

Support of Different Character Sets with *NATCONV.INI*

The settings in the configuration file *NATCONV.INI* apply to the A format. For the U format, the ICU library is used.

This chapter describes how Natural supports different character sets. It covers the following topics:

- Why is the Support of Different Character Sets Important?
 - How to Use Different Character Sets
-

Why is the Support of Different Character Sets Important?

The support of multiple languages with different character sets represents Natural's approach towards internationalization. It can help you when using:

- upper-/lower-case translation of language-specific characters;
- language-specific characters in Natural identifiers, object names and library names;
- language-specific characters in an operand compared with a mask definition (see *MASK Option* in the *Programming Guide*).

How to Use Different Character Sets

All check, translation and classification tables used by Natural to support language-specific characters reside in the configuration file *NATCONV.INI*. By default, this file is located in Natural's *etc* directory.

You can modify *NATCONV.INI* to support local or application-specific character sets.

In a standard application, *NATCONV.INI* need not and should not be modified, because this could lead to serious inconsistencies, in particular if Natural objects and database data are already present.

Any modifications of *NATCONV.INI* should be well considered and carefully performed, otherwise problems might occur that are difficult to locate.

NATCONV.INI is subdivided in sections and subsections. The following sections are defined:

Section	Description
CHARACTERSET-DEFINITION	<p>This section defines the name of the internal character set. The default is "ISO8859_1".</p> <p>If you choose a different character set, subsections for this character set must be contained in the sections described below.</p>
CASE-TRANSLATION	<p>This section contains the tables required for the conversion from upper-case to lower-case which is performed when one of the following is specified:</p> <ul style="list-style-type: none"> ● the terminal command %U, ● the field attribute AD=T, ● the statement EXAMINE TRANSLATE. <p>This conversion is done within the internal character set. If the internal character set is, for example, "ISO8859_5", the following two subsections must be contained in this section:</p> <ul style="list-style-type: none"> ● [ISO8859_5->UPPER] ● [ISO8859_5->LOWER]
IDENTIFIER-VALIDATION	<p>This section contains the tables required for the validation of identifiers (that is, user-defined variables in source programs), object names and library names. It contains a subsection for each defined internal character set.</p> <p>The special characters "#" (for non-database variables), "+" (for application-independent variables), "@" (for SQL and Adabas null or length indicators) and "&" (for dynamic source generation) can be redefined in this section. In addition, the set of valid first and subsequent characters for identifiers, object names and library names can be modified.</p> <p>Note: When extending the set of valid characters for object names with values greater than "x7f" (decimal 127), the sorting sequence of the objects (for example, during a LIST * command) may not be in the numerical order.</p>
CHARACTER-CLASSIFICATION	<p>This section contains the tables required for the classification of characters, which, for example, are used when evaluating the MASK option. It contains a subsection for each defined internal character set.</p>

The section CHARACTERSET-DEFINITION and each subsection contain lines which describe how characters are to be converted and which characters are related with which attributes. These lines are represented as follows:

```

line      ::= key = value
key       ::= name_key | range_key
name_key  ::= keyword{ CHARS }
keyword   ::= INTERNAL-CHARACTERSET | NON-DB-VARI | DYNAMIC-SOURCE |
            GLOBAL-VARI | FIRST-CHAR | SUBSEQUENT-CHAR |
            LIB-FIRST-CHAR | LIB-SUBSEQUENT-CHAR | ALTERNATE-CARET
            ISASCII | ISALPHA | ISALNUM | ISDIGIT | ISXDIGIT |
            ISLOWER | ISUPPER | ISCNTRL | ISPRINT | ISPUNCT |
            ISGRAPH | ISSPACE
range_key ::= hexnum | hexnum-hexnum
value     ::= val { , val }
val       ::= hexnum | hexnum-hexnum
hexnum    ::= xhexdighexdigit | xhexdighexdigit

```

Notes:

1. If the `range_key` variable is specified on the left-hand side, the number of values specified on the right-hand side must correspond to the number of values specified in the key range, unless only one value is specified on the right-hand side, which is then assigned to each element of the key range.
2. When the `name_key` variable is specified on the left-hand side and the corresponding list of character codes does not fit in one line, it can be continued on the next line by specifying "name_key =" again. You must not start the lines with leading blanks or tabulators.

Examples of Valid Lines

<code>x00-x1f = x00</code>	All characters between "x00" and "x1f" are converted to "x00".
<code>x00-x7f = x00-x7f</code>	All characters between "x00" and "x7f" are not converted.
<code>x00-x08 = x00,x01-x07,x00</code>	The characters "x00" and "x08" are converted to "x00" and characters between "x01" and "x07" are not converted.
<code>ISALPHA = x41-x5a,x61-x7a,xc0-xd6,xd8 ISALPHA = xd9-xf6,xf8-xff</code>	The attribute ISALPHA is assigned to all characters specified in these two lines.

Examples of Invalid Lines

<code>x41 = 'A'</code>	All characters must be specified in hexadecimal format.
<code>0x00-0x1f = 0x00</code>	Hexadecimal values have to be specified in either of the following ways: <i>xdigitdigit</i> <i>Xdigitdigit</i>
<code>x00-x0f = x00,x01</code>	The number of specified values does not correspond to the number of elements in the key range.