

CHAPTER 12

Database Collations and International Languages

About this chapter

This chapter describes how to ensure that Adaptive Server Anywhere treats your data according to the language in which you work.

A **collation** is an ordering for a particular character set. Choosing the proper collation ensures that the sorting and comparison operations produce the proper results for the language in which the data is stored.

This chapter describes the built-in collations provided with Adaptive Server Anywhere and describes how to construct custom collations. It also describes other aspects of the software that are of particular interest to users of languages other than English.

Contents

Topic	Page
Introduction	290
Introduction to character sets and collations	293
Using character set translation	297
Choosing a database collation	300
Creating databases with custom collations	309
Character set translation details	315

Introduction

This section provides an introduction to the issues you may face when working in an environment that uses more than one character set, or when using languages other than English.

Overview of language issues

Pieces in the client/server puzzle

Database users working at client applications may see or access text that comes from several possible sources:

- ◆ **Data in the database** Strings and other text data are stored in the database. The database server processes these strings when responding to requests.

For example, the database server may be asked to supply all the last names beginning with a letter ordered less than N in a table. This request requires string comparisons to be carried out, and assumes a character set ordering.
- ◆ **Database server software messages** Applications can cause database errors to be generated. For example, an application may submit a query that references a column that does not exist. In this case, the database server returns a warning or error message. This message is held in a **language library**, which is a DLL or shared library called by Adaptive Server Anywhere.
- ◆ **Client application** The client application interface displays text, and internally the client application may process text.
- ◆ **Client software messages** The client library uses the same language library as the database server to provide messages to the client application.
- ◆ **Operating system** The client operating system has text displayed on its interface, and may also process text.

For a satisfactory working environment, all these sources of text must work together. Loosely speaking, they must all be working in the user's language, but what exactly is involved in this?

Pieces in the character set puzzle

There are several distinct aspects to character storage and display by computer software:

- ◆ Each piece of software works with a **character set**. A character set is a set of symbols, including letters, digits, spaces and other symbols.

- ◆ To handle these characters, each piece of software employs a character set **encoding**, in which each character is mapped onto one or more **bytes** of information, typically represented as hexadecimal numbers.
- ◆ Characters are printed or displayed on a screen using a **font**, which is a mapping between characters in the character set and their appearance.
- ◆ Operating systems also use a **keyboard mapping** to map keys or key combinations on the keyboard to characters in the character set.
- ◆ Any software that needs to sort characters (for example, list names alphabetically) uses a **collation**. A collation is a combination of a character encoding (a map between characters and hexadecimal numbers) and a **sort order** for the characters. There may be more than one sort order for each character set; for example, a case-sensitive order and a case-insensitive order.

Tasks for the database server

The database server receives strings from client applications as streams of bytes. It associates these bytes with characters according to the collation specified when the database was created. If the data is held in an indexed column, the index is sorted according to the sort order of the collation.

Client side tasks

It is up to the operating system of the computer on which the client application is running to handle the following aspects of character strings:

- ◆ Which character is stored when a particular key on the keyboard is pressed.
- ◆ What a character looks like on your computer screen.
- ◆ What characters are available to the application.
- ◆ What byte or combination of bytes is stored for each character.

Who needs to read this chapter?

If your character data consists of characters in the English alphabet, you probably do not need to use this chapter.

If you have an existing setup that does not display language problems, you probably do not need to read this chapter.

How Adaptive Server Anywhere addresses language issues

Adaptive Server Anywhere has support for all the language issues discussed in the previous sections.

- ◆ **Data in the database** When you create a database, you can select a collation of your choice. By choosing a collation that is suitable for the data in your database, you ensure that query results and so on are sorted properly, and that string comparisons are executed properly.
ℳ For more information, see "Choosing a database collation" on page 300.
- ◆ **Message strings** The Adaptive Server Anywhere language library holds messages in the language of the software. If you have purchased an English language version of Adaptive Server Anywhere, the messages are in English, and so on.
- ◆ **Character set translation** Adaptive Server Anywhere can be configured to carry out character set translation. This means that message and query results are passed back to the client application in the character set requested by the client, even if this is different from the character set used in the database collation.
ℳ For more information, see "Using character set translation" on page 297.
- ◆ **Connection strings** Connection strings are processed outside any database, and handling connection strings with proper attention to character sets is a separate topic.

