TeamConnection frequently asked questions: National Language Support (NLS) and Double-Byte Character Sets (DBCS)

Document Number TR 29.2266

Angel Rivera, Keith Purcell

CMVC/TeamConnection Development
IBM Software Solutions
Research Triangle Park, North Carolina
Copyright (C) 1997 IBM Corp.
All rights reserved.

ABSTRACT

This technical report describes how to use TeamConnection Version 2 in situations that require National Language Support (NLS) and Double-Byte Character Sets (DBCS). Several important related issues are described, such as the need for using the same code page among the clients and the server, and how to setup an AIX workstation to handle different locales.

ITIRC KEYWORDS

TeamConnection

NLS

DBCS

SBCS

ABOUT THE AUTHORS

ANGEL RIVERA

Mr. Rivera is an Advisory Software Engineer working with the TeamConnection and CMVC development teams. He joined IBM in 1989 and since then has worked in the development and support of library systems.

Mr. Rivera has an M.S. in Electrical Engineering from The University of Texas at Austin, and a B.S. in Electronic Systems Engineering from the Instituto Tecnológico y de Estudios Superiores de Monterrey, México.

KEITH PURCELL

Mr. Purcell is an Advisory Software Engineer who has worked on several support and development teams, including TeamConnection. He joined IBM in 1979 and has worked with over 10 programming languages in more than 8 operating system environments in programming and support roles.

Mr. Purcell has an A.S. in Mathematics Computer Science from Dutchess Community College in Poughkeepsie, N.Y.

CONTENTS

ABSTRACT . . . iii
ITIRC KEYWORDS . . . iii

ABOUT THE AUTHORS . . . v
Angel Rivera . . . v
Keith Purcell . . . v

Figures . . . viii

Introduction . . . 1
Bibliography . . . 1
Bibliography for TeamConnection . . . 1
Bibliography for NLS and DBCS . . . 1

Overall view of NLS and DBCS support provided by TeamConnection . . . 3
Supported locales (languages and code pages) . . . 5
Supported Single-Byte Character Set (SBCS) locales . . . 5
Supported Double-Byte Character Set (DBCS) locales . . . 6
Supported platforms by TeamConnection . . . 7
Locales supported for the ObjectStore database . . . 8

Characteristics and limitations of the support for NLS and DBCS . . . 9
There is no conversion of code points when exchanging data . . . 9
There is no impact if using English characters . . . 9
There is impact if using non-English characters . . . 10
To maximize compatibility, use same/similar code page . . . 11
Once a family is created, do not change the code page . . . 11
Using UNICODE in the future to solve incompatibilities . . . 11
Exceptions to the handling of characters in TeamConnection . . . 12
The ¦ split vertical bar character could be changed . . . 12
Keyword expansion . . . 12
CR (carriage return) and LF (line feed) . . . 12
All clients in the same host must use the same language (Intel only) . . . 13
Untraslated strings that are visible to the users . . . 14
DBCS Limitations . . . 14

Installation, administration and runtime issues . . . 17
Installation issues related to NLS and DBCS . . . 17

Using a similar directory structure across all the platforms . . . 17
Installation issues for UNIX . . . 19
Installation issues with OS/2 and Windows . . . 19
Family administration issues . . . 20
A family should use the same language all the time . . . 20
Client runtime issues . . . 21
A client should use the same language all the time . . . 21

Miscellaneous topics that are specific to operating systems . . . 25
Problem resolution . . . 25
Potential problems if the language locale is not set properly . . . 25
When using LANG=C in AIX, the message catalogs are ignored! . . . 25
Using misc/tcmsgcat to verify the NLS message catalog settings . . . 25
Using misc/showmsg to display a given message from the catalog . . . 26
Problems when the UNIX teamcgui window does not show up . . . 27
Topics related to the message catalog used by TeamConnection . . . 27
Using the "teamc report" command to test the NLS settings . . . 28
How to know when TeamConnection is not using the message catalog . . . 29
Setting and verify the current locale setting in UNIX . . . 29
How to enter SBCS extended characters using an English keyboard . . . 29
How to specify NLS settings for AIX GUI on aixterm/dtterm windows . . . 30
Using aixterm . . . 30
Using dtterm . . . 31
Creating a Teamcgui resource file for AIX . . . 32
Using iconv to convert files to other code pages . . . 32
How to install additional locales for AIX . . . 33
How to change the keyboard for an AIX host . . . 34
How to install a non-English PC keyboard to an X-station . . . 34
How to specify a different keyboard mapping . . . 36

Copyrights, Trademarks and Service marks . . . 37

FIGURES

1. How TeamConnection handles the locale environment variables . . . 3
2. SBCS locales supported by TeamConnection . . . 5
3. SBCS locales supported by TeamConnection . . . 6
4. Platform support for TeamConnection components . . . 7

5. Graphical representation of code point 100 in several code pages 9
6. Graphical representation of code point 100 in several code pages 10
7. Sample output of "tcmsgcat.aix" 26
8. Sample output of "showmsg" 27
9. Sample output of "teamc report -testServer" 28
10. Sample output of "teamc report -testClient" 28

INTRODUCTION

This technical report describes how to use TeamConnection Version 2 in situations that require National Language Support (NLS) and Double-Byte Character Sets (DBCS).

The topics in this technical report are:

The overview of the NLS and DBCS support provided by TeamConnection Version 2, such as usage of locale XPG/4 I18N programming model, supported locales and platforms.

The main characteristics and limitations related to NLS/DBCS, such as the interoperability between clients and server, and special cases for the manipulation of data by TeamConnection.

The main issues related to installation, administration and runtime, such as directory structure of the installed code, and you should not change the code page of an existing TeamConnection family.

Miscellaneous topics, such as how to install locales in AIX.

Note: In OS/2 and Windows, the locale support is not provided by the operating system and it is provided by TeamConnection during installation.

BIBLIOGRAPHY

Bibliography for TeamConnection

For more information on how to use TeamConnection, you can consult the following manuals and publications:

SC34-4551 TeamConnection, Administrator's Guide
SC34-4552 Getting Started with the TeamConnection Clients
SC34-4499 TeamConnection, User's Guide
SC34-45 1 TeamConnection, Commands Reference
SC34-45 TeamConnection, Quick Commands Reference

Bibliography for NLS and DBCS

GG24-3850, AIX Version 3.2 for RISC System/6000, National Language Support.

SC23-2431, Internationalization of AIX Software, A Programmer's Guide

SE09-8001-02, National Language Design Guide Volume 1

This manual contains very good information on how to enable an application for NLS.

