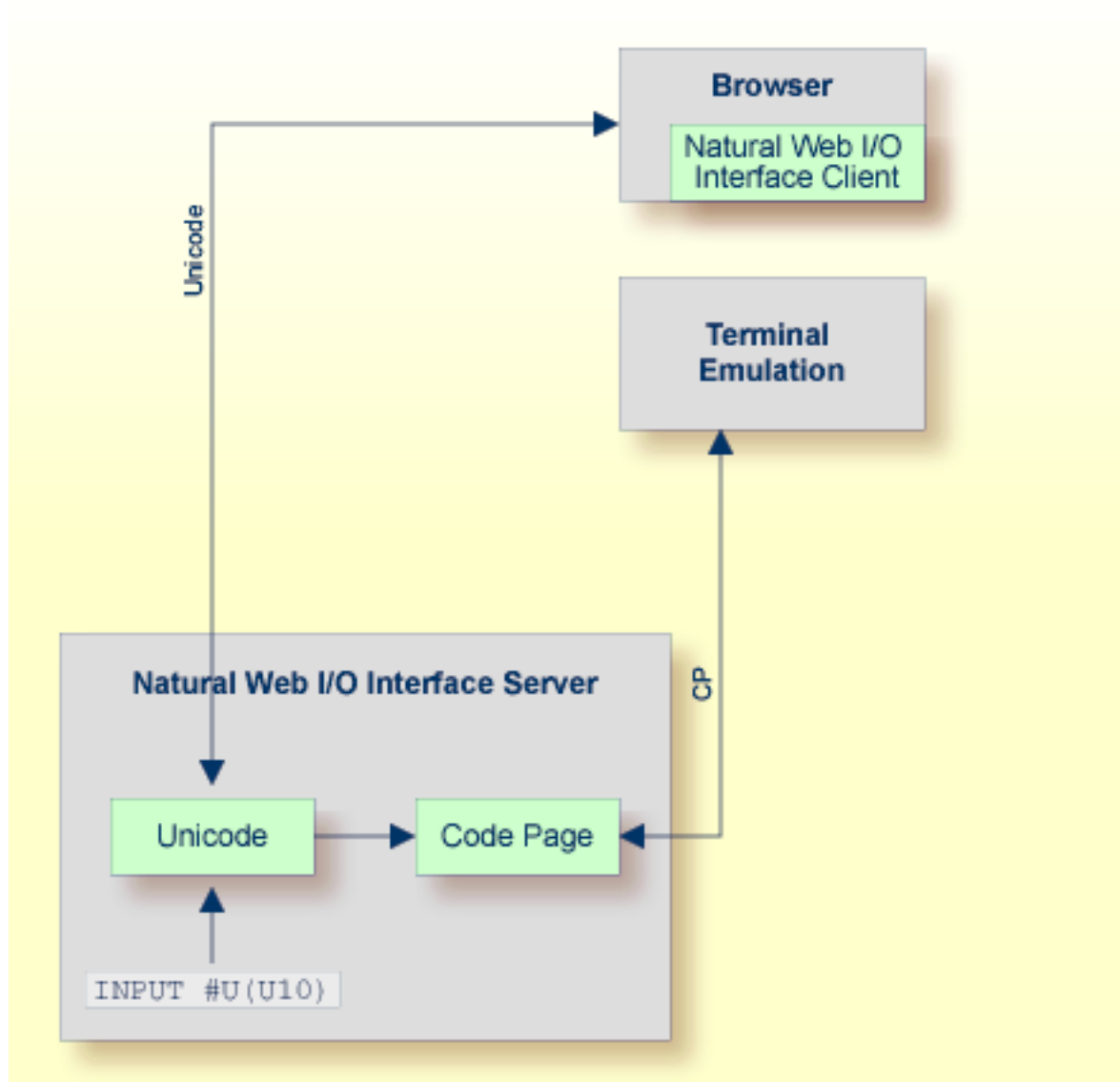
■ Windows

In a remote Windows environment, the Natural Web I/O Interface client is always used, regardless of the setting of the profile parameter `WEBIO`.

Runtime Environment

The Natural Web I/O Interface client appears when running applications with Natural for UNIX, Natural for OpenVMS, Natural for Mainframes or Natural for Windows. It runs in a web/application server.

The following graphic shows the runtime environment for Unicode applications:



Natural recognizes automatically whether the session has been started from the Natural Web I/O Interface client or from the terminal emulation.

Prerequisites for using the Natural Web I/O Interface client:

- **Natural for UNIX and Natural for OpenVMS**

It is required that the Natural Web I/O Interface server (which is implemented as a daemon) has been installed and activated. See *Natural Web I/O Interface* in the Natural for UNIX and in the Natural for OpenVMS documentation.

- **Natural for Mainframes**

It is required that the Natural Web I/O Interface server has been installed and configured. See *Natural Web I/O Interface* in the Natural for Mainframes documentation. Moreover, the Web I/O terminal converter module NATWEB must be linked to the Natural nucleus. The Natural profile parameter TMODEL can be used to determine the user screen size.

- **Natural for Windows**

It is required that the Natural Web I/O Interface server (which is implemented as a service) has been installed and activated. See *Natural Web I/O Interface* in the Natural for Windows documentation.

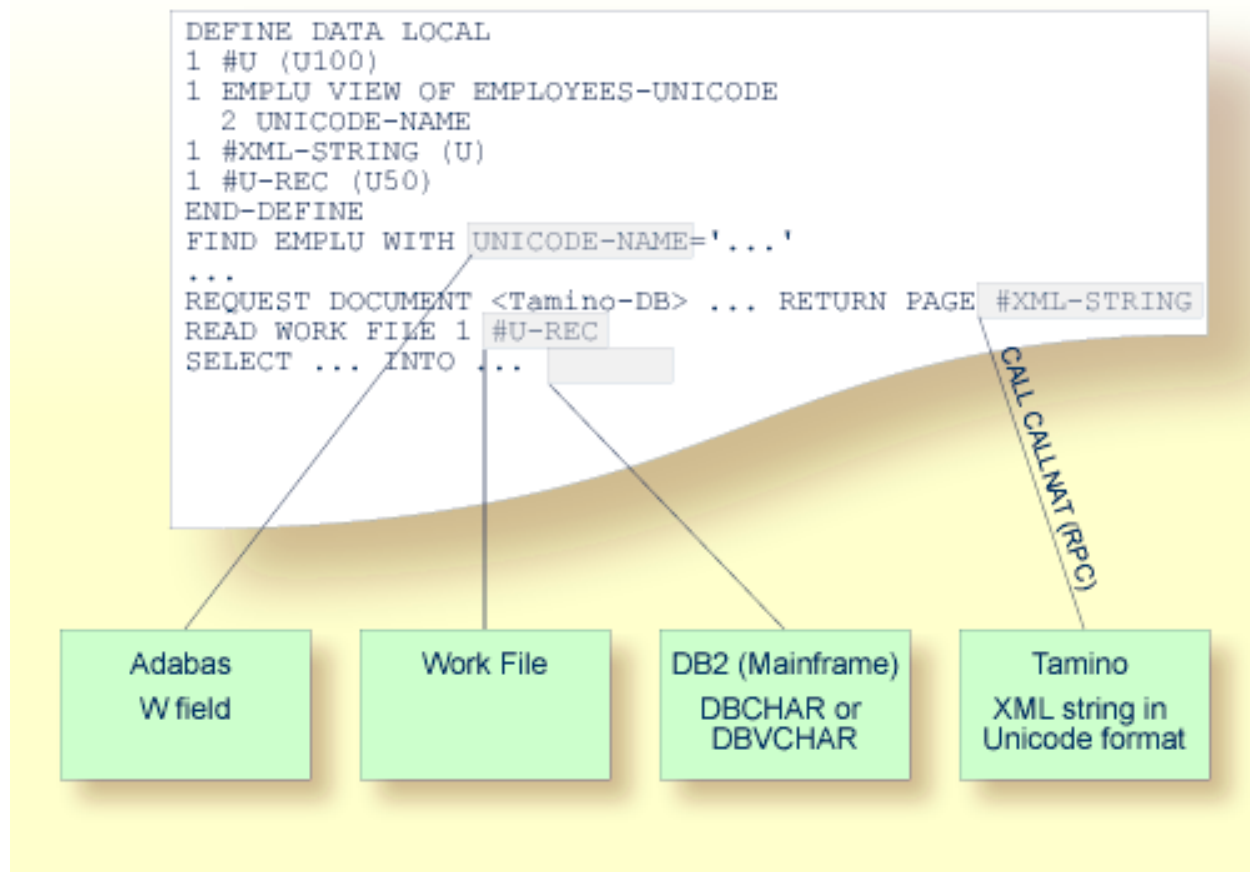
7

Unicode Data Storage

| | |
|---|----|
| ■ Unicode Data/Parameter Access | 42 |
| ■ Database Management System Interfaces | 42 |
| ■ Work Files and Print Files on Windows, UNIX and OpenVMS Platforms | 43 |
| ■ Work Files and Print Files on Mainframe Platforms | 47 |

Unicode Data/Parameter Access

The following graphic shows how Unicode data and parameters are accessed.