The other sections in this chapter provide detailed descriptions of character set and language issues.

Introduction to character sets and collations

When you create a database, you specify a collating sequence or **collation** to be used by the database. A collation is a character set, and a sorting order for characters in the database. Whenever the database compares strings, sorts strings, or carries out other string operations such as case conversion, it does so using the collating sequence. The database carries out sorting and string comparison when statements such as the following are submitted:

- ◆ Queries with an ORDER BY clause.
- ◆ Expressions that use string functions, such as LOCATE, SIMILAR, SOUNDEX.
- ◆ Conditions using the LIKE keyword.

The database also uses character sets in identifiers (column names and so on). In deciding whether a string is a valid or unique identifier, the database uses the database collation.

Single-byte character sets and code pages

Many languages have few enough characters to be represented in a single-byte character set. In such a character set, each character is represented by a single **byte**: a two-digit hexadecimal number.

At most, 256 characters can be represented in a single-byte. No single-byte character set can hold all of the characters used internationally, including accented characters. IBM solved this problem by developing a set of **code pages**, each of which describes a set of characters appropriate for one or more national languages. For example, code page 869 contains the Greek character set, and code page 850 contains an international character set suitable for representing many characters in a variety of languages.

Adaptive Server Anywhere supplies a set of single-byte collations (code pages and collation orderings) suitable for many languages of European origin.

Upper and lower pages

With few exceptions, characters 0 to 127 are the same for all the single-byte code pages. The mapping for this range of characters is called the **ASCII** character set. It includes the English language alphabet in upper and lower case, as well as common punctuation symbols and the digits. This range is often called the **seven-bit** range (because only seven bits are needed) or the **lower** page. The characters from 128 to 256 are called **extended characters**, or **upper** code-page characters, and vary from code page to code page.

There is generally no problem with code page compatibility if the only characters used are from the English alphabet, as these are represented in the ASCII portion of each code page (0 to 127). However, if other characters are used, as is generally the case in any non-English environment, there can be problems if the database and the application use different code pages.

Example

Suppose a database holding French language strings uses code page 850, and the client operating system uses code page 437. The character Å (upper case A grave) is held in the database as character 183 (hexadecimal B7). In code page 437, character 183 is a graphical character. The client application receives this byte and the operating system displays it on the screen, the user sees a graphical character instead of an A grave.

Code pages in Windows and Windows NT

For PC users, the issue is complicated because there are at least two code pages in use on most PC's. MS-DOS, as well as character-mode applications (those using the console or "DOS box") in Windows 95 and Windows NT, use code pages taken from the IBM set. These are called **OEM code pages** (Original Equipment Manufacturer) for historical reasons.

Windows operating systems do not require the line drawing characters that were held in the extended characters of the OEM code pages, so they use a different set of code pages. These pages were based on the ANSI standard and are therefore commonly called **ANSI code pages**.


Adaptive Server Anywhere supports collations based on both OEM and ANSI code pages.

Example

Consider the following situation:

- ◆ A PC is running the Windows 95 operating system with ANSI code page 1252.
- ◆ The code page for character-mode applications is OEM code page 437.
- ◆ Text is held in a database created using the collation corresponding to OEM code page 850.


An upper case A grave in the database is stored as character 183. This value is displayed as a graphical character in a character-mode application. The same character is displayed as a dot on a Windows application.

 For information about choosing a single-byte collation for your database, see "Choosing a database collation" on page 300.

Multibyte character sets

Some languages, such as Japanese and Chinese, have many more than 256 characters. These characters cannot all be represented using a single byte, but can be represented in multibyte character sets. In addition, some character sets use the much larger number of characters available in a multibyte representation to represent characters from many languages in a single, more comprehensive, character set.

Multibyte character sets are of two types. Some are **variable width**, in which some characters are single-byte characters, others are double-byte, and so on. Other sets are **fixed width**, in which all characters in the set have the same number of bytes. Adaptive Server Anywhere supports only variable-width character sets.