SE09-8002-02, National Language Design Guide Volume 2.

This manual provides information on the IBM language codes (consult the "Language codes" chapter).

OVERALL VIEW OF NLS AND DBCS SUPPORT PROVIDED BY TEAMCONNECTION

.LANGUAGE AND CULTURE SENSITIVE INFORMATION IN TEAMCONNECTION

TeamConnection supports the I18N (Internationalization) locale model proposed by XPG/4 (X/Open Portability Guide, issue 4) in which the language and culture sensitive information is not hard coded in the executables; instead, they are provided as system resources by means of a "locale" that the user can specify at run-time.

One of the components of a locale is the code page in which the characters will be handled. For example, in AIX, the default locale is "en_US" which is for English in the USA and the associated code page is ISO8859-1, which is different than the default one used for English in OS/2 (code page IBM-850) but similar to the one used for English in Windows (code page MS-1252 Latin 1).

The locale model for XPG/4 establishes several environment variables that can be used for controlling the culture sensitive information. The table Figure 1 describes these environment variables, their function and how TeamConnection deals with them.

Figure 1 (Page 1 of 2). How TeamConnection handles the locale environment variables

Locale environment variable	**Function**	**How TeamConnection uses it**
LANG	Specifies the installation default locale.	Identifies the path for the message catalogs and other language-sensitive files.
LC_ALL	Overrides the value of other LC_* environment variables.	It is not explicitly exploited by TeamConnection.
LC_COLLATE	Determines the character-collation or string-collation rules.	It is ignored by TeamConnection. (See Note 1).
LC_CTYPE	Determines the character handling rules governing the interpretation of sequences of bytes of text data characters and classification of characters.	It is ignored by TeamConnection.

Figure 1 (Page 2 of 2). How TeamConnection handles the locale environment variables

Locale environment variable	Function	How TeamConnection uses it
LC_MESSAGES	Determines the rules governing affirmative and negative responses, and the locale for messages and menus.	It is ignored by TeamConnection.
LC_MONETARY	Determine the rules governing monetary-related formatting.	It is ignored by TeamConnection, because it does not handle this kind of information.
LC_NUMERIC	Determine the rules governing non-monetary numeric formatting.	It is ignored by TeamConnection, because it handles only integer numbers with no separation for thousands.
LC_TIME	Determine the rules governing date and time formatting.	It is ignored by TeamConnection. There is no special processing for the date and time information. (See Note 2).

Notes:

1. TeamConnection itself does not perform any sorting of data, instead, the sorting is performed by the database. Because only the English version of the database is provided, by default the English character collation rules are used. However, there is an exception: the Japanese locale is provided for the database.
2. The date and time in TeamConnection is represented as follows:

```
YYYY/mm/dd hh:mm:ss
                    -> seconds
                ----> minutes
            -------> hours

          -> day
       ----> month
   -------> year
```

   Because 4 digits are used for the year, TeamConnection is compliant with the 2000 Year specifications.

SUPPORTED LOCALES (LANGUAGES AND CODE PAGES)

TeamConnection Version 2 provides support for the following locales (which include the translated message catalogs):

Single-Byte Character Set (SBCS) locales:

See Figure 2.

Double-Byte Character Set (DBCS) locales:

See Figure 3 on page 6.

Supported Single-Byte Character Set (SBCS) locales

Each row in the following table represents what code pages and locales are compatible accross the different platforms. For example, the locale "En_US" is different than the locale "en_US" and therefore each locale is explicitly described in a separate entry.

Figure 2. SBCS locales supported by TeamConnection

Language and Country	Locale	Code Page OS/2	Code Page Windows NT/95	Code Page AIX 4
English, USA (enu)	En_US	IBM-437, IBM-850	DOS: MS-437, MS-850 (note 1)	IBM-850
English, USA (enu)	en_US	n/a	GUI: MS-1252 (note 1)	ISO8859-1
Portuguese, Portugal (ptb)	Pt_PT	850	MS-850 (note 1)	IBM-850
Portuguese, Brazil (ptb)	pt_BR	n/a	GUI: MS-1252 (note 1)	ISO8859-1

Notes:

1. When using directly the TeamConnection line commands from a DOS Command Prompt in Windows, the DOS code pages are used (such as MS-850).

   In contrast, when using the TeamConnection GUI the MS code pages are used (such as MS-1252).

It is important to emphasize that the locales "Pt_PT" and "pt_BR" are different. If you use a TeamConnection family in AIX with the ISO locale pt_BR (code page ISO8859-1), then you will see "code page incompatibility" problems (some characters will NOT be shown or will not look OK) if you use a TeamConnection client in OS/2 with the Pt_PT locale (code page IBM-850).

Supported Double-Byte Character Set (DBCS) locales

Each row in the following table represents what code pages and locales are compatible accross the different platforms. For example, the locale "Ja_JP" is different than the locale "ja_JP" and therefore each locale is explicitly described in a separate entry.

Figure 3. SBCS locales supported by TeamConnection

Language and Country	Locale	Code Page OS/2	Code Page Windows NT/95	Code Page AIX 4
Japanese, Japan (jpn)	Ja_JP	IBM-932 (note 1)	MS-932 = IBM-943	IBM-932
Japanese, Japan (jpn)	ja_JP	n/a	n/a	IBM-eucJP
Korean, Korea (kor)	ko_KR	IBM-949	MS-949 (UHC) = IBM-1363 (note 2)	IBM-eucKR
Simplified Chinese, RPC (chs)	zh_CN	IBM-1381	MS-936 (GBK) = IBM-1386 (note 3)	IBM-eucCN
Traditional Chinese, Taiwan (cht)	Zh_TW	IBM-950 (note 4)	MS-950 (note 4)	BIG5 (note 4)
Traditional Chinese, Taiwan (cht)	zh_TW	n/a	n/a	IBM-eucTW = IBM-964 (note 5)

Notes:

1. The Japanese IBM-932, IBM-942 and IBM-943 have very small differences between them, but generally speaking, they are compatible with each other.
2. The Korean code page for Windows NT/95 is called UHC (Unified Hangeul Code).

   The IBM-1363 code page extends IBM-949 by adding missing Hangeul characters with no change of assignments in code points for IBM-949.
3. The code page for Simplified Chinese for Windows NT/95 is called GBK (Guo Biao Kuo).

   The IBM-1386 code page extends IBM-1381 by adding missing Unicode characters with no change of assignments in code points for IBM-1381.
4. The PC code page for Traditional Chinese is called Big-5.