Database Management System Interfaces

The following topics are covered below:

- [Accessing Unicode Data in an Adabas Database](#)

- [Accessing Unicode Data in a DB2 Database](#)

Accessing Unicode Data in an Adabas Database

Natural enables users to access wide-character fields (format W) in an Adabas database.

Data Definition Module

Adabas wide-character fields (W) are mapped to the Natural data format U (Unicode).

Access Configuration

Natural receives data from Adabas and sends data to Adabas using UTF-16 as common encoding.

This encoding is specified with the `OPRB` parameter and is sent to Adabas with the open request. It is used for wide-character fields and applies to the entire Adabas user session.

For detailed information, see *Unicode Data* in the *Accessing Data in an Adabas Database* part of the *Programming Guide*.

Accessing Unicode Data in a DB2 Database

Natural enables users to access `CHAR` and/or `WCHAR` fields in a DB2 database as Unicode data.

See also *Natural for DB2* in the *Database Management System Interfaces* documentation.

Work Files and Print Files on Windows, UNIX and OpenVMS Platforms

The following topics are covered below:

- [WRITE WORK FILE](#)
- [READ WORK FILE](#)
- [Special Considerations for Work File Type Transfer](#)
- [Print Files](#)

WRITE WORK FILE

The information below applies for the statement `WRITE WORK FILE`. See the *Statements* documentation for detailed information on this statement.

Code Page Data

The following work file types write code page data:

- ASCII and ASCII compressed
- Unformatted

- CSV
- Entire Connection

The work file type and the code page must be defined in the Configuration Utility. For further information, see *Work Files* in the *Configuration Utility* documentation.

All Natural data defined with the operands A (alphanumeric) and U (Unicode) are converted to the specified code page. If a code page has not been specified, all data are converted to the default code page which is defined with the CP parameter.



Note: In the work file, all written A and U operand data are in code page format.

If U operand data are to be written into these work files and afterwards read from these work files without loss of data, you have to define UTF-8 as the code page (in the Configuration Utility). In this case, all A and U operand data are written in UTF-8 format. A subsequent `READ WORK FILE` statement where the work file is also configured using code page UTF-8 reads the operand U data without loss of data.



Notes:

1. Work file data which have been written in UTF-8 format can be read by text editors which support UTF-8 (for example, Notepad on the Windows platform).
2. Natural data defined with the operand B (binary) are not converted to the code page which has been specified in the Configuration Utility. These data are written as they are stored in Natural, without any code page conversion.

If one of the above-mentioned work file types is specified and the code page UTF-8 is defined for the work file, the work file attributes BOM (write byte order mark) and NOBOM (do not write byte order mark) take effect. These attributes can be specified in the **Work Files** category of the Configuration Utility and with the `DEFINE WORK FILE` statement. If the code page UTF-8 is defined for the work file and the work file attribute BOM is specified, the UTF-8 byte order mark (hexadecimal representation: H'EFBBBF') is written at the beginning of the work file, in front of the work file data.

If a work file type other than the above-mentioned work file types is used for writing the work file, or if a code page other than UTF-8 is defined for the work file, the specification of the attribute BOM is ignored during runtime. The following table shows the runtime behavior during the processing of the statements `WRITE WORK FILE` and `READ WORK FILE`:

| Code Page and Attribute Setting | WRITE WORK FILE | READ WORK FILE |
|--|---|--|
| <p>The code page UTF-8 is not specified for the work file (default).</p> <p>The work file attributes BOM and NOBOM have no effect.</p> | <p>No UTF-8 byte order mark is written.</p> <p>No conversion to UTF-8.</p> | <p>No check for UTF-8 byte order mark.</p> <p>No conversion from UTF-8.</p> |
| <p>The code page UTF-8 is specified for the work file.</p> <p>The work file attribute BOM is specified.</p> | <p>UTF-8 byte order mark is written.</p> <p>A and U fields are converted to UTF-8.</p> | <p>Check for UTF-8 byte order mark.</p> <p>If an UTF-8 byte order mark is found, it is removed from the work file data. A fields are converted from UTF-8 to the default code page. U fields are converted from UTF-8 to the Natural internal runtime representation UTF-16.</p> |
| <p>The code page UTF-8 is specified for the work file.</p> <p>The work file attribute NOBOM (default) is specified.</p> | <p>No UTF-8 byte order mark is written.</p> <p>A and U fields are converted to UTF-8.</p> | <p>Check for UTF-8 byte order mark.</p> <p>If an UTF-8 byte order mark is found, it is removed from the work file data. A fields are converted from UTF-8 to the default code page. U fields are converted from UTF-8 to the Natural internal runtime representation UTF-16.</p> |

Binary Data

The following work file types write binary data (for example, UTF-16 for operand format U):

- SAG
- Portable

Natural data defined with the operands A and U are not converted to code page. These data are written to the work file in binary format. For U operand data, this is done in UTF-16.

READ WORK FILE

The information below applies for the statement `READ WORK FILE`. See the *Statements* documentation for detailed information on this statement. Take note of the restrictions that are listed for the `RECORD` option (in the Natural for Windows, Natural for UNIX and Natural for OpenVMS documentation).

Code Page Data

When the following work file types are used, the work file data that are read into Natural U (Unicode) operands are converted from the specified code page to UTF-16.

- ASCII and ASCII compressed
- Unformatted

- CSV
- Entire Connection

Data that are read into A (alphanumeric) operands are converted, if required, from the specified code page to the default code page which has been defined with the parameter `CP`.

If one of the above-mentioned work file types is specified and the code page UTF-8 is defined for the work file, the `READ WORK FILE` statement automatically checks the work file for an UTF-8 byte order mark. If an UTF-8 byte order mark is found at the beginning of the work file, it is removed. The data that are read from the work file are converted from UTF-8 to the default code page.

If data are read from another work file type, the check for a byte order mark is not performed and a byte order mark is therefore not removed.

For information on the runtime behavior during the processing of the statements `WRITE WORK FILE` and `READ WORK FILE`, see the table in the [previous](#) section.

Binary Data

When the following work file types are used, the work file data are read into the Natural operands A and U without conversion (that is: they are read in binary format):

- SAG
- Portable

The work file type Portable supports endian conversion for data of operand format U.

Special Considerations for Work File Type Transfer

Operand format U is generally supported for the work file type Transfer. If Entire Connection is not able to read or write Unicode for the selected file type, a runtime error message is displayed.

Print Files

The handling for Unicode data in print files depends on the selected logical device's (LPT1 to LPT31) print method, currently either GUI (Windows only) or TTY.

Regardless of the print method, data are passed to the Natural printing services in UTF-16 format. That is, any format A field data will already have been converted to Unicode.

GUI Print Method

With this Windows-only print method, the data are passed to the Windows printer driver in Unicode (UTF-16) format. Because this is the standard method for printing data in Windows, the driver invariably handles this data appropriately. This is therefore the recommended print method under Windows if any characters that are not within the system code page are being used.

TTY Print Method

With this print method, the data are, by default, converted from the internal (UTF-16) format into the system code page. However, by using a printer profile, it is possible to specify that the data should instead either be converted into UTF-8 format, or be subjected to an additional conversion to an arbitrary external code page. For more information on these alternatives, see *Printer Profiles* in the *Configuration Utility* documentation.

The rationale behind the default behavior of converting the data into the system code page is based on the current lack of printers capable of directly accepting raw text files in UTF-8 format.

