Web Application Development

Unicode Systems: Outside Communication for ABAP Developers

Exercises / Solutions

Dr. Christian Hansen
Server Technology
Internationalization
SAP AG

THE BEST-RUN BUSINESSES RUN SAP

Preparation

1. Availability

The exercises are available with SAP_BASIS 6.20 support package SP21.

2. Logon language

The standard logon language for the exercises is English.

3. GUI settings

For logon in the Unicode systems you need the following GUI settings:

- SAPLogon → click on the Unicode system → properties → Advanced: switch off 'Default Code Page' and enter code page Nr. 4110
- Activate the Multibyte options at the first logon (ALT+F12 → options → I18N) and restart the SAPGui

4. Copy exercises

Before you start with the exercises you need to create own copies of the demo programs:

- Copy TECHED_UNICODE_EXERCISE_n to ZTECHED_UNICODE_EXERCISE_n_XX where XX ist the two digit group number and n varies from 11 to 22.
- Copy the function group TECHED_UNICODE_RFC to ZTECHED_UNICODE_RFC_XX where XX ist the two digit group number (SE38 --> enter SAPLTECHED_UNICODE_RFC -> press copy button -> enter new function group name ZTECHED_UNICODE_RFC_XX).
- Copy the function module names UNICODE_RFC… to ZUNICODE_RFC…_XX.
- Adapt the calls of the function modules in your programs ZTECHED_UNICODE_EXERCISE_n_XX from UNICODE_RFC… to ZUNIOCDE_RFC…_XX (n=11 to 18).
- Do the same in the Unicode system and in the non-Unicode system.

5. Parameter

For the exercises you need to know the values of the following parameters. Please note that all parameters are case sensitive. Path names need to end on slash (/, Unix) or backslash (\, Windows).

Unicode system name	<UC>
Non-Unicode system name	<NUC>

Destination of the Unicode system:	<destination_UC>
Destination of the non-Unicode system:	<destination_NUC>
Path for filetransfer on the application server (UC)	<path_appl_UC>
Path for filetransfer on the application server (NUC)	<path_appl_NUC>
Path for filetransfer on the frontend	<path_GUI>

Example:

<UC>	= UP2
<NUC>	= NU2
<destination_UC>	= UP2_hansenc
<destination_NUC>	= NU2_hansenc
<path_appl_UC>	= D:\training\
<path_appl_NUC>	= D:\training\
<path_GUI>	= C:\sapworkdir\

6. General settings

- Please use the ALV grid display (SE16 → F6 → choose radio button “ALV Grid display”) to display table contents
- Enable line lengths greater than 72 (SE38 → Utilities → Settings → Editor → uncheck ‘Standard line length (72)’)

7. Table initialization

If the tables TECHED03_COLORS and TECHED03_UTEXT are still empty you have to run the programs RSCPCOLORS_03 and RSCPUTEXT_03 to initialize them.

Exercises

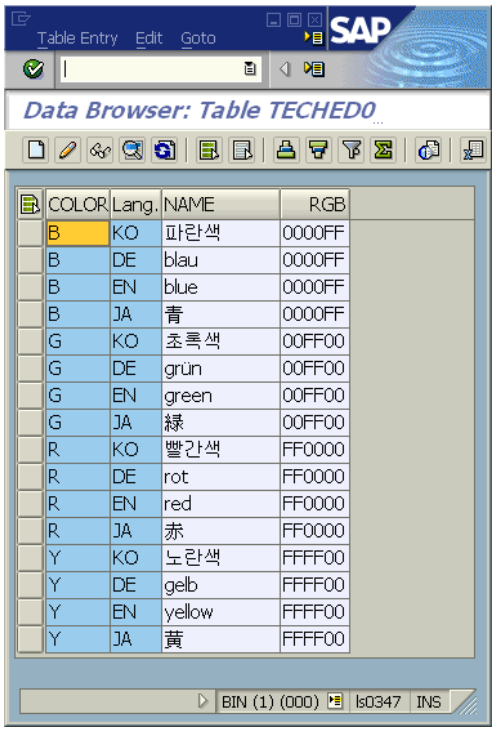
The different exercises involve transfering example data from the Unicode System to a non-Unicode system and vice versa.