5. The EUC code page for Traditional Chinese (IBM-eucTW) for AIX 4.1 has been enhanced with respect to AIX 3.2, but it keeps the same locale name (zh_TW). This means that if the user in AIX 4.1 exploits the new characters in the enhanced locale version, there could be compatibility problems when the user uses the old locale version.

It is important to emphasize that the locales "Ja_JP" and "ja_JP", and "Zh_TW" and "zh_TW" are different. If you use a TeamConnection family in AIX with the EUC locale ja_JP (code page IBM-eucJP), then you will see "code page incompatibility" problems (some characters will NOT be shown or will not look OK) if you use a TeamConnection client in OS/2 with the Ja_JP locale (code page IBM-932). The similar problem if you use incompatible Traditional Chinese code pages.

SUPPORTED PLATFORMS BY TEAMCONNECTION

The supported platforms by TeamConnection are shown in Figure 4.

Figure 4. Platform support for TeamConnection components

Platform	Server	Client	Build Agent	Build Processor
OS/2 Warp (3.x)	YES	YES	YES	YES
OS/2 Merlin (4.x)	YES	YES	YES	YES
Windows 3.1 (16-bit)	no	GUI	no	no
Windows 95 (32-bit)	no	YES	no	YES
Windows NT 3.51 (32-bit)	YES	YES	YES	YES
Windows NT 4.x (32-bit)	YES	YES	YES	YES
MVS	no	no	no	YES
AIX 4	YES	YES	YES	YES
HP-UX 10	YES	YES	YES	YES

Notes:

1. YES: for client means: both GUI and line commands
2. GUI: means ONLY the graphical user interface, no line commands
3. no: means that there is no support

LOCALES SUPPORTED FOR THE OBJECTSTORE DATABASE

TeamConnection uses the ObjectStore object-oriented database management system (DBMS), which is enabled to handle DBCS, regardless of the locale.

The installation of TeamConnection also includes the installation of ObjectStore and only the English locales for ObjectStore are included in the installation images. This means that if there are messages issued by ObjectStore, then these messages will be in English, regardless of the locale of the client or server.

Note: In UNIX, the installation images for ObjectStore include also the Japanese locales. If the ObjectStore server is running with the Japanese locale, then the messages from ObjectStore will be in Japanese.

CHARACTERISTICS AND LIMITATIONS OF THE SUPPORT FOR NLS AND DBCS

THERE IS NO CONVERSION OF CODE POINTS WHEN EXCHANGING DATA

The TeamConnection clients and servers do not alter the code points of the data. This means that the data is NOT converted from one code page to another when entered by the user, when stored in the database used by the family or when exchanged between a client and the server.

The information about the code page in which the data was entered is not stored with TeamConnection objects; furthermore, there is no exchange of information between the client and the server to indicate which code page is being used by each of them.

There is no impact if using English characters

Because most code pages have the same code points for the first 128 characters, which includes all the characters used in the English alphabet, then in practice there is no effect in using different code pages between clients and servers, if using only English characters.

As an example, the default multilingual code page for OS/2 is IBM-850, for Windows is MS-1252 Latin 1, and for AIX Version 4 is ISO8859-1; in these code pages the first 128 characters are the same, and thus, there is no impact in code points the English characters are used when entering remarks for a defect in the OS/2 client, then stored in the AIX server and retrieved by the Windows client.

For example, the code point value of 100 is the lower case letter "d" which has the same graphic representation in most of the code pages, as exemplified in Figure 5.

Figure 5 (Page 1 of 2). Graphical representation of code point 100 in several code pages

Platform	Locale	Code Page	Representation
OS/2	English	IBM-437	lower case 'd'
OS/2	English	IBM-850	lower case 'd'
Windows, DOS mode	English	MS-437	lower case 'd'
Windows, DOS mode	English	MS-850	lower case 'd'

Figure 5 (Page 2 of 2). Graphical representation of code point 100 in several code pages

Platform	Locale	Code Page	Representation
Windows, Graphical	English	MS-1252	lower case 'd'
AIX	En_US	IBM-850	lower case 'd'
AIX	en_US	ISO8859-1	lower case 'd'
OS/2	Japanese	IBM-932	lower case 'd'
Windows	Japanese	MS-932	lower case 'd'
AIX	ja_JP	IBM-eucJP	lower case 'd'

There is impact if using non-English characters

However, if the customer wants to use non-English characters, which are characters with code points greater than 128 such as accented characters, umlauts, double-byte characters, then the code pages differ greatly in this respect.

For example, the character with code point value of 252 (which can be entered by pressing ALT and typing 2, 5 and 2 from the numeric keypad in most systems) has the following different representations, as shown in Figure 6.

Figure 6. Graphical representation of code point 100 in several code pages

Platform	Locale	Code Page	Representation
OS/2	English	IBM-437	superscript 'n'
OS/2	English	IBM-850	superscript '3'
Windows, DOS mode	English	MS-437	superscript 'n'
Windows, DOS mode	English	MS-850	superscript '3'
Windows, Graphical	English	MS-1252	lower case 'u' with dieresis
AIX	En_US	IBM-850	superscript '3'
AIX	en_US	ISO8859-1	lower case 'u' with dieresis
OS/2	Japanese	IBM-932	First byte of DBCS character
Windows	Japanese	MS-932	First byte of DBCS character
AIX	ja_JP	IBM-eucJP	First byte of DBCS character

In the above case, a German customer using Windows in Graphical Mode, with code page MS-1252 may enter a string that contains the u with umlaut and store it in TeamConnection,

but the same customer when retrieving the data from OS/2 using IBM-850 code page, the character in the string will be shown as the number 3 in superscript.

To maximize compatibility, use same/similar code page

As shown in “There is impact if using non-English characters” on page 10, it is important that the customers who are using multiple platforms with TeamConnection, must understand the implications of using different code pages when dealing with non-English characters.

If possible, the customer should use the same (or similar) code page in the TeamConnection client and in the server.

Once a family is created, do not change the code page

To avoid compatibility problems, if a family is created and used with a given code page, then this code page should not be changed later on.

For example, if a family is created with the Japanese IBM-932 code page in OS/2 and then migrated to the Japanese IBM-eucJP code page in AIX, then there might be several DBCS characters that are valid in the IBM-932 code page that will not be displayed properly when using the IBM-eucJP code page.

Using UNICODE in the future to solve incompatibilities

In the future, once the support for the UNICODE code page is widespread and available in all the platforms supported by TeamConnection, then the customer could choose to use the UNICODE code page for the clients and the server, and in this way, avoid the current incompatibility between different code pages.