Work Files and Print Files on Mainframe Platforms

The following topics are covered below:

- [Work Files](#)
- [Print Files](#)

Work Files

No special consideration is given to Unicode data when writing or reading work files. Like all other data types, Unicode data is written and read as is, without conversion.

Print Files

When sending Unicode data to print files, one or two conversion steps take place.

In a first step, Unicode data contained in a print line is converted to the default code page of the session. As a consequence, all characters which are not contained in this default code page are replaced with the substitution character.

Before passing this converted print line to the actual print access method, it is additionally checked whether a code page has been specified for the logical printer. This may have been accomplished with the `CODEPAGE` operand of the `DEFINE PRINTER` statement or the `CP` subparameter of the `PRINT` parameter. If such a code page has been given, the whole print line (not only the Unicode part of it) is converted accordingly in a second step.

The converted print line is passed to the access method, which means that print access methods do *not* receive Unicode data.

Example:

```
DEFINE PRINTER (1) CODEPAGE 'IBM01140'  
WRITE (1) 'HELLO' U'WORLD'  
END
```

8

Platform Differences

| | |
|---|----|
| ■ Windows, UNIX and OpenVMS Platforms | 50 |
| ■ Mainframe Platforms | 51 |

Windows, UNIX and OpenVMS Platforms

On Windows, UNIX and OpenVMS platforms, Natural has internally been Unicode-enabled. This means that many structures containing strings have Unicode format now. For example, the Natural source area has now Unicode format. For this reason, Unicode data can be handled at runtime in the Natural I/O as well as in the Natural development environment when writing and cataloging Natural code.

For the first version, there are some exceptions: the Natural dialogs (editor and runtime) are not Unicode-enabled. These modules will be Unicode-enabled in a later version.

Even if Natural is Unicode-enabled internally, all existing data currently has code page format. As a consequence, all this data is converted from code page format to Unicode format when used in Natural Version 6.2 or above. For example, if a source is opened with the program editor, a conversion from the code page file format to the Unicode source area format is performed. Even if you do not use the U format, this is of advantage: you can now see all language-specific characters, no matter which system code page is installed. However, the user is responsible for defining the correct code page information. See [Migrating Existing Applications](#) for more details.

When cataloging Natural objects, all constants which are not defined with the U prefix are converted to the code page of the corresponding source. If the source has UTF-8 format, these constants are converted to the default code page.



Note: In most cases, Unicode data requires more memory space than code page data. Therefore, the Natural parameter `USIZE` may need to be increased with Natural Version 6.2 or above.

Windows

Unicode is fully supported in the local Natural for Windows environment.

The editors are Unicode-enabled and it is possible to enter all possible characters. When saving the source, Natural first tries to convert the source to the original code page. If this fails because the source contains characters which are not found in this code page, further processing depends on the setting of the parameter `SUTF8`. If `SUTF8` is `ON`, the source will be saved in UTF-8 format. If `SUTF8` is `OFF`, the user will be asked whether to save the source in the original code page or to cancel the current save. If the user decides to save the source in the original code page, the characters which are not found will be replaced with substitution characters. In addition, it is possible to select a code page explicitly in the **Save As** dialog box.

The program editor has been enhanced in order to support the Unicode bidirectional algorithm.

The output window is also Unicode-enabled. When characters are entered via the keyboard, A format fields accept only the characters which are available in the default code page.

UNIX and OpenVMS

Full Unicode support is only available with SPoD and the Natural Web I/O Interface. SPoD is necessary for entering Unicode input in Natural sources; the same applies as described above for the local Natural for Windows environment. The Natural Web I/O Interface is necessary for Unicode I/O from Natural applications.

If Natural is used via a terminal emulation, all output will be converted from Unicode to the default code page before displaying it. Characters which are not available in the default code page will be replaced with the substitution character of the default code page. Similar input is only possible on base of the default code page.



Note: Natural sources which have UTF-8 format can no longer be opened with the native Natural for UNIX or Natural for OpenVMS editors.

Mainframe Platforms

The Natural runtime environment is enabled for Unicode support. Unicode characters are converted to the default code page (value of the system variable `*CODEPAGE`) before they are displayed on the terminal. Unicode characters which have no equivalent in the default code page are replaced by a substitution character.

With the Natural Web I/O Interface under SPoD, Unicode characters are fully supported by the terminal emulation. In this case, U format fields are displayed and can be entered correctly as Unicode. They are not converted to the equivalent in the default code page. The Natural Web I/O Interface is activated by the NDV server configuration parameter `TERMINAL_EMULATION=WEBIO`. The system variable `*DEVICE` contains `BROWSER`.

The Natural compiler, the editors and the Natural system file do not support object sources that are encoded in Unicode. Unicode constants coded in an object source are saved in the default code page, and the cataloged object contains the Unicode code points. The only way to define Unicode constants which do not have an equivalent in the default code page is to use hexadecimal definitions (UH).

Code page conversion and Unicode support make use of functionality provided by the ICU library. The size of the ICU modules providing this functionality depends on the used ICU functionality. If neither code page conversion nor Unicode support are required, these modules do not have to be linked to the Natural nucleus. For improved flexibility, it is also possible to link these modules dynamically to the Natural nucleus during initialization of the Natural session. The use of ICU functionality increases the required Natural thread size.

The following topics are covered below:

- [NATICU Modules for Different Purposes](#)
- [Session Modes](#)

- CFICU Parameter
- Shared FUSER
- CPAGE Compiler Option
- Program Sources
- NTCPAGE Macro
- Unicode and Code Page Support for Databases
- Translation Tables
- Support of Multi-Byte Code Pages
- ICU Buffer Pool

NATICU Modules for Different Purposes

To enable Natural for Unicode and code page support, an ICU library module has to be linked.

To execute only Natural applications that require neither Unicode nor code page support (profile parameters `CFICU=OFF` and `CP=OFF` are set), none of the supplied ICU modules needs to be linked.

Natural offers different implementations of the ICU library for different purposes:

■ NATICU

This implementation is intended to be used in most European countries as well as in north and south American countries. It contains a reduced set of code pages and locale IDs for English, German, French and Spanish language areas. Due to the reduced set of supported languages, it is relatively small.

Another feature of this module is collation services. Collation services are used to compare Unicode strings. They consider the fact that the alphabetical order varies from language to language. It is a big challenge to accommodate the world's languages and writing systems and the different orders that are used. However, the ICU collation service provides excellent means for comparing strings in a locale-sensitive fashion. For example, the character "Ä" is sorted in German locale between "A" and "B"; in Swedish locale, it is sorted after "Z". In Lithuanian, the character "y" is sorted between "i" and "k". The ICU implementation of collation services is compliant to the Unicode Collation Algorithm and conforms to ISO 14651. The algorithms have been designed and reviewed by experts in multilingual collation, and are therefore robust and comprehensive.

NATICU provides the code pages and locales listed below.

■ NATICUCV

This implementation is the same as NATICU, but without collation services since this is a very large package inside ICU. If NATICUCV is used, a binary comparison is performed for Unicode strings instead of the locale-dependant comparison of collation services. If all of your Unicode data result from the same code page and all Unicode data are normalized, then you can use this module.

NATICUCV provides the code pages and locales listed below.

■ NATICUXL

This implementation is intended to be used in all areas of the world that are not covered by the reduced amount of code pages and locale IDs of NATICU.

NATICUXL contains all code pages and locale IDs provided by the currently supported ICU version. For an overview of the supported code pages and local IDs, refer to the ICU homepage (see <http://demo.icu-project.org/icu-bin/convexp>).

You may either statically link an ICU library module to your Natural nucleus or dynamically load it during session initialization using the RCA technique via the session parameters RCA and RCALIAS.

To load NATICU:

```
RCA=NATICU
```

To load NATICUCV:

```
RCA=NATICU,RCALIAS=(NATICU,NATICUCV)
```

To load NATICUXL:

```
RCA=NATICU,RCALIAS=(NATICU,NATICUXL)
```



Note: You may dynamically load an ICU library module during session initialization even though an ICU library module is already statically linked to your Natural nucleus. In this case, the statically linked ICU library module is ignored and replaced by the use of the dynamically loaded ICU library module.