Solutions are in TECHED_UNICODE_SOLUTION_n.

1. Send single code page and MDMP data via RFC

1.1. Type hiding and missing language keys

The original data is stored in table TECHED03_COLORS. In the Unicode system you see the following content.



COLOR	Lang.	NAME	RGB
B	KO	파란색	0000FF
B	DE	blau	0000FF
B	EN	blue	0000FF
B	JA	青	0000FF
G	KO	초록색	00FF00
G	DE	grün	00FF00
G	EN	green	00FF00
G	JA	緑	00FF00
R	KO	빨간색	FF0000
R	DE	rot	FF0000
R	EN	red	FF0000
R	JA	赤	FF0000
Y	KO	노란색	FFFF00
Y	DE	gelb	FFFF00
Y	EN	yellow	FFFF00
Y	JA	黄	FFFF00

You can see the full content in all languages at the same time only in the Unicode system. In the non-Unicode system you will only be able to see correctly texts that belong to your logon language.

In the target systems the results are written into the result table TECHED03_RESULT1. You may view the results using transaction SE16. Please use the ALV grid display (SE16 → F6 → choose radio button “ALV Grid display”). In a Unicode system the output looks like the following screen shot:

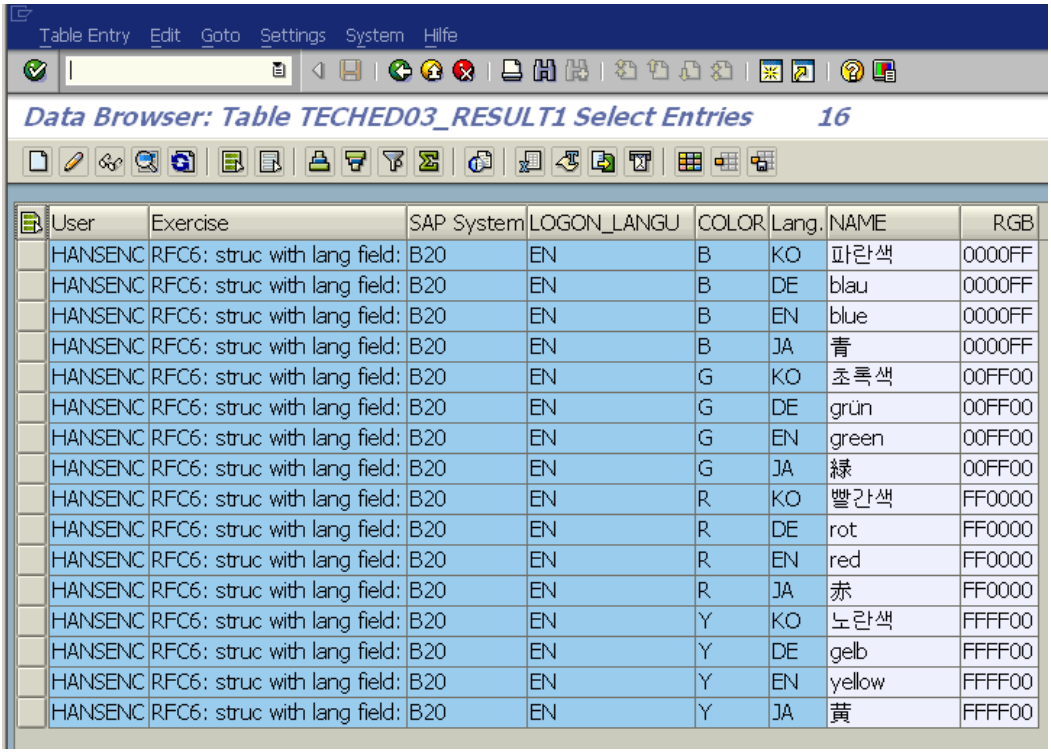


Table Entry Edit Goto Settings System Hilfe

Data Browser: Table TECHED03_RESULT1 Select Entries 16

User	Exercise	SAP System	LOGON_LANGU	COLOR	Lang.	NAME	RGB
HANSENC	RFC6: struc with lang field:	B20	EN	B	KO	파란색	0000FF
HANSENC	RFC6: struc with lang field:	B20	EN	B	DE	blau	0000FF
HANSENC	RFC6: struc with lang field:	B20	EN	B	EN	blue	0000FF
HANSENC	RFC6: struc with lang field:	B20	EN	B	JA	青	0000FF
HANSENC	RFC6: struc with lang field:	B20	EN	G	KO	초록색	00FF00
HANSENC	RFC6: struc with lang field:	B20	EN	G	DE	grün	00FF00
HANSENC	RFC6: struc with lang field:	B20	EN	G	EN	green	00FF00
HANSENC	RFC6: struc with lang field:	B20	EN	G	JA	緑	00FF00
HANSENC	RFC6: struc with lang field:	B20	EN	R	KO	빨간색	FF0000
HANSENC	RFC6: struc with lang field:	B20	EN	R	DE	rot	FF0000
HANSENC	RFC6: struc with lang field:	B20	EN	R	EN	red	FF0000
HANSENC	RFC6: struc with lang field:	B20	EN	R	JA	赤	FF0000