Another alternative that we studied to solve to this incompatibility problem between code pages was to add an extra field for EVERY SINGLE piece of data that is handled by TeamConnection in order to identify the code page that was used when the data was originated; then, this would require that the TeamConnection server should get the code page used by each client that is requesting a service, and then do the necessary conversions when exchanging the data. Because this alternative is very expensive to implement and has a lot of ramifications, and because UNICODE is the right way for the long term, we are not implementing this alternative to tag each piece of data.

EXCEPTIONS TO THE HANDLING OF CHARACTERS IN TEAMCONNECTION

The ¦ split vertical bar character could be changed

The ¦ split vertical bar character is used to separate the fields in the "teamc report -raw" command. Thus, if this character is found in a field that is shown by this command, such as in the abstract of a defect, then the character is changed to "!" (exclamation point) by the TeamConnection client. Thus, the server does not see these split vertical bar characters.

The reason for this change is to avoid confusion during the parsing of the -raw output because the split vertical bar is used to separate the fields. If in the output to be parsed there is a split vertical bar character that is NOT intended to be a separator of a field, then the parsing routine will not be able to guess that this particular split vertical bar should not be considered as a field separator. In other words, ALL split vertical separator bars are considered to be field separators, and thus, any such characters in the abstract will not be parsed appropriately.

For example, when opening a defect, if the abstract field is left blank then the first 63 characters of the remarks field will be placed in the abstract. The abstract is a field that is shown with the "teamc report -raw" command, but the remarks field is not shown with this command. Thus, if the first 63 characters of the remarks have split vertical bar characters they will be left untouched in the actual remarks, but they will be converted to "!" in the abstract.

Keyword expansion

TeamConnection supports the expansion of certain keywords embedded in the text during the extraction of text files. The routines that handle the expansion are NLS and DBCS enabled.

The important characteristic to remember is that the expansion is done by the TeamConnection family server and not by the client.

CR (carriage return) and LF (line feed)

Although this is not an NLS issue, this is another topic that is worth to include in this technical report, because some users may think, incorrectly, that this could be caused by code conversion processing done by TeamConnection.

The end of a line of text in OS/2 or in Windows is represented by the character pair CR-LF (carriage return and line feed), whereas in UNIX is represented simply by the character LF (line feed).

In TeamConnection, the model of what-you-see-is-what-you-get is used. This means that if a user creates a file in TeamConnection, regardless of the platform of the server, then TeamConnection will NOT do any conversion of LF or CRLF on that file. There are choices in the -extract action to allow for more fine tuning of these on-the-fly-conversions. For example, an AIX user may wish to extract with only LF a file that was stored originally from OS/2 that has CRLF.

The following file will be the source file to be used in the rest of the examples in this section:

```
This is line 1
This is line 2
This is line 3
```

If the source file is created from an OS/2 client and later on is extracted into a UNIX client without CRLF conversion, then the resulting file will have the CR character at the end of each line and the file would look like:

```
This is line 1^M
This is line 2^M
This is line 3^M
```

If the source file is created from a UNIX client and later on is extracted into an OS/2 client without CRLF conversion, then the resulting file will not have the CR character at the end of each line and the file would look like:

```
This is line 1
              This is line 2
                            This is line 3
```

ALL CLIENTS IN THE SAME HOST MUST USE THE SAME LANGUAGE (INTEL ONLY)

For OS/2 and Windows NT clients, all the TeamConnection clients that execute from one single host must use the same language if they run at the same time.

This limitation is due to the inherent limitation of these platforms in which ONLY ONE version of given DLL can be loaded at the same time, and because these platforms are not fully compliant with the XPG/4 model that allows usage of multiple locales. If there are different versions of some DLLs for each language, and if the English version of the DLL is loaded, the Japanese one cannot be loaded at the same time. This precludes having clients that have different languages to run at the same time.

UNTRASLATED STRINGS THAT ARE VISIBLE TO THE USERS

There are certain kinds of strings that are visible to the user that are not translatable:

command, action and flag names
state names
database table and view names
database table column headings
action name used in the audit log, the mail notifications, and the authority and interest tables
the type field in the config database table

DBCS LIMITATIONS

1. The administration tools for the TeamConnection Server expect SBCS characters as the reply for Yes (y) and No (n).
2. The administration tools for the TeamConnection Server have the following limitations for DBCS:
   a. The *.ld files (authority, interest, cfgcomproc and cfgrelproc) in the family account can accept DBCS characters in the first field for each entry. The maximum size for this field is 15 bytes.
   b. The config.ld file in the family account can accept DBCS characters in the following fields (the positions are defined from left to right):

      Field position 1 ("Field Type"): limit is 15 bytes

      Field position 2 ("Value"): limit is 15 bytes

      Field position 6 ("Description"): limit is 63 bytes

   c. The chfield program can accept only SBCS characters in the following fields:

      CMD attribute

      DB Column Name

   d. The chfield program can accept DBCS characters in the following fields:

      Field label: limit is 15 bytes

      Title label: limit is 15 bytes

      Type: must be a valid type defined in config.ld (limit is 15 bytes).
3. The TeamConnection Commands Reference manual, in Appendix A, "Querying the TeamConnection database", shows the datatype and the size limit for the attributes of the TeamConnection objects; however, the actual size limit for many of the character attributes

is smaller than the specified limit. For example, the field "login" in the "Users" table shows that the limit is 31 bytes, but in reality only 15 characters (SBCS or DBCS) can be stored in that field. The fields affected are usually related to names, such as the User login, the Component name, etc.

If you specify a string that has DBCS characters and that the size of the string goes beyond the limit, then the following error message will be displayed by the TeamConnection server:

```
1 -149 Your request cannot be completed.
  The attribute flag argument    xxx is not valid.
```

4. Warning on the use of 0x7C as a second byte in a DBCS character

   The Ox7C character corresponds to the vertical bar ('|') which in TeamConnection is interpreted as a field separator when dealing with reports and with handling windows and fields in the GUI.

   You can use this value as the 2nd byte of a DBCS character, however, when the data that contains this 2nd byte is handled in a TeamConnection client that has an SBCS code page (and not a DBCS code page), then, the output shown by the client may be displaced, that is, the 0x7C value will be interpreted as the field separator. Moreover, this situation will apply for any string in the *.ld files and in the configurable fields.

INSTALLATION, ADMINISTRATION AND RUNTIME ISSUES

INSTALLATION ISSUES RELATED TO NLS AND DBCS

The installation process for TeamConnection is similar in UNIX, in OS/2 and Windows with respect to NLS. The similarities and the differences are explained in the following sections.