NATICU and NATICUCV provide the following code pages and locales:

| Code Pages | Locales |
|-------------------------|---------|
| IBM037 | de_DE |
| IBM273 | en_US |
| IBM1025 | es_ES |
| IBM1026 | fr_FR |
| IBM1047 | sv_SE |
| IBM1097 | |
| IBM01140 | |
| IBM01141 | |
| IBM01145 | |
| IBM01146 | |
| IBM01147 | |
| US (alias for IBM01140) | |
| DE (alias for IBM01141) | |
| ES (alias for IBM01145) | |

| Code Pages | Locales |
|---|---------|
| EN (alias for IBM01146) | |
| FR (alias for IBM01147) | |
| IBM-37_P100-1995,SWAPLFNL | |
| IBM-1047_P100-1995,SWAPLFNL | |
| IBM-1140_P100-1997,SWAPLFNL | |
| EBCDIC-XML-US | |
| EDF03DRV (Siemens code page) | |
| EDF03IRV (Siemens code page) | |
| EDF04DRV (Siemens code page) | |
| EDF04IRV (Siemens code page) | |
| IBM-290 (Japanese code page SBCS) | |
| IBM-930 (Japanese code page SBCS/DBCS) | |
| IBM-939 (Japanese code page SBCS/DBCS) | |
| IBM-1390 (Japanese code page SBCS/DBCS) | |
| IBM-1399 (Japanese code page SBCS/DBCS) | |
| IBM-932 (Japanese code page ASCII MBCS) | |
| IBM-942 (Japanese code page ASCII MBCS) | |
| IBM-943 (Japanese code page ASCII MBCS) | |
| EUC-JP (Japanese code page ASCII MBCS) | |
| IBM-420 (RTL code page) | |
| IBM-424 (RTL code page) | |
| IBM-916 (RTL code page) | |

Session Modes

The parameters `CFICU` and `CP` can be used to adjust Natural to specific purposes:

| Settings | Description |
|--|--|
| <code>CFICU=OFF</code> , <code>CP=OFF</code> | Compatibility mode. For running existing applications without Unicode and without code page support. Legacy translation tables are used for I/O translation. Compared with former versions, there is no significant increase in resource consumption (CPU time and buffer usage). This mode does not need ICU to be linked to the Natural nucleus. |
| <code>CFICU=ON</code> , <code>CP=OFF</code> | For new applications that are using Unicode and code page conversion (MOVE ENCODED) but not default code page support. Therefore, the system variable <code>*CODEPAGE</code> is empty. It is possible to use U format variables, but it is not possible to use, for example, <code>MOVE A TO U</code> , since this requires the default code page information. The error NAT3411 will be issued indicating that no default code page is available. |
| <code>CFICU=ON</code> , <code>CP=value</code> * | For new applications that are using full Unicode as well as code page support. |
| <code>CFICU=OFF</code> , <code>CP=value</code> * | This combination is possible, but does not make sense, because code page support needs ICU services for conversion. Therefore, <code>CFICU=ON</code> is enforced in this case and a session initialization message is issued. |

* where *value* is any value other than `OFF`.

CFICU Parameter

The parameter `CFICU` and its subparameters are explained in detail in the *Parameter Reference*. Some of the subparameters have an impact on the performance.

If collation services are used to compare Unicode strings, both strings are checked whether they are normalized or not. The check itself consumes a lot of CPU time. If you are sure that the strings are already normalized, you can switch off the check (`COLNORM=OFF`).

In Unicode, it is possible to represent the same character as one code point or as a combination of two or more code points. For example, the German character "ä" can be represented by "U+00E4" or by the combination of the code points "U+0061" and "U+0308". The conversion from Unicode to, for example, IBM01140 treats combined characters as single code points and produces an "a" followed by a substitution character since code point "U+0308" is not represented in the target code page. With `CNVNORM=ON`, a normalization is performed right before the actual conversion. The normalization consumes additional CPU time and temporary storage. If you are sure that no combining characters are involved in `MOVE` statements (except `MOVE NORMALIZED`), you should set `CNVNORM` to `OFF` to increase performance. Note that all possible combinations are represented by a single coded Unicode code point.

Conversion from Unicode to code page and vice versa is not high-performance. The reason is that the ICU implementation is written in C++ and that it covers nearly all Unicode, code page and language aspects in the world. However, some code pages can be mapped to Unicode (and vice versa) via translation tables to accelerate conversion. Accelerator tables are activated with the `CPOPT` subparameter. If it is set to `ON`, Natural automatically creates two accelerator tables during session initialization by using ICU conversion functions. The first table (with a size of 512 bytes) is used for conversion from code page to Unicode and the other table (with a size of 65535 bytes) is used for conversion from Unicode to code page. During a Natural session, all conversions are then executed via the accelerator tables instead of ICU calls. Accelerator tables are only provided for the default code page (`*CODEPAGE`). Temporary code pages (for example, in `MOVE ENCODED` statements) do not use accelerator tables if the module `NATCPTAB` is not linked. If it linked, up to 30 accelerator tables based on the ICU database are used to speed up performance.

Shared FUSER

Since Natural sources are not converted to Unicode or UTF-8 before saving, they can still be read by previous Natural versions. The additional code page information of Natural Version 4.2 (or above) sources is stored in the header of the source. The code page information in the header is simply ignored if a source is accessed by a previous Natural version.

CPAGE Compiler Option

The compiler option `CPAGE` creates objects that can be executed with a code page which is different from the code page used at creation time. This means that all alphanumeric constants of the object which are coded with the code page at creation time, have to be converted to the code page which is active at execution time. To make it possible for the Natural object loader to find and convert alphanumeric constants, an additional table is created by the compiler. This increases the size of the generated object, depending on the number of used alphanumeric constants. The conversion at runtime consumes additional CPU time. If the default code page (value of the system variable `*CODEPAGE`) is the same as the code page at creation time or if the session has no default code page (`CP=OFF`), no conversion is done. Conversion errors are ignored, independent from the setting of the parameter `CPCVERR`. If the compiler option `CPAGE` is set to `OFF`, no conversion is performed at runtime and the alphanumeric constants are treated as they are.

The following sample program is cataloged with code page IBM01141 (German) and is executed with default code page IBM01140 (us). The characters "Ä", "Ö" and "Ü" are defined in both code pages, but at different code points.

Example 1 - CPAGE=OFF:

```
OPTIONS CPAGE=OFF
WRITE *CODEPAGE  'ÄÖÜ'
END
```

Output with code page IBM01140 (us):

```
Page      1
IBM01140                                     ¢\!
```

Example 2 - CPAGE=ON:

```
OPTIONS CPAGE=ON
WRITE *CODEPAGE  'ÄÖÜ'
END
```

Output with code page IBM01140 (us):

```
Page      1
IBM01140                                     ÄÖÜ
```

Program Sources

Natural sources on the mainframe are not stored in Unicode format but in the default code page of the current Natural session. The name of the code page is stored in the directory of the source. Therefore, as compared to previous Natural versions, the size of a source remains unchanged. But there is a check by the editor whether the code page of the source is equal to the default code page of the Natural session. If the code pages are different, the source is converted into the default code page with the possibility of conversion errors. If a character of the source is not mapped in the default code page, a window appears in the editor to allow manual conversion of the failed characters. For example, a source which has been created with code IBM01140 contains the following line:

```
WRITE '100 €'
```

If the source is edited again with Natural running with code page IBM037, a conversion error occurs since the character "€" is not mapped in code page IBM037.

Note that the conversion is done when the editor is started and not when the source is loaded.

NTCPAGE Macro

The most common standard for code page names is the IANA name. Therefore, the system variable *CODEPAGE contains the IANA name of the default code page. In the world of IBM, a code page is qualified by its Coded Character Set ID (CCSID). For Siemens, the Coded Character Set Name (CCSN) is most popular. Currently, Adabas uses the Entire Conversion Service definition (ADAECS). The macro NTCPAGE can be used to assign these different names to the unambiguous IANA name. NTCPAGE is part of the Natural configuration file (NATCONFIG).

The CP parameter accepts only those values which are defined in the Natural configuration file. It does not matter whether the IANA name, the CCSID/CCSN or the alias name is entered with the CP parameter. The alias name can be a user-defined name which is used to assign a more significant name to the code page. In any case, *CODEPAGE contains the IANA name of the selected code page.

In addition, a place holder character can be defined for a code page. It overwrites the default substitution character of that code page, which is normally a non-displayable character (for example, H'3F' in an EBCDIC code page). The place holder character can be used to avoid that non-displayable characters are sent to terminals.