HANSENC	RFC6: struc with lang field:	B20	EN	Y	KO	노란색	FFFF00
HANSENC	RFC6: struc with lang field:	B20	EN	Y	DE	gelb	FFFF00
HANSENC	RFC6: struc with lang field:	B20	EN	Y	EN	yellow	FFFF00
HANSENC	RFC6: struc with lang field:	B20	EN	Y	JA	黄	FFFF00

You can see the full content in all languages at the same time only in the Unicode system. In the non-Unicode system you will be able to see correctly texts that belong to your logon language.

Hint: Narrow down the selection to the exercise/solution you are working on.

→ **Those who are interested only in the example solution should look into TECHED_UNICODE_SOLUTION_15**

1.1.1. TECHED_UNICODE_EXERCISE_11

This is the worst case scenario: Binary data is cast into a character container and send via RFC. Logon in the Unicode and the non-Unicode system in different languages (DE, KO, JA) , send the data and look at the result in the target system. Try to understand some of the results. (If you need to look at the binary content that is send and received you should use the debugger). In a first step towards the solution, introduce a language field to get a proper assignment between language and code page.

1.1.2. TECHED_UNICODE_EXERCISE_12

Now you have a language key included. Nevertheless, binary data is still cast into a character container and send via RFC. Try to convert the binary data (RGB values) into a character representation, replace the cast with a move, and send the character representation in the character container. If you solved this problem you may proceed with exercise TECHED_UNICODE_EXERCISE_14.

1.1.3. TECHED_UNICODE_EXERCISE_13

In this example you use an character representation for the binary data but have again forgotten a language key.

1.1.4. TECHED_UNICODE_EXERCISE_14

Now you have a language key included and use a character representation for the binary data and yet something goes wrong. Analyze and understand the layout shifts.
try two different methods to solve the problem:

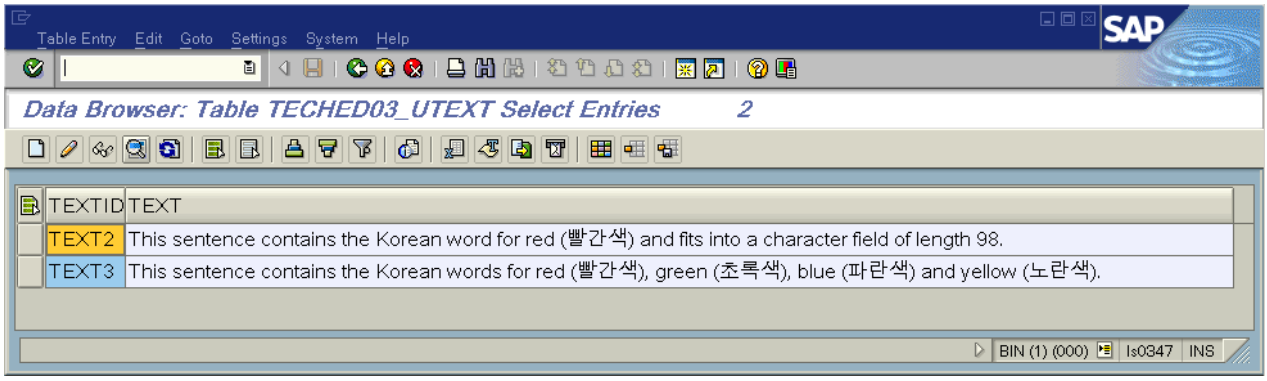
a) Use the function module NLS_GET_LANGU_CP_TAB to get the language/code page assigment and the class CL_NLS_STRUC_CONTAINER to correct the layout shift. (→TECHED_UNICODE_SOLUTION_14:)
b) Don't use an character field as transport container any more, be transparent and only use the correct structure type.

1.1.5. TECHED_UNICODE_EXERCISE_15

In this the data is send with the correct transparent structure data type. Nevertheless not all data is transmitted correctly. Find the reason for this behavior.
Hint: look for the text language attribute in SE11.

1.2. Wrong length assumptions

The original data is stored in the table TECHED03_UTEXT. In the Unicode system the content looks like this:



In the target systems the results are written into the result tables TECHED03_RESULT2.
For exercises TECHED_UNICODE_EXERCISE_16 and 18 you need to logon in Korean.

→ **Those who are interested only in the example solution should look into TECHED_UNICODE_SOLUTION_16/18**

1.2.1. TECHED_UNICODE_EXERCISE_16

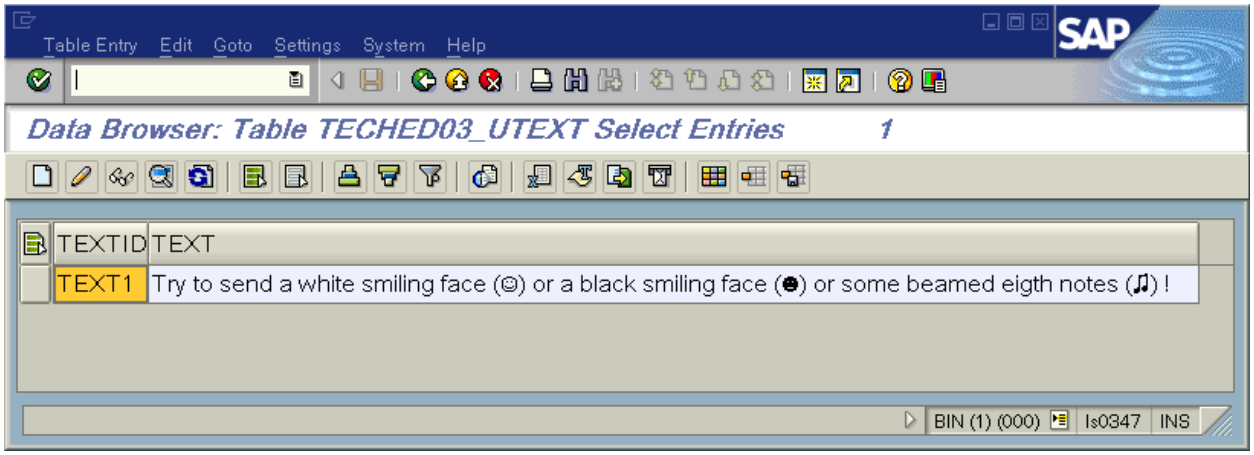
TEXT3 is packed into a table of fixed line size of 16 characters. The table is sent via RFC to the partner system. What is the reason for the character truncations or the insertion of unwanted spaces?

1.2.2. TECHED_UNICODE_EXERCISE_18

TEXT2 is is moved to a character field exactly matching the length in the Unicode system. What is the reason for the data truncation in the non-Unicode system?