 For information on the multibyte character sets, see "Using multibyte collations" on page 307.

Displaying your current character settings

Each operating system has its own system for handling character sets, encodings, and collation sequences. To find out information about the current settings on your operating system, you can:

- ◆ At a system command prompt, type **chcp** to display the current code page. On Windows and Windows NT PCs, this returns the OEM code page.
- ◆ In Windows 3.x, Windows 95, and Windows NT, see the Regional Settings in the Control Panel. The Regional settings correspond to an ANSI code page.

Sorting characters using collations

A **collation** is a sorting order for characters in a character set encoding or code page. The collation sequence is based on the encoded value of the characters.

The collation sequence includes the notion of alphabetic ordering of letters, and extends it to include all characters in the character set, including digits and space characters.

Associating more than one character with each sort position

More than one character can be associated with each sort position. This is useful if you wish, for example, to treat an accented character the same as the character without an accent.

Two characters with the same sort position are considered identical in all ways by the database. Therefore, if a collation assigned the characters *a* and *e* to the same sort order, then a query with the following search condition:

```
WHERE col1 = 'want'.
```

is satisfied by a row for which **col1** contains the entry "went".

At each sort position, lower- and uppercase forms of a character can be indicated. For case-sensitive databases, the lower- and uppercase characters are not treated as equivalent. For case-insensitive databases, the lower- and uppercase versions of the character are considered equivalent.

Using character set translation

This section outlines how different components of Adaptive Server Anywhere handle code pages, and how character set translation enables systems where different components have different character sets to function properly.

Character set translation can be carried out among character sets that represent the same characters, but at different values. There needs to be a degree of compatibility between the character sets for this to be possible. For example, character set translation between EUC-JIS and Shift-JIS character sets, but not between EUC-JIS and OEM code page 850.

To enable character-set translation, you must start the database server using the `-ct` command-line option. For example:

```
dbeng6 -ct asademo.db
```

Avoiding character-set translation

There is a performance cost associated with character set translation. If you can set up an environment such that no character set translation is required, then you do not have to pay this cost, and your setup is simpler to maintain.

If you work with a single-byte character set and are concerned only with seven-bit ASCII characters (values 1 through 127), then you do not need character set translation. Even if the code pages are different in the database and on the client operating system, they are compatible over this range of characters. Many English-language installations will meet these requirements.

If you do require use of extended characters, there are other steps you may be able to take:

- ◆ If the code page on your client machine operating system matches that used in the database, no character set translation is needed for data in the database.

For example, in many environments it is appropriate to use the WinLatin1 collation in your database, which corresponds to the Windows NT code page in many environments.

- ◆ If you are able to use a version of Adaptive Server Anywhere built for your language, and if you use the WinLatin1 code page on your operating system, no character set translation is needed for database messages. The character set used in the Adaptive Server Anywhere message strings is code page 1252, which corresponds to WinLatin1.

Also, recall that character set translation takes place only if the database server is started using the `-ct` command-line switch.

ODBC code page translation

Adaptive Server Anywhere provides an **ODBC translation driver**. This converts characters between OEM and ANSI code pages. It allows Windows applications using ANSI code pages to be compatible with databases that use OEM code pages in their collations.

Not needed if you use ANSI character sets

If you use an ANSI character set in your database, and are using ANSI character set applications, you do not need to use this translation driver.

The translation driver carries out a mapping between the OEM code page in use in the "DOS box" and the ANSI code page used in the Windows operating system. If your database uses the same code page as the OEM code page, the characters are translated properly. If your database does not use the same code page as your machine's OEM code page, you will still have compatibility problems.

Embedded SQL does not provide any such code page translation mechanism.

Sybase Central and Interactive SQL code page translation

Interactive SQL and Sybase Central both have OEM to ANSI internal code page translation if the database uses an OEM character set. As with the ODBC translation driver, there is an assumption that the OEM code page on the local machine is the same as that in the database.

For Interactive SQL, you can turn off the translation if you wish, by setting the Interactive SQL option **CHAR_OEM_Translation** to a value of OFF.