After the installation process, the executable code and the language related files will be installed in separate directories that are system wide, that is, they are not exclusive to one account.

When a TeamConnection family is created, several files are copied into the directory for the TeamConnection family; several of these files contain language sensitive information (such as the config.ld file and the files in the chfField directory). The family administrator can modify these files for the specific family; these files are not shared with other families.

Using a similar directory structure across all the platforms

Even though there are differences in the NLS facilities that are available from the UNIX and the Intel (OS/2 and Windows) platforms the installation of TeamConnection in these platforms create a similar directory structure which top directory is shown below (using the default directory):

AIX 4 /usr/teamc

HP-UX 10 /opt/teamc

OS/2 c:\teamc

Windows 3.1 c:\ProgramF\TeamC

Windows NT and 95 c:\Program Files\TeamConnection

Storing the language-independent files

The language-independent files for the TeamConnection code are stored in similar directories, as shown in the following example. The teamc server daemon (teamcd) is located in the subdirectory "bin", from the TeamConnection top directory as shown below:

AIX 4 /usr/teamc/bin

HP-UX 10 /opt/teamc/bin

OS/2 c:\teamc\bin

Windows 3.1 c:\ProgramF\TeamC\bin

Windows NT and 95 c:\Program Files\TeamConnection\bin

Storing the language-dependent files

In a similar way, the language-dependent files for the TeamConnection code are stored in a similar subdirectory structure, which is the subdirectory "nls" as parent and then the "msg" for messages and "cfg" for configuration items.

For example, the ISO US English message catalog will be stored as shown below, using the default location:

AIX 4 /usr/teamc/nls/msg/en_US (which really is a symbolic link to /usr/lib/nls/msg/en_US)

HP-UX 10 /opt/teamc/nls/msg/C (which really is a symbolic link to /usr/lib/nls/msg/C)

OS/2 c:\teamc\nls\msg\en_US

Windows 3.1 c:\ProgramF\TeamC\nls\msg\en_US

Windows NT and 95 c:\Program Files\TeamConnection\nls\msg\en_US

List of language-dependent files

The "nls" directory (see previous section for the complete path) contains the following subdirectories and files:

nls/msg/<locale>/
    All message catalog files, such as teamcv2.cat
    All help files.
    All resource DLLs for the GUI that are specific to a language.

nls/doc/<locale>/
    All documentation: PostScript, HTML, online, etc.

nls/cfg/<locale>/
    All configuration files, such as config.ld, and files for
     the configurable fields.
    The original teamc2 .ini file

Installation issues for UNIX

During the installation process for TeamConnection in UNIX, it is necessary to select the appropriate language version to install. The code is not bundled together with the language sensitive information. That is, there is an individual installable package just for the language sensitive information that could be installed independently.

Because AIX 4 and HP-UX 10 operating systems already include the explicit support for the XPG/4 I18N locale model, the TeamConnection installation process will not install additional files for this matter (as in OS/2 and Windows).

The message catalog that contains the language sensitive information is located by the executable code by means of the combination of the NLSPATH and LANG environment variable. By default, this variable is set to:

```
set NLSPATH=/usr/lib/nls/msg/%L/%N
```

Where:

%L	is a variable that at runtime represents the value of the LANG environment variable; it must be in uppercase.
%N	is a variable that at runtime represents the name of the message catalog to be used; it must be in uppercase.

Installation issues with OS/2 and Windows

During the installation process for TeamConnection in OS/2 and Windows, , it is necessary to select the appropriate language version to install. The code and the language sensitive information is bundled together in a package and it is installed appropriately. That is, there is not an individual package just for the language sensitive information that can be installed independently.

Because the OS/2 and Windows operating systems do not include at this moment explicit support for the XPG/4 I18N locale model, the TeamConnection installation process will install the necessary support for this model if necessary.

The message catalog that contains the language sensitive information is located by the executable code by means of the NLSPATH environment variable. By default, this variable is set to:

```
set NLSPATH=c:\teamc\nls\%N
```

Where:

c:\teamc	represents the appropriate drive and top directory where the TeamConnection code is installed in your system
nls	is the directory that contains the NLS related files
%N	is a variable that at runtime represents the name of the

message catalog to be used; it must be in uppercase.

FAMILY ADMINISTRATION ISSUES

A family should use the same language all the time

Although technically it could be possible for a family to be created using the en_US locale and then change it later on to another language, we consider that this process has the potential to cause a lot of confusion with the users, specially for the mapping of code points.

Therefore, this is treated as a limitation and if the customers try it, it is at their own risk and we will not help them. However, the customer may decide to delete the family, change the language by reinstalling the code for Intel and specify the new language, or to install the new language message catalogs for UNIX and change the LANG variable, and then create a new family to use the new setting.

This decision affects the arrangement of the subdirectories of a family: there is no provision in neither UNIX or Intel to have language dependent directories inside the family directory.

The following sections contain examples that will clarify this point.

UNIX

An AIX customer installed the TeamConnection server, using the en_US locale. The config.ld file (which is language dependent) resides in:

/usr/lib/nls/cfg/en_US/config.ld

The "testfam" TeamConnection family is created, and the config.ld file is copied from the system directory mentioned above, to the top directory of the family:

/home/testfam/config.ld

Notice that there is no "/home/testfam/en_US/config.ld" path.

Intel

An OS/2 customer installed the TeamConnection server, using the en_US locale. The config.ld file (which is language dependent) resides in:

c:\teamc\nls\cfg\en_US\config.ld

The "testfam" TeamConnection family is created, and the config.ld file is copied from the system directory mentioned above into the top directory of the family:

c:\testfam\config.ld

Notice that there is no "c:\testfam\en_US\config.ld" path.

CLIENT RUNTIME ISSUES

A client should use the same language all the time

Although technically it could be possible for a TeamConnection client to be installed with one language (such as the IBM-850 code page in OS/2 or the en_US locale in AIX) and then change the language in the middle, we consider that this process has the potential to cause a lot of confusion with the users, specially for the mapping of code points with the teamc20.ini file, as explained below.

In the UNIX platforms, thanks to the use of the LANG variable, it would be possible to install additional message catalogs for other languages and the user could setup the language to use by setting the variable LANG. However, the teamc20.ini file for the GUI will NOT be changed, and this file may contain characters that were valid in the original setup but that cannot be displayed in the new setup.

Because the Intel platforms do not provide the LANG variable, then it is not possible to have message catalogs for multiple languages for TeamConnection. This means that if the customer decides to change the language then it is necessary to reinstall the code specifying the new language.

The following sections contain examples that will clarify this point.