1.3. Data not in the receivers code page

The original data is stored in the table TECHED03_UTEXT. In the Unicode system the content looks like:



In the target systems the results are written into the result tables TECHED03_RESULT2.

1.3.1. TECHED_UNICODE_EXERCISE_17

TEXT1 is sent from the Unicode system to the non-Unicode system. What can you do to transfer the data without it being replaced by a #?
(Answer: nothing. It is not possible. The characters are just not in the receiver's code page.)

2. Transfer data via file on the application server

In this exercises you are transfering data via file on the application server using the OPEN DATASET and TRANSFER/READ statements. If you don't have direct access to the files on the application server (to view them with a web browser or to open them with an editor) you may use the program TECHED_UNICODE_FILE_COPY to transfer a copy of the files to your local PC into the <path_GUI> directory.

→ Those who are interested only in the example solutions should look into TECHED_UNICODE_SOLUTION_19/20 :

- **'binary_mode_xml___'**
- **'text_mode_utf8____'**
- **'legacy_text_mode__'**
- **'legacy_binary_mode'**

2.1. Writing files

2.1.1. TECHED_UNICODE_EXERCISE_19

In this exercise you are writing the content of table TECHED03_COLORS into different files which shall be reread on the second system in exercise TECHED_UNICODE_EXERCISE_20. For each language one file is created and, in addition, one file is created containing all languages intermixed.

Please transfer data with the following OPEN DATASET modi:

- OPEN DATASET …FOR OUTPUT IN TEXT MODE
- OPEN DATASET …FOR OUTPUT IN BINARY MODE
- OPEN DATASET …FOR OUTPUT IN LEGACY TEXT MODE
- OPEN DATASET …FOR OUTPUT IN LEGACY BINARY MODE

Hint:
Make proper use of the language keys and set the system code page with the statement "SET LOCALE LANGUAGE …" where neccessary.
In case of "OPEN DATASET …FOR OUTPUT IN TEXT MODE" you must not transfer a structure (Dump!). Use a textual representation and "OPEN DATASET …FOR OUTPUT IN TEXT MODE ENCODING UTF-8" and output data field by field. You also can create an XML file with CALL TRANSFORMATION. Take a look at the XML file with your browser after copying it to your local PC with report TECHED_UNICODE_FILE_COPY.

2.2. Reading files

2.2.1. TECHED_UNICODE_EXERCISE_20

Reread the files you have created in ZTECHED_UNICODE_EXERCISE_19_XX (or take the files created with solution TECHED_UNICODE_SOLUTION_19) . The results again are stored to table TECHED03_RESULT1.

Hint:
Care of possible layout shifts with class CL_NLS_STRUC_CONTAINER.
Rereading the file that was written in the Unicode system in "binary mode" on the non-Unicode system is difficult. Please look at the solution and explain the usage of CL_ABAP_CONV_IN_CE and CL_ABAP_VIEW_OFFLEN.
You are asked about the end of line marker (EOF) used while the file was created. The default values are CTLF on Windows and CR on Unix platforms. (Please note, that the default values may be changed by the profile parameter abap/NTfmode).

3. Transfer data via file on the frontend

In this exercise transfer data via file on the local PC. Use the function modules GUI_UPLOAD / GUI_DOWNLOAD to save and read the files.

3.1. Writing files

3.1.1. TECHED_UNICODE_EXERCISE_21

Download the file with GUI_DOWNLOAD. Choose encoding UTF-8 as the code page. Set the system code page with "SET LOCALE LANGUAGE …". Download the data with field separator and make use of the APPEND possibility to download the data record by record.

3.2. Reading files

3.2.1. TECHED_UNICODE_EXERCISE_22

Reread the files you have created in ZTECHED_UNICODE_EXERCISE_21_XX (or take the files created with solution TECHED_UNICODE_SOLUTION_21).
Hint:
In order to read the content of the MDMP file read the file several times with different system code pages (SET LOCALE LANGUAGE) and select the data with matching language keys.

Copyright

.