🔗 For more information on OEM to ANSI character set translation in Interactive SQL, see "CHAR_OEM_TRANSLATION option" on page 146 of the book *Adaptive Server Anywhere Reference Manual*.

Character translation for database messages

Error and other messages from the database software are held in a **language library**. Localized versions of this library are provided with localized versions of Adaptive Server Anywhere. The messages for each language assume an ANSI code page.

Some database messages have placeholders that are filled in from the database. For example, if you execute a query with a column that does not exist, the returned error messages is:

Column *column-name* not found

where *column-name* is filled in from the database.

If the database collation uses a character set that is different from the Language DLL but compatible with it (such as EUC-JIS compared to Shift-JIS), then the database server automatically translates database messages into the database collation prior to sending to the client. If necessary, further character set translation from database to client is carried out in order to ensure that the message reaches the client in the appropriate character set.

Messages are always translated, if necessary, into the database collation character set, regardless of whether the `-ct` command-line option is used. The `-ct` command-line option affects only the character set conversion on the way to the client.

Connection strings and character sets

Connection strings present a special case for character set translation. The connection string is parsed by the client library, in order to locate or start a database server. This parsing is done with no knowledge of the server character set or language.

The connection string is parsed as follows:

- 1 It is broken down into its *keyword = value* components. This can be done independently of character set, as long as you do not use the { curly braces } around CommLinks parameters. Instead, use the recommended (parentheses).
- 2 The server is located. The server name must be constructed from lower page (seven-bit) ASCII characters.
- 3 The DatabaseName or DatabaseFile parameter is interpreted in the server character set.
- 4 Once the database is located, the remaining connection parameters are interpreted according to its character set.

Choosing a database collation

You choose a collation when you create a database. The collation is an option on the CREATE DATABASE statement and in the Initialization utility.

Supplied collations

The following collations are supplied with Adaptive Server Anywhere. You can obtain this list by entering the following command at a system command line:

```
dbinit /l
```

Collation name	Type	Description
internal	OEM	Based on Code page 850
437	OEM	Code page 437, ASCII, United States
850	OEM	Code page 850, ASCII, Multilingual
852	OEM	Code page 852, ASCII, Slavic/Latin II
860	OEM	Code page 860, ASCII, Portugal
863	OEM	Code page 863, ASCII, Canada-French
865	OEM	Code page 865, ASCII, Norway
EBCDIC	ANSI	EBCDIC
437EBCDIC	OEM	Code Page 437, EBCDIC
437LATIN1	OEM	Code Page 437, Latin 1
437ESP	OEM	Code Page 437, Spanish
437SVE	OEM	Code Page 437, Swedish/Finnish
819CYR	ANSI	Code Page 819, Cyrillic
819DAN	ANSI	Code Page 819, Danish
819ELL	ANSI	Code Page 819, Greek
819ESP	ANSI	Code Page 819, Spanish
819ISL	ANSI	Code Page 819, Icelandic
819LATIN1	ANSI	Code Page 819, Latin 1
819LATIN2	ANSI	Code Page 819, Latin 2
819NOR	ANSI	Code Page 819, Norwegian
819RUS	ANSI	Code Page 819, Russian

Collation name	Type	Description
819SVE	ANSI	Code Page 819, Swedish/Finnish
819TRK	ANSI	Code Page 819, Turkish
850CYR	OEM	Code Page 850, Cyrillic
850DAN	OEM	Code Page 850, Danish
850ELL	OEM	Code Page 850, Greek
850ESP	OEM	Code Page 850, Spanish
850ISL	OEM	Code Page 850, Icelandic
850LATIN1	OEM	Code Page 850, Latin 1
850LATIN2	OEM	Code Page 850, Latin 2
850NOR	OEM	Code Page 850, Norwegian
850RUS	OEM	Code Page 850, Russian
850SVE	OEM	Code Page 850, Swedish/Finnish
850TRK	OEM	Code Page 850, Turkish
852LATIN2	OEM	Code Page 852, Latin 2
852CYR	OEM	Code Page 852, Cyrillic
855CYR	OEM	Code Page 855, Cyrillic
856HEB	OEM	Code Page 856, Hebrew
857TRK	OEM	Code Page 857, Turkish
860LATIN1	OEM	Code Page 860, Latin 1
861ISL	OEM	Code Page 861, Icelandic
862HEB	OEM	Code Page 862, Hebrew
863LATIN1	OEM	Code Page 863, Latin 1
865NOR	OEM	Code Page 865, Norwegian
866RUS	OEM	Code Page 866, Russian
869ELL	OEM	Code Page 869, Greek
920TRK	ANSI	Code page 920, Turkish
SJIS	ANSI	Japanese Shift-JIS Encoding
SJIS2	ANSI	Japanese Shift-JIS Encoding, Sybase Adaptive Server Enterprise-compatible
EUC_JAPAN	ANSI	Japanese EUC JIS X 0208-1990 and JIS X 0212-1990 Encoding
EUC_CHINA	ANSI	Chinese GB 2312-80 Encoding