UNIX

A Japanese AIX customer installs the TeamConnection client using the ja_JP and en_US message catalogs.

The teamc20.ini file (which is language dependent) resides in:

/usr/lib/nls/cfg/ja_JP/teamc2 .ini
/usr/lib/nls/cfg/en_US/teamc2 .ini

The customer uses the Japanese GUI for the first time (LANG=ja_JP), and the GUI detects that the following file does not exist:

$HOME/teamc2 .ini

The customer uses the GUI and creates several entries written in Japanese in the task list which are stored in the teamc20.ini file.

The customer exists the GUI.

The user invokes the GUI again, and the GUI detects that the teamc20.ini file exists in $HOME and therefore the GUI uses it, and does not try to overwrite it with the file in the directory /usr/lib/nls/cfg/ja_JP.

The customer decides then to switch the locale to en_US, by setting LANG=en_US, exits and logs in again.

The user brings up the English TeamConnection GUI and now the task list shows entries that may not be legible because their original code points were set with the ja_JP locale.

Intel

A Japanese OS/2 customer installs the TeamConnection client and specifies the ja_JP language only, because she cannot install multiple languages. The code page is IBM-932.

The PATH and the NLSPATH variables point to the ja_JP directories.

The teamc20.ini file (which is language dependent) resides in:

c:\teamc\nls\cfg\ja_JP\teamc2 .ini

The customer uses the Japanese GUI for the first time (LANG=ja_JP), and the GUI detects that the following file does not exist:

c:\os2\teamc2 .ini

The GUI copies the original teamc20.ini file from the appropriate ja_JP directory ...

c:\teamc\nls\cfg\ja_JP\teamc2 .ini

... into ...

c:\os2\teamc2 .ini

The customer uses the GUI and creates several entries written in Japanese in the task list, and this list is stored in the teamc20.ini file.

The customer exists the GUI.

The user invokes the GUI again, and the GUI detects that the teamc20.ini file exists in c:\os2 and therefore the GUI uses it, and does not try to overwrite it with the file in the directory c:\teamc\nls\cfg\ja_JP.

The customer decides then to switch the locale to en_US, by uninstalling the TeamConnection client and reinstalling it again specifying now the language en_US, and reboots.

The PATH and the NLSPATH variables are updated and they do not point to the non-existing ja_JP directories, but point to the new en_US directories.

If the customer keeps the same code page, IBM-932, then when the user brings up the English TeamConnection GUI, the task list shows entries that are legible because their original code points were set with the IBM-932 code page.

If the customer changes the code page, let's say to IBM-850, then when the user brings up the English TeamConnection GUI the task list shows entries that may not be legible because their original code points were set with the IBM-932 code page.

MISCELLANEOUS TOPICS THAT ARE SPECIFIC TO OPERATING SYSTEMS

PROBLEM RESOLUTION

Potential problems if the language locale is not set properly

Potential problems if the language locale is not set properly:

Unexpected message catalogs would be picked up.

Note: Default hard coded messages are in English.

GUI labels, titles, and messages could be unreadable.

Data conversion errors could be resulted.

If any of the above listed presents the symptom of the problem, verify the locale and reset appropriately.

When using LANG=C in AIX, the message catalogs are ignored!

In the IBM red book "AIX 3.2 National Language Support", GG24-3850, says in page 16: "when the default C locale is the current locale, the hardcoded default message catalogs in the executable code will be used instead of the translated message file." The main point for confusion is that there is a directory in /usr/lib/nls/msg/C which is totally ignored! Thus, the user can place the message catalogs there, but they will never be used! This means that if the TeamConnection Server is using the LANG=C locale, then the output for "teamc report -testServer" will always be that the message catalog is not available and thus, the server will use the internal English messages which are embedded during compilation and used as default.

Using misc/tcmsgcat to verify the NLS message catalog settings

The utility "misc/tcmsgcat.aix" provided in the CD-ROM for TeamConnection can be used to verify that all the NLS variables related to the message catalog for TeamConnection are properly configured and that the message catalog itself can be opened; furthermore, the first message in the message catalog is read, and it contains the name of the language used with the strings in the message catalog.

A sample output with the English en_US message catalog is shown in Figure 7 on page 26.

```
Testing for the existence of the message catalog: teamcv2.cat

The value of the LANG variable is:              en_US
The value of the LC_MESSAGES variable is: not explicitly defined
The value of the NLSPATH variable is:
/usr/lib/nls/msg/%L/%N

SUCCESS! the message catalog teamcv2.cat was opened.

Printing from the catalog, the message in position: 1
English

You can use 'dspmsg msg.cat -s 1 <number>' to display other messages
Example, the first message in teamcv2.cat does not require arguments:
   dspmsg teamcv2.cat -s 1 1

Example, the second message requires 5 arguments (add dummy s ):
   dspmsg teamcv2.cat -s 1 2 s  s1 s2 s3 s4 s5
```

Figure 7. Sample output of "tcmsgcat.aix"

Actually, you can use the TeamConnection utility "showmsg" which is a front end to the AIX utility "dspmsg"; for more information see “Using misc/showmsg to display a given message from the catalog.”

Using misc/showmsg to display a given message from the catalog

The utility "misc/showmsg" provided in the CD-ROM for TeamConnection can be used to display a particular message from the TeamConnection message catalog.

This utility is actually more helpful for our development team, because in order to display a given message we need to do the to find out the actual number to use with the utility showmsg. The utility uses dummy positional input variables which will be replaced, if needed, when displaying the message.

If we want to display the message "0011-209" from the message catalog, we need to identify which is the mnemonic and the index for the message, which can be obtained from the file src/inc/tcmsgdef.hpp which is generated during the construction of the message catalog. For example, the search for "0011-209" gives the following entry:

```
... "msgLicenseMonitorInvalidDateRange","  11-2 9",253 );
                                                                    --> index
                                                                --> message number
                                        --> mnemonic
```

Then, we use the index 2530 with the utility "showmsg" to verify if indeed we obtain the text for the message 0011-209, as shown in Figure 8.

```
message 253  from message catalog: /usr/lib/nls/msg/en_US/teamcv2.cat
  11-2 9 The date range from s1 to s2
            is not valid. The end date must be greater than the begin date.
```

Figure 8. Sample output of "showmsg"

Problems when the UNIX teamcgui window does not show up

When using UNIX, if you start the GUI by using "teamcgui" or "teamcgui &" (to send the process to the background), it takes several seconds for the GUI to actually appear in your display.