Example:

```
NTCPAGE IANA=IBM01140,CCSID=1140,ECS=1140,ALIAS='US',PHC=003F
```

The values IBM01140, 1140 or US can be entered with the CP parameter to activate the code page. *CODEPAGE contains the name IBM01140. The substitution character of the code page will be replaced by "U+003F", which is a quotation mark (?).

The number of available code pages depends on the used ICU implementation.

Starting with Natural Version 4.2.5 and ICU Version 3.8 respectively, the appropriate NTCPAGE entry can be omitted. Instead, all code pages defined in the currently used data package can be used by Natural. An NTCPAGE entry is only necessary if an alternative alias name or place holder character is desired.

Unicode and Code Page Support for Databases

Adabas supports a wide range of code pages to handle the world's languages. Character encoding and data conversion take place within Adabas using Unicode as the default encoding for both storage (file encoding) and user representation (user encoding). If code page or Unicode support (CFICU=ON) is enforced for a session, the Natural application must specify a user encoding and communicate it to the Adabas nucleus when this session is opened (OP command). The ADALNK module converts Adabas buffer data depending of the setting of the caller. On mainframes, the Entire Conversion Services (ECS) are used for conversion, not ICU. On Open Systems, Adabas uses ICU instead. Therefore, the ECS name must be defined in the related NTCPAGE entry in NATCONFIG. At open time for the database, the Natural nucleus sends an OP command with the ACODE setting for all A fields and the WCODE setting (4095 which means UTF-16) for all W fields in the record buffer for the mainframe version of Adabas, with the WCHARSET setting in the record buffer for the Open Systems version of Adabas. The ACODE and/or WCODE option must be defined in the OPRB parameter for this database.

For more information on Adabas conversion, see the description of the OP command and the information on supplied UES (universal encoding support) encodings in the Adabas documentation for mainframes.

Translation Tables

Natural uses various tables for character translation and character-type definition. The contents of the tables can be modified via session parameters (TAB, UTAB1, UTAB2 and SCTAB) during the start of a Natural session.

If Natural is running with code page support (that is: the parameter CP has been set to ON, AUTO or to the name of a code page, the tables cannot be modified by the user. In this case, the following Natural startup message will be issued to notify the user that the above mentioned session parameters are not considered:

Character translation parameter *table-name* ignored due to CFICU=ON.

Natural adjusts the tables automatically to the requirements of the default code page (value of the system variable *CODEPAGE). See also *Translation Tables* in the *Operations* documentation.

Support of Multi-Byte Code Pages

Natural supports multi-byte code pages (MBCS) such as IBM-939 which is a Japanese code page based on EBCDIC and DBCS. Multi-byte code pages can be selected using the CP parameter (by setting CP to AUTO (if supported) or to the name of a code page). If Natural is running with a multi-byte code page, it uses internal I/O buffers which are based on Unicode. This means that all data written into the internal I/O buffers by an I/O statement are converted to Unicode. Due to the requirements of Unicode and multi-byte code pages, the size of the I/O buffers is increased as compared to the traditional I/O since Unicode characters need twice as much space as EBCDIC characters and enhanced attributes are needed to describe a field.

In the case of single-byte code pages (SBCS) such as IBM-1140, the traditional EBCDIC-based I/O is still used to preserve resources.

ICU Buffer Pool

The ICU buffer pool is an optional local buffer pool which can be used to improve performance in a multi-user environment (for example, CICS). ICU-related data consists of static as well as dynamically generated ICU data. The amount of static data is fixed and depends on the appropriate ICU version. The amount of dynamically generated ICU data depends, for example, on the used converters and collation objects.

The ICU buffer pool is available in the following environments:

- CICS under z/OS and z/VSE
- Com-plete under z/OS and z/VSE
- openUTM
- batch server environments under z/OS and z/VSE

If an ICU buffer pool is not used, all ICU-related data is stored in buffers that are allocated for each individual Natural session. If an ICU buffer pool is used, the data are shared between Natural sessions. Because the ICU buffer pool is not affected by terminal I/O, processing of the ICU-related data usually located in the session-specific buffers before and after a terminal I/O is avoided and thus performance is significantly increased when an ICU buffer pool is used.

An ICU buffer pool can be defined using the profile parameter definition `BPI=(TYPE=ICU, NAME=' ', SIZE=value)` or the equivalent definition in the Natural parameter module using the `NTBPI` macro. The `SIZE` keyword subparameter should be set to `SIZE=200` at least. If more complex converters such as multi-byte converters or additional services such as collation service are used, `SIZE=600` is recommended.

It is not possible to use the ICU buffer pool with different NATICU versions. If two different NATICU versions are used in the same environment, the ICU buffer pool is initialized by the first Natural session with the NATICU version of that session. If another Natural session with a different NATICU version is started afterwards, NATICU detects that the ICU buffer pool is already used by another NATICU version, and will store all ICU-related data in session-specific buffers as with previous Natural versions. It is recommended to terminate all sessions before refreshing NATICU or refreshing Natural, if NATICU is linked to Natural, or switching to another NATICU version, to ensure that the ICU buffer pool will be used again after a restart of NATICU. If NATICU is loaded dynamically via RCA=NATICU, a refresh of the Natural nucleus has no impact on NATICU and the ICU buffer pool.

The ICU buffer pool is initialized by the first Natural session using it. The number of Natural sessions currently using the ICU buffer pool is counted. When the last Natural session using the ICU buffer pool terminates, it is cleaned up.

ICU prefers to use the local ICU buffer pool. If the local ICU buffer cannot be used, the initialization message NAT3419 will be displayed. The following possible reasons are displayed in the message text:

- RC=1 Local ICU buffer pool is not available.
- RC=2 Size of local ICU buffer pool is not sufficient.
- RC=3 Local ICU buffer pool is already used by a different ICU nucleus.

If the keyword subparameter BPONLY of profile parameter CFICU or the corresponding macro NTCFICU is set to ON, the ICU initialization will be terminated with the consequence that no ICU handler is available for the current Natural session.

If the keyword subparameter BPONLY is set to OFF, the required buffers are allocated in the Natural thread and the ICU initialization will be continued.

NAT3419 will be omitted if BPONLY is set to OFF and an ICU buffer pool is not available.

9

Migrating Existing Applications

| | |
|---|----|
| ■ Impact of Unicode on Existing Applications | 62 |
| ■ Migrating Existing Objects on Windows, UNIX and OpenVMS Platforms | 62 |
| ■ Migrating Existing Objects on Mainframe Platforms | 63 |
| ■ Adding Unicode Support to Existing Applications | 65 |
| ■ Migrating Natural Remote Procedure Calls (RPC) | 66 |

Impact of Unicode on Existing Applications

Windows, UNIX and OpenVMS Platforms

On Windows, UNIX and OpenVMS platforms, Natural has internally been Unicode-enabled which means that many structures containing strings have Unicode format now. For example, the Natural source area has now Unicode format. For this reason, data which is only available in code page format is internally converted to Unicode format. This applies, for example, to the Natural sources and to the Natural library names and object names. However, a conversion from code page to Unicode and vice versa can only be performed successfully if the correct code page is used for conversion. Even if an application is not changed but only re-cataloged, the code page information is important because for cataloging an object is loaded into the Natural source area. If all objects are coded in the system code page, no changes are necessary. If the objects are not coded in the system code page, see [Migrating Existing Objects on Windows, UNIX and OpenVMS Platforms](#) for further information.

The internal Unicode structure will most probably need more memory. If you have defined a low value for the profile parameter `USIZE`, it may be necessary to increase this value.

Mainframe Platforms

There is no impact of Unicode on existing applications. The internal structures have not been changed and no conversion of A format fields is enforced. This means that existing Natural applications should execute without any effort. Make sure that the parameters `CFICU` and `CP` have to be set to `OFF`. Even the need to link one of the ICU load modules (`NATICU`, `NATICUCV` or `NATICUXL`) becomes redundant in this case. Only the I/O buffers have been noticeably increased since the attributes have been enhanced to support potential Unicode fields. If `CP` is set to `OFF`, the system variable `*CODEPAGE` is cleared and the well-known translation tables (such as standard table or alternative table) are continued to be used for I/O translations.

