Platform Differences

This chapter covers the following topics:

- Windows, UNIX and OpenVMS Platforms
- Mainframe Platforms

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 IBM273 IBM1025 IBM1026 IBM1047 IBM1097 IBM01140 IBM01141 IBM01145 IBM01146 IBM01147 US (alias for IBM01140) DE (alias for IBM01141) ES (alias for IBM01145) 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)	de_DE en_US es_ES fr_FR sv_SE

Session Modes

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

Settings	Description
CFICU=OFF, CP=OFF	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.
CFICU=ON, CP=OFF	For new applications that are using Unicode and code page conversion (MOVE ENCODED) but not default code page support. Therefore, the system variable *CODEPAGE is empty. It is possible to use U format variables, but it is not possible to use, for example, MOVE A TO U, since this requires the default code page information. The error NAT3411 will be issued indicating that no default code page is available.
CFICU=ON, CP=*value* *	For new applications that are using full Unicode as well as code page support.
CFICU=OFF, CP=*value* *	This combination is possible, but does not make sense, because code page support needs ICU services for conversion. Therefore, CFICU=ON 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. On z/VSE, z/OS and VM/CMS, a code page is qualified by its Coded Character Set ID (CCSID). On BS2000/OSD, 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 (`NATCONFG`).

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 NATCONFG. 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
- *open*UTM
- 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.