However, if after several minutes of waiting and still you do not see a window for the TeamConnection window, you can kill the runaway process:

```
ps -ef | grep teamcgui
Identify the process id for teamcgui
kill -15 processIdforTeamcgui
```

Then check the following:

Ensure that the DISPLAY variable is properly set to your X server.

The teamc20.ini file might be corrupted. You can delete it or rename it, and start again teamcgui.

TOPICS RELATED TO THE MESSAGE CATALOG USED BY TEAMCONNECTION

Using the "teamc report" command to test the NLS settings

You can use the commands to verify if TeamConnection is accessing the message catalog:

For the server: "teamc report -testServer"

A sample output is shown in Figure 9 on page 28.

```
Connect to Family Name:                tcocto
Server TCP/IP Name:                     carcps22.raleigh.ibm.com
Server IP Address:                     9.67.238.38
Server TCP/IP Port Number:             13

Server Specific Information ---------------------------------
Product Version:                        2. .4
Operating System:                       AIX
Message catalog language:              English
Server Mode:                            non-maintenance
Authentication Level:                  HOST_ONLY
```

Figure 9. Sample output of "teamc report -testServer"

For the client: "teamc report -testClient"

A sample output is shown in Figure 10.

```
Product Version:                  2.0.5
Message catalog language:         English
Environment variables:
Variable:          Setting:
------------------- ----------------------------------
TC_USER            rivera
TC_BECOME           rivera
TC_FAMILY          tcocto@carcps22@1300
TC_RELEASE          v205
TC_TOP
TC_WORKAREA
TC_BUILDPOOL
```

Figure 10. Sample output of "teamc report -testClient"

How to know when TeamConnection is not using the message catalog

If the output of the commands "teamc report -testServer" or "teamc report -testClient" produce an output line that is similar to the following:

```
Message catalog language:          English (Internal)
```

Then, this means that TeamConnection cannot find the message catalog and is using the internal default English messages that are embedded in the executables during compilation. The most likely cause for this problem is an improper definition of the NLSPATH environment variable.

SETTING AND VERIFY THE CURRENT LOCALE SETTING IN UNIX

To set the language locale to something different from the current setting, issue the following depending on what SHELL environment you are in:

- Korn Shell

  export LANG=Ja_JP (or) export LANG=ja_JP

- To verify the current locale setting, issue the following command:

  locale

  The output of the command looks something similar to the following:

  ```
  LANG=Ja_JP
  LC_COLLATE="Ja_JP"
  LC_CTYPE="Ja_JP"
  LC_MONETARY="Ja_JP"
  LC_NUMERIC="Ja_JP"
  LC_TIME="Ja_JP"
  LC_MESSAGES="Ja_JP"
  LC_ALL=
  ```

HOW TO ENTER SBCS EXTENDED CHARACTERS USING AN ENGLISH KEYBOARD

It is possible to enter any SBCS character defined in the current code page, including characters that are not available from the layout of your keyboard in the following environments:

In OS/2 windows.

In the DOS command prompt in Windows

In AIX "aixterm".

In AIX "dtterm" for non-English locales.

Strange enough, if using English locales, you cannot enter these extended characters, but if using Ja_JP for example, you can!

For example, the US English keyboard does not have accented characters, but it is possible to enter them by means of the following procedure:

1. Ensure that the "Num Lock" mode is activated in the numeric keypad, which is located in the right side of the standard keyboard. If your keyboard has a LED indicator for this mode, then it should be on.
2. Press and hold the Alt key.
3. Enter the 3 digits that correspond to the code point in decimal, such as 252: entering the digits 2, followed by 5 and 2.
4. Finally, release the Alt key.

This action can be summarized as "Alt-XXX", such as "Alt-252" for the above example.

Of course, if you have a keyboard that can enter extended characters, then you do not have to use the Alt-XXX method.

HOW TO SPECIFY NLS SETTINGS FOR AIX GUI ON AIXTERM/DTTERM WINDOWS

If you are going to use the TeamConnection line commands and the TeamConnection GUI in Motif in AIX and using different locales, then you may need to open a new window with the aixterm or dtterm (which is provided by the Common Desktop Environment) utilities.

Although these utilities provide a similar function, they differ considerably with respect to the NLS support.

Using aixterm

1. Open an X-window session with aixterm with the desired locale (code-page) and the appropriate font. The -T "string" parameter for aixterm will determine the title for the window.

   Some examples are shown below:

For IBM-85 :
```
aixterm -T "IBM-85 "        -lang En_US -fn Rom14 &
```

For ISO-8859-1:
```
aixterm -T "ISO-8859-1" -lang en_US -fn Rom14.iso1 &
```

For Ja_JP:
```
aixterm -T "Ja_JP"          -lang Ja_JP -fn  gothic- -19  &
```

For ja_JP:
```
aixterm -T "ja_JP"          -lang ja_JP -fn  gothic- -19  &
```

2. Perform the quick test for the code page, by entering Alt-252. If code page IBM-850, then you will see 3-superscript; if code page ISO-8859-1, you will see u-umlaut.
3. For DBCS, ensure that the aixterm window has the keyboard status area located at the bottom left corner of the window; also, ensure that you can enter SBCS and DBCS characters.

Using dtterm

1. Open an X-window session with dtterm with the desired locale (code-page) and the appropriate font. The -name "string" parameter for dtterm will determine the title for the window.

   Some examples are shown below:

For IBM-85 :
```
LANG=En_US dtterm -name "IBM-85 "        -fn Rom14 &
```

For ISO-8859-1:
```
LANG=en_US dtterm -name "ISO-8859-1" -fn Rom14.iso1 &
```

For Ja_JP:
```
LANG=Ja_JP dtterm -name "Ja_JP"          -fn  gothic- -19  &
```

For ja_JP:
```
LANG=ja_JP dtterm -name "ja_JP"          -fn  gothic- -19  &
```

2. Perform the quick test for the code page, by entering Alt-252. If code page IBM-850, then you will see 3-superscript; if code page ISO-8859-1, you will see u-umlaut.
3. For DBCS, ensure that the aixterm window has the keyboard status area located at the bottom left corner of the window; also, ensure that you can enter SBCS and DBCS characters.

CREATING A TEAMCGUI RESOURCE FILE FOR AIX

You can create a resource file for the TeamConnection GUI for AIX, called "Teamcgui" (the first T of this file name is in uppercase, and the rest of the string is in lowercase) in your home directory to specify the appropriate font.