Migrating Existing Objects on Windows, UNIX and OpenVMS Platforms

Natural has been extended so that the code page information can be defined on several levels:

- The Natural profile parameter `CP` defines the default Natural code page.
- For several objects (Natural sources, Natural batch input/output files, work files of type ASCII, ASCII compressed, Unformatted and CSV) an object-specific code page can be defined.

If neither an object-specific code page nor a default code page is defined, Natural will use the operating system's code page.

Since it is not possible to identify the correct code page automatically, it is important that you define the required code page information yourself. The following scenarios are possible:

| Status | Effort | Action |
|--|---|---|
| All data is available in the operating system's code page. | No effort | No action. |
| All data is stored with one code page, but this code page differs from the operating system's code page. | Easy | The Natural profile parameter <code>CP</code> has to be set to the correct code page. |
| The data is available in different code pages. | Depends on the number of sources and code pages | <p>The correct code page has to be defined for every Natural object:</p> <ul style="list-style-type: none"> ■ Sources If only few objects are affected, change the code page via the Properties dialog box. If several objects (for example, an entire library) are affected, use the <code>FTOUCH</code> utility for changing the code page. ■ Batch Files Set the Natural profile parameters <code>CPOBJIN</code>, <code>CPSYNIN</code> and <code>CPPRINT</code> to the correct code page. ■ Work Files Set correct code page for the work files in the Configuration Utility. |
| Different code pages are mixed in one object (for example, in a source) | High | The object has to be rewritten in UTF-8 format. |

Migrating Existing Objects on Mainframe Platforms

Natural has been extended so that the code page information can be defined on several levels:

- The Natural profile parameter `CP` defines the default Natural code page.
- For several objects (Natural sources, Natural batch input/output files, print reports, Adabas files) an object-specific code page can be defined.

If neither an object-specific code page nor a default code page is defined (that is, `CP=OFF` applies), Natural does not convert any data.

Since it is not possible to identify the correct code page automatically, it is important that you define the required code page information yourself. The following scenarios are possible:

| Status | Effort | Action |
|--|---|--|
| All data is available in the operating system's code page. | No effort | No action. |
| All data is stored with one code page, but this code page differs from the operating system's code page. | Easy | The Natural profile parameter CP has to be set to the correct code page. Make sure that the I/O device supports this code page. CP=AUTO forces Natural to run with the code page of the I/O device. |
| The data is available in different code pages. | Depends on the number of sources and code pages | <p>The correct code page has to be defined for every Natural object:</p> <ul style="list-style-type: none"> ■ Sources Save each object in the session with the correct code page. ■ Batch Files Set the Natural profile parameters CPOBJIN, CPSYNIN and CPPRINT to the correct code page. ■ Adabas Files (ECS enabled) Set the Natural profile parameter OPRB with the ACODE option. |
| Different code pages are mixed in one object (for example, in a source) | High | The object has to be rewritten in the appropriate code page format. |

Sources which have been saved or stowed with previous Natural versions do not have code page information. The code page field of the directory is empty.

Since Natural sources are not saved in Unicode format, the source has to be converted into the default code page (value of the system variable *CODEPAGE) that applies to the session. If code page support is switched off (CP=OFF), the code page information of the source is ignored and no conversion is performed. Alphanumeric constants have to be adjusted to the default code page when they are loaded into the source area.

Since Natural sources are not saved in Unicode format, alphanumeric constants have to be adjusted to the default code page during start of the object. This can be achieved with the CPAGE compiler option. If CPAGE is set to ON, an additional table is generated into the object. The Natural loader uses this table to convert every alphanumeric constant to the default code page (value of the system variable *CODEPAGE). Depending on the amount of alphanumeric constants, the additional table increases the size of the resulting object and the conversion consumes additional CPU time.

It is important that dependent objects (for example, a program and a local data area used by the program) use the same code page. If dependent objects use different code pages, it should be ascertained that the used characters (for example, "#") are mapped to the same code points in the

used code pages. The following objects and data do not have an associated object-specific or data-specific code page:

- Data definition modules (DDMs),
- Predict rules,
- Predict XRef data.

Care should be taken if such data is used in or produced by objects for which an object-specific code page has been defined. If the application itself does not necessarily have to be code page enabled and you want the application to be code page sensitive with respect to the data that is being processed, you should consider to use the profile parameter `SRETAIN` with the value `(ON, EXCEPTNEW)`.

Adding Unicode Support to Existing Applications

It is easy to extend existing applications with new source code based on the U format. The following rules have to be regarded for the U format (as compared with the A format):

- A `REDEFINE` of U to a format other than U should be avoided because this may result in split characters.
- U format is endian-dependant. This has to be considered when moving between the formats B and U.
- Align U in `DEFINE DATA` for performance reasons (better performance on UNIX and OpenVMS).
- Keep in mind that characters may be lost when moving U to A.

If you want to change existing fields from A format to U format, the following rules have to be regarded:

- Code which assumes a specific encoding of strings has to be changed (for example, comparison with a B field).
- All `REDEFINE` statements of the field have to be checked for their validity.
- A `REDEFINE` to N is not possible (that is: you will not get the expected result).
- The database field has to be migrated to Unicode (provided that this is supported by your database).
- You may have to change the length of the field: if the A field contains DBCS characters, half the length is required for the U field.

Migrating Natural Remote Procedure Calls (RPC)

The profile parameter `CP` has been renamed to `CPRPC`. In earlier Natural versions, `CP` was used to specify the name of the code page used by the Entire Conversion Service (ECS) and applied only to the Natural Remote Procedure Call when the transport protocol ACI (that is EntireX Broker) was used.

As of version 6.2 (Windows and UNIX), version 6.3 (OpenVMS) and version 4.2 (mainframe), a new `CP` parameter is available which defines the default code page for Natural data. When you are working with Natural RPC and have previously used the `CP` parameter dynamically, you have to change this parameter to `CPRPC`.

Windows, UNIX and OpenVMS Platforms

When you use parameter files from a previous version, you need not change anything; the Configuration Utility automatically migrates `CP` to `CPRPC`.

Mainframe Platforms

The parameter `CP` is used in conjunction with the parameter macro `NTCPAGE` (in the source module `NATCONFIG`) to specify the name of the default code page for Natural data or to automatically take the code page name from the user terminal.

The parameter `CPRPC` is used with the profile parameter `RPC` and the corresponding macro `NTRPC`.

10

Special Considerations and Limitations

| | |
|---|----|
| ■ Windows, UNIX and OpenVMS Platforms | 68 |
| ■ Mainframe Platforms | 68 |

Windows, UNIX and OpenVMS Platforms

- The dialog editor, which is provided with Natural for Windows, and dialog-based runtime is not Unicode-enabled.
- The editors provided with Natural for UNIX and Natural for OpenVMS are not Unicode-enabled.
- If the `DL` parameter is specified for a field which is longer than 250 characters, a maximum of 250 characters will be displayed in the field.
- A Natural source line may not be longer than 250 bytes. The program editor, which works on Unicode format, checks only that the number of UTF-16 code units is not greater than 250. However, depending on the encoding of the source, the line length may increase when converting the encoding from UTF-16 to the source encoding. For example, the UTF-8 encoding requires up to 4 bytes for a Chinese character; an error will be displayed in this case and the changes will not be saved.
- For UNIX and OpenVMS, Unicode is only supported at runtime with the Natural Web I/O Interface. If an application is run in the terminal emulation or xterm and Unicode strings are displayed, strange effects may occur.
- Compared with previous Natural versions, the performance is degraded since several conversions between code page and Unicode have to be performed.

Mainframe Platforms

- The editors provided with Natural for Mainframes are not Unicode-enabled.
- The size of the I/O buffers has been increased to be prepared for Unicode fields.
- Full Unicode I/O is only supported at runtime with the Natural Web I/O Interface. If an application is run in the terminal emulation and Unicode strings are displayed, some Unicode characters may not be displayed correctly.

11

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.

Natural provides a basic support for bidirectional languages. On Windows, this support is activated when both the Natural default code page and the Windows system code page are defined as bidirectional code pages. If Natural does not define a specific code page, it is sufficient when a bidirectional Windows system code page has been defined. On UNIX and OpenVMS, the support for bidirectional languages is activated when the Natural default code page is a bidirectional code page. On mainframes, support for bidirectional languages cannot be activated automatically; the user always has to specify all required parameters (for example, `PM=I`) as described below .

The output of Natural programs can be controlled using the profile parameter `PM`, the terminal command `%V`, and the session parameter `PM`.

On mainframes, UNIX and OpenVMS, the profile parameter `D0` (Display Order) is additionally 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. With the parameter `D0`, these applications are enabled to run compatibly also with I/O devices that support bidirectional data. This is for instance the case if an application runs in a browser with the Natural Web I/O Interface.

The profile parameter `PM` defines the default screen direction. When `PM` is set to `R` (reset), the default screen direction is left-to-right. When `PM` is set to `I` (inverse), the default screen direction is right-to-left. All non-alphanumeric fields, system variables and (only on mainframes) PF key lines are automatically inverted by Natural so that they are displayed correctly from right-to-left if the screen direction is right-to-left. On UNIX and OpenVMS, PF key lines are not inverted; they are always shown from left-to-right.

The terminal command `%V` can be used to change the screen direction. If the screen direction is right-to-left, the layout of the current window is mirrored, which means that the origin of all

window components or fields is the upper right corner. The screen direction is changed to right-to-left using %VON and is reverted to left-to-right using %VOFF.

The session parameter `PM` reverses the direction of a field. The effect of “reversing the direction of a field” depends on the statement in which the `PM` parameter is used and the platform. If the `PM` parameter is used in a `MOVE` statement, the content of the field is simply reversed (that is, the first character will become the last character, and so on); the result does not depend on the characters of the field. Trailing blanks are removed before the field is reversed.

For example, the following program

```
DEFINE DATA LOCAL
1  TEST1  (A10)
1  TEST2  (A10)
END-DEFINE
TEST1 := 'program'

MOVE TEST1 (PM=I) TO TEST2
INPUT TEST1 (AD=0) TEST2 (AD=0)

END
```

produces the following output:

```
TEST1 program  TEST2 margorp
```

where “margorp” is the reversed version of “program”.

When the `PM` parameter is used for IO statements such as `INPUT` or `DISPLAY`, its effect is even more complex. In this case, the field direction is based on the screen direction:

- If the screen direction is left-to-right and `PM=I` is applied to a field, the field direction changes to right-to-left.
- If the screen direction is right-to-left and `PM=I` is applied to a field, the field direction changes to left-to-right.

On Windows and browser terminals (Natural Web I/O Interface), “reversing the field direction” does not mean that the characters of the field are simply reversed. Instead, the complex bidirectional algorithm is applied (for more information, see the Microsoft Windows documentation). On other terminals (character-oriented), however, the characters of a field are not resorted; they are simply reversed.

In the following example, the characters assigned to the variable TEST have been entered in the following sequence:

a b c 1 2 3 1 2 3

The following is an example program for Windows. The characters of the constant are already resorted when entering them in the program editor.

```
DEFINE DATA LOCAL
1  TEST  (A20)
END-DEFINE
TEST := 'abc 123 123'

SET CONTROL 'voff'

INPUT TEST (AD=0) /
      TEST (AD=0 PM=I)

SET CONTROL 'von'

INPUT TEST (AD=0) /
      TEST (AD=0 PM=I)
END
```

This program produces the following two screens on Windows:

```
TEST abc 123 123
TEST          123 123 abc
```

and

```
123 123 abc TEST
abc 123 123 TEST
```

The following is an example program for UNIX, OpenVMS and mainframes. If the characters are entered in the same sequence, the program is displayed in the following way, because the characters are simply displayed in the keying sequence.

```
DEFINE DATA LOCAL
1  TEST  (A20)
END-DEFINE
TEST := 'abc 123 123'

SET CONTROL 'voff'

INPUT TEST (AD=0) /
      TEST (AD=0 PM=I)
```

```
SET CONTROL 'von'

INPUT TEST (AD=0) /
      TEST (AD=0 PM=I)
END
```

On UNIX and OpenVMS, this program produces the following two screens:

```
TEST abc 123
TEST      321 cba
```

and

```
      321 cba TSET
abc 123      TSET
```

On mainframes, this program produces two identical screens (on mainframes, the statements SET CONTROL 'voff' and SET CONTROL 'von' do not apply to alphanumeric fields). Both screens look as follows:

```
TEST abc 123
TEST      321 cba
```

On Windows, UNIX and OpenVMS, the map editor simplifies the handling of maps with bidirectional fields by offering the **Reverse Map** command. This command changes the display direction of the current map. The position of the fields is not changed; only the view is changed. On Windows, this command applies only to the current map. On UNIX and OpenVMS, a flag is set so that all following maps are displayed reversed; a following **Reverse Map** command will restore the original situation.

On Windows, the output of dialogs can be controlled in a similar way: both the dialog itself and most of the dialog controls offer an RTL attribute. If the RTL attribute of the dialog is checked, the screen direction of the dialog is right-to-left. If the RTL attribute of other controls is checked, the direction of these controls is right-to-left.

The profile parameter PM defines the default setting of the RTL attribute for new dialogs. When PM is set to R (reset), the RTL attribute is unchecked by default. When PM is set to I (inverse), the RTL attribute is checked by default. The default setting of the RTL attribute for newly created controls of a dialog is derived from the RTL attribute setting of the dialog.

If the RTL attribute of a dialog is changed when the dialog already contains controls, a dialog appears asking whether the RTL attributes of the controls should also be changed.

When working with bidirectional languages on Windows, "GUI" is the preferred print method. With the print method "GUI", the printed page will show the same layout as the window displayed on the screen. The sorting of the field characters is identical. If the print method "TTY" is used, the

printed layout will most probably differ from the layout of the screen window because the field characters are printed in logical sequence. For fields with right-to-left direction, all characters are simply reversed (that is, the first character will become the last character, and so on).

12

Double-Byte Character Support

In most East Asian languages, language-specific characters in code page strings (that is, Natural format A) are represented by 2 bytes (the so-called double-byte characters) and ASCII characters (EBCDIC on mainframes) are represented by 1 byte. Thus, a code pages string consists of characters with different lengths: some have 1 byte and others have 2 bytes.

Natural provides a basic support for double-byte characters. On Windows, this support is activated when both the Natural default code page and the Windows system code page are defined as double-byte code pages. If Natural does not define a specific code page, it is sufficient when a double-byte Windows system code page has been defined. On UNIX and OpenVMS, the support for double-byte characters is activated when the Natural default code page is a double-byte code page. On mainframes, the profile parameter `CP` must be set to an EBCDIC MBCS code page, for example IBM-942.

When double-byte character support is enabled, Natural assures for all string manipulations that a double-byte character is treated as a unit. This is essential for keeping the meaning of a string.

If a single leading or trailing byte of a double-byte character is left over after the manipulation of a variable of format A (for example, after extracting a substring with the `SUBSTRING` option), this byte is replaced with a blank character.

For the example below, the code page `Shift_JIS` is selected. Variable `#A` contains a string which consists of four characters. The first and last character is the double-byte character "FULL WIDTH LATIN SMALL LETTER B" which is represented in code page `Shift_JIS` by the byte sequence `H'8282'`. The second and third character is the single byte character "LATIN SMALL LETTER A" which is represented by one byte `H'61'`. Thus, the hexadecimal representation of the full string is `H'828261618282'`.

```

DEFINE DATA LOCAL
  1  #A  (A10)
END-DEFINE

#A := ' b aa b '

WRITE #A #A (EM=H(6))
EXAMINE #A FOR PATTERN ' B ' REPLACE 'a'
WRITE #A #A (EM=H(6))

END

```

Without double-byte character support the output of the above program is as follows:

| | | | |
|--------|--------------|----------|----------|
| Page | 1 | 07-02-07 | 17:22:09 |
| b aa b | 828261618282 | | |
| B a b | 826161828220 | | |

This is the result of not having treated the character " b " (H'8282' in code page Shift_JIS) as one unit. The trailing byte of this character and the following character "a" (H'61') are falsely interpreted as the double-byte character " B " (H'8261' in code page Shift_JIS).