For example, for Japanese, you can specify the following small size font:

```
fontList:              gothic- -19 :
XmText fontList:  gothic- -19 :
XmList fontList:  gothic- -19 :
```

You could use these other fonts for Japanese:

```
Small font:      gothic- -19 :
Medium font:    mincho- -27 :
Large font:      mincho- -35 :
```

Important Note: You must specify the colon at the end of the font specification; if the colon is not present, such as in:

```
gothic- -19
```

instead of:

```
gothic- -19 :
```

otherwise the keyboard status area will not be shown in the bottom left corner of the TeamConnection GUI.

USING ICONV TO CONVERT FILES TO OTHER CODE PAGES

The source files for the message catalog and other source files such as config.ld for TeamConnection were created in US English, using code page IBM-850 in OS/2. Then these files were sent to the translation centers and the translation was done using the corresponding code pages for OS/2.

This means that in order to support the code pages in UNIX, we needed to convert the code pages for certain files. This conversion can be accomplished by installing the appropriate locales and using the utility "iconv", as shown below:

```
from EN_US to en_US:
iconv -f IBM-85  -t ISO8859-1 input-file > output-file

from Ja_JP to ja_JP:
iconv -f IBM-932 -t IBM-eucJP input-file > output-file

ko_KR:
not needed
```

from Pt_PT to pt_BR:
iconv -f IBM-85 -t ISO8859-1 input-file > output-file

zh_CN:
not needed

from Big-5 to zh_TW:
iconv -f big5 -t IBM-eucTW input-file > output-file

HOW TO INSTALL ADDITIONAL LOCALES FOR AIX

1. Insert the appropriate CD-ROM with the installation images for the AIX operating system.
2. Login as root
3. Invoke smit (or smitty)
4. Select: System Environments
5. Select: Manage Language Environment
6. Select: Add Additional Language Environments
7. Select: CULTURAL convention to install

   This item includes the locale information and the conversion files between code pages related to the locale.

   User F4 to list the choices from the CD-ROM. The following entries were installed in our build and testing machines:

   IBM-85 English (USA) [En_US]
   ISO8859-1 English (USA) [en_US]
   IBM-eucCN Chinese (Simplified EUC) [zh_CN]
   IBM-eucTW Chinese (Traditional) [zh_TW]
   IBM-eucJP Japanese (EUC) [ja_JP]
   IBM-932 Japanese (PC) [Ja_JP]
   IBM-eucKR Korean [ko_KR]
   ISO8859-1 Portuguese (Brazil) [pt_BR]

8. Select: LANGUAGE translation to install

   This item includes the message catalogs that will be stored in /usr/lib/nls/msg.

   Use F4 to list the choices from the CD-ROM.

9. Press enter to install the desired locales and/or message catalogs.
10. Press F10 to exit from smit.

HOW TO CHANGE THE KEYBOARD FOR AN AIX HOST

Note: You MUST use the RISC keyboard and NOT a PC keyboard (which will not work, you will need to telnet from another host to reconfigure smit, shutdown, change keyboard again and boot)

1. Insert the appropriate CD-ROM with the installation images for the AIX operating system.
2. Login as root
3. Invoke smit (or smitty)
4. Select: System Environments
5. Select: Manage Language Environment
6. Select: Change/Show Primary Language Environment
7. Select: Change/Show Cultural Convention, Language, or Keyboard
8. Enter the desired values for the following items:

   ```
   Primary CULTURAL convention
   Primary LANGUAGE translation
   Primary  KEYBOARD
   ```

   Use F4 to list the choices from the CD-ROM.

   The choices must match, otherwise there will be an error. For example, if installing a Japanese keyboard but keeping the English cultural convention, then this combination does not match and will not succeed.
9. Press enter to make the change.
10. Exit smit, by pressing F10.
11. Shutdown the machine and DO NOT REBOOT yet:

    ```
    shutdown -F
    ```
12. Change the physical keyboard.
13. Boot the machine
14. Verify that you can use the new keyboard

HOW TO INSTALL A NON-ENGLISH PC KEYBOARD TO AN X-STATION

The AIX hosts that run on RISC boxes require keyboards that are not compatible with the PC standard keyboards.

This means that if you have a Japanese keyboard for a PC, for example, you cannot use this keyboard with a RISC machine. That is, you must get a Japanese keyboard that is specifically designed for the RISC machine.

However, it is possible to connect a PC keyboard to an X-station which can communicate with a RISC server. Follow these instructions to accomplish this task:

1. The PC <language> keyboard (where: <language> is the supported language) C Posix Language, and C Posix cultural convention seems to be the most versitile for using different keyboard definitions from X-stations.

   This convention can be set via smitty:

   a. smitty

   b. Devices

   c. Xstation Configuration

   d. Add an Xstation

   e. Select the desired model

   f. From the "Add an Xstation" window, specify the following values that are related to NLS settings.

   ```
   CULTURAL convention      <the desired locale>
   LANGUAGE                   ISO8859-1   C  (POSIX)        [C]
   KEYBOARD                   ISO8859-1   C (POSIX) keyboard   [C]
   ```

   Then press enter to activate.

   g. Exit smitty

2. Once the correct keyboard definition is configuered for the X-station changing cultural convention, and language settings are as simple as changing the LC_ALL and LOCALE environment variables.

3. Try not to use the dtlogin panel form X-stations.

   When configuring the X-station do not use the /etc/dt/config instructions from x_sta_mgr install for CDE. Use the default x session to bring up a dtsession from /usr/dt/bin/dtsession instead, after setting the desired language environment variables

   The is because dt has 2 required fixes for dt to work in an X-station configuration from the default system. These fixes are not guaranteed to fix all the problems, though.

How to specify a different keyboard mapping

Let's suppose that you have installed the Ja_JP locale and you only have an English keyboard. You can still enter Japanese characters using the English keyboard by doing the following:

export LANG=Ja_JP

xmodmap /usr/lpp/X11/defaults/xmodmap/$LANG/keyboard

You may need to find out what is the actual keyboard mapping in order to activate the different input modes and to enter Japanese characters.

COPYRIGHTS, TRADEMARKS AND SERVICE MARKS

The following terms used in this technical report, are trademarks or service marks of the indicated companies:

TRADEMARK, REGISTERED TRADEMARK OR SERVICE MARK	COMPANY
AIX, OS/2, IBM, TeamConnection RISC System/6	IBM Corporation
UNIX, XPG/4	X/Open Co., Ltd.
OSF, OSF Motif, Common Desktop Environment, CDE	Open Software Foundation, Inc.
HP-UX	Hewlett-Packard Company
Microsoft, Windows	Microsoft Corporation
X Window System	Massachusetts Institute of Technology
Unicode	Unicode Inc.
PostScript	Adobe Systems, Inc.
Intel	Intel Corp.

END OF DOCUMENT