With double-byte character support, the output of the program is as expected:

| | | | |
|--------|--------------|----------|----------|
| Page | 1 | 07-02-07 | 17:22:09 |
| b aa b | 828261618282 | | |
| b aa b | 828261618282 | | |

13

Frequently Asked Questions

| | |
|--|----|
| ■ Why do I get the startup error "Invalid code page specified"? | 78 |
| ■ What is the "default code page"? | 78 |
| ■ What default code page is used? | 78 |
| ■ Should I save all Natural sources in UTF-8 format? | 78 |
| ■ How can I handle UTF-8 encoding with Natural code? | 79 |
| ■ Why are some characters not displayed correctly? | 79 |
| ■ Why do I get an error when I want to edit a Natural source? | 79 |
| ■ Why do I get an error when I want to save a Natural source? | 79 |
| ■ How can I find out the encoding of a Natural source? | 80 |
| ■ How can I change the encoding of a Natural source? | 80 |
| ■ How can I convert an existing Natural source into UTF-8 format (Windows, UNIX and OpenVMS only)? | 80 |
| ■ Which substitution character is used if a character cannot be converted? | 81 |
| ■ Can I use Natural 4.2 sources with previous Natural versions? | 81 |
| ■ Can I use UTF-8 sources with previous Natural versions? | 81 |
| ■ Why do I get a conversion error when cataloging a source which has UTF-8 format? | 81 |
| ■ Why do I get garbage On UNIX or OpenVMS when displaying U format via a terminal emulation? | 82 |
| ■ Can I work with a current SPoD client and an older SPoD server? | 82 |
| ■ Can I work with a current SPoD server and an older SPoD client? | 82 |

Why do I get the startup error "Invalid code page specified"?

The code page you have defined with the profile parameter `CP` does either not exist (see <http://demo.icu-project.org/icu-bin/convexp> for valid ICU code pages and <http://www.iana.org/assignments/character-sets> for the appropriate IANA names) or is an invalid default code page for the platform (for example, an EBCDIC code page cannot be used on a Windows, UNIX or OpenVMS platform).

On mainframe platforms, the code page needs to be specified in the Natural configuration file via `NTPAGE` (see also [NTCPAGE Macro](#)). Code pages that are not entered here are rejected as invalid, although they are available in the ICU implementation. Check whether the same IANA name, CCSID/CCSN or alias name as specified in `NATCONFIG` is used.

What is the "default code page"?

The default code page is the code page which is the result of the evaluation of the profile parameter `CP`. If `CP` is not filled (Windows, UNIX and OpenVMS), it is the current operating system code page.

What default code page is used?

The default code page which is used by Natural for conversions between code page and Unicode and vice versa can be detected by displaying the content of the system variable `*CODEPAGE`.

Should I save all Natural sources in UTF-8 format?

It depends on the characters you want to use and on the platforms on which your sources are located. If you want to use Unicode constants, UTF-8 is the only possibility to store all combinations of characters. However, you can define hexadecimal UH constants which can also be stored in code page sources. The disadvantage of hexadecimal constants is that you have to know the UTF-16 encoding for every character of the constant. On mainframes, UTF-8 format for sources is not possible at all. On UNIX and OpenVMS, UTF-8 sources can only be handled via SPoD; they cannot be handled locally on UNIX or OpenVMS.

How can I handle UTF-8 encoding with Natural code?

Use the `MOVE ENCODED` statement for conversion from UTF-8 to UTF-16: the code page "UTF-8" has to be used for the A format variable.

Why are some characters not displayed correctly?

Check if you are using the correct code page. If the code page is correct, check if the selected font supports the characters you want to display.

Why do I get an error when I want to edit a Natural source?

The code page which is defined for the source is not correct. When converting the contents of the source to Unicode, a conversion error occurs. Change the encoding of the source so that the conversion to Unicode is successful.

On mainframe platforms, the source is saved with the code page at creation time. You get a conversion error when the source could not be converted from the code page of the saved source into the code page of the current Natural session. You can start Natural with the code page of the source to avoid conversion or you can adjust non-convertible characters in the window which appears when the editor is started.

Why do I get an error when I want to save a Natural source?

You have entered characters in the source which cannot be converted to the code page which was used to read the source. Check if you have entered these characters by mistake or if you really want to save the characters in the source. In the first case, remove the faulty characters and save the source. In the second case, save the source in UTF-8 format or, if the characters are contained in U constants, use UH constants instead.

If you have not entered any characters which are not contained in the code page of the source, check whether the profile parameter `SRETAIN` has been set to `OFF`. In this case, the source will be saved with the default code page. If the concerned source was previously saved with a different code page, a conversion error may occur.

If you are connected to a mainframe environment via SPoD, the source from the mainframe is converted and edited in Unicode in the SPoD environment. If it is saved, it has to be converted

into the code page of the Natural server. A conversion error may occur if a Unicode character is not mapped in the code page of the Natural server session.

If you are in a native Natural for Mainframes environment (without SPoD) you do not get errors when saving a source since a conversion is not performed. The source is saved with the code page information of the current Natural session.

How can I find out the encoding of a Natural source?

In Natural Studio, invoke the **Properties** dialog box for the source node. The **General** page shows the encoding of the source. If the **Encoding** text box is empty, no specific encoding is stored for the source. This means that the default encoding is used when reading the source.

The list view windows of Natural Studio also show the encodings of all listed objects.

On mainframe platforms, code page information is part of the Natural source directory. Use the `LIST DIR` command on Natural for Mainframes to display the directory.

How can I change the encoding of a Natural source?

In Natural Studio, invoke the **Properties** dialog box for the source node. The **General** page shows the encoding of the source. If this is not the correct encoding, you can change it by choosing the **Change** button: a list of available code pages is shown and you can select the correct encoding for the source.

On mainframe platforms, you should start your Natural session with the desired code page using the `CP` parameter. Set the parameter `SRETAIN` to `OFF`, edit the source and save it. Now the source has the modified code page information. Or, you can use the `SYSCP` utility to check or change the code page assignment of a source.

How can I convert an existing Natural source into UTF-8 format (Windows, UNIX and OpenVMS only)?

Open the source in the Natural editor with the correct code page. Save the source with **Save As** and in the **Save As** dialog box, select UTF-8 as the encoding.

Which substitution character is used if a character cannot be converted?

This depends on the direction of the conversion: if a code page character cannot be converted to Unicode, the Unicode substitution character "U+FFFD" is used. If a Unicode character cannot be converted to a code page, the substitution character which is defined by ICU for this code page is used.

On Natural for Mainframes, the substitution character of the code page or, if specified in the configuration file, the place holder character is used.

For the conversion from Unicode to the default code page, the substitution character can be changed on Windows, UNIX and OpenVMS platforms by setting the profile parameter `SUBCHAR`.

Can I use Natural 4.2 sources with previous Natural versions?

On Natural for Mainframes, you can use Natural 4.2 sources with previous Natural versions. The layout of the source has not been changed and the additional code page information of Natural 4.2 sources will simply be ignored if the source is accessed with a previous version.

Can I use UTF-8 sources with previous Natural versions?

No. Previous Natural versions do not know any code page information; a UTF-8 source will be interpreted as the current system code page.

Why do I get a conversion error when cataloging a source which has UTF-8 format?

A Natural source with UTF-8 format cannot be cataloged because a code point cannot be converted (Windows, UNIX and OpenVMS only).

All A constants in a source with UTF-8 format are converted to the default code page when storing them in the generated program. Either remove the characters which are not contained in the default code page from the A constants or use U constants instead of A constants.

Why do I get garbage On UNIX or OpenVMS when displaying U format via a terminal emulation?

All characters which are not contained in the default code page will be replaced with the substitution character of the code page before displaying the output on a terminal emulation. For an ASCII code page, the substitution character defined by the ICU conversion table is often "0x1A", which could be a control character on UNIX or OpenVMS terminals. It is strongly recommended to use the Natural Web I/O Interface when using U format in I/O statements. If using a terminal emulation is essential, the substitution character (SUBCHAR) can be changed to a printable character (for example, "?").

On mainframe platforms, you can still use your terminal emulation since it is possible to replace the substitution character by a displayable place holder character via the `NTCPAGE` macro. The place holder character avoids garbage in case of non-convertible characters.

Can I work with a current SPoD client and an older SPoD server?

Yes, but you should set the code page of the SPoD client to the code page of the server sources.

See also *Prerequisites for Natural Single Point of Development* at http://documentation.software-ag.com/natural/spod_prereq/prereq.htm.

Can I work with a current SPoD server and an older SPoD client?

Yes, but this is not recommended if you have defined encodings for sources.

See also *Prerequisites for Natural Single Point of Development* at http://documentation.software-ag.com/natural/spod_prereq/prereq.htm.

Index

C

code page support, 1

U

Unicode support, 1