Collation name	Type	Description
EUC_TAIWAN	ANSI	Taiwanese Big 5 Encoding
EUC_KOREA	ANSI	Korean KS C 5601-1992 Encoding
UTF8	ANSI	UCS-4 Transformation Format
ISO_1	ANSI	ISO8859-1, Latin 1
ISO_BINENG	ANSI	Binary ordering, English ISO/ASCII 7-bit letter case mappings
WIN_LATIN1	ANSI	Windows Latin 1, similar to ISO_1, ISO8859-1, with extensions
WIN_LATIN5	ANSI	A superset of ISO 8859-9 with characters also in 80-9F

ANSI or OEM?

Adaptive Server Anywhere collations are based on code pages that are designated as either ANSI or OEM. In most cases, use of an ANSI code page is recommended.

If you choose to use an ANSI code page, you must *not* use the ODBC translation driver in the ODBC data source configuration window.

If you choose to use an OEM code page, you must do the following:

- ◆ Choose a code page that matches the OEM code pages on your users' client machines.
- ◆ When setting up data sources for Windows-based ODBC applications, *do* choose the Adaptive Server Anywhere translation driver in the ODBC data source configuration.

The translation driver converts between the OEM code page on your machine and the ANSI code page used by Windows. If the database collation is a different OEM code page than the one on your machine, results may not be perfect.

Both Interactive SQL and Sybase Central detect whether the database collation is ANSI or OEM by checking the first few characters, and either enable or disable translation as needed.

ℳ For more information about code page translation in Interactive SQL, see "CHAR_OEM_TRANSLATION option" on page 146 of the book *Adaptive Server Anywhere Reference Manual*.

Supplied ANSI collations

Adaptive Server Anywhere provides the following ANSI collations:

- ◆ 920TRK
- ◆ SJIS
- ◆ SJIS2
- ◆ EUC_JAPAN
- ◆ EUC_CHINA
- ◆ EUC_TAIWAN
- ◆ EUC_KOREA
- ◆ UTF8
- ◆ ISO_1
- ◆ ISO_BINENG
- ◆ WIN_LATIN1
- ◆ WIN_LATIN5

The ISO_1
collation

ISO_1 is provided for compatibility with the Adaptive Server Enterprise default ISO_1 collation. The differences are as follows:

- ◆ The lower case letter sharp s (`\xDF`) sorts with the lower case s in Adaptive Server Anywhere, but after **ss** in Adaptive Server Enterprise.
- ◆ Ligatures are two characters combined into a single character. The ligatures corresponding to **AE** and **ae** (`\xC6` and `\xE6`) sort after **A** and **a** respectively in Adaptive Server Anywhere, but after **AE** and **ae** in Adaptive Server Enterprise.

The WIN_LATIN1
collation

WIN_LATIN1 is similar to ISO_1, except that Windows has defined characters in places where ISO_1 says "undefined", specifically the range 0x80-0xBF. The differences from Adaptive Server Enterprise's ISO_1 are as follows:

- ◆ The upper case and lower case Icelandic Eth (`\xD0` and `\xF0`) is sorted with D in Adaptive Server Anywhere, but after all other letters in Adaptive Server Enterprise.
- ◆ The upper case and lower case Icelandic Thorn (`\xD0` and `\xF0`) is sorted with T in Adaptive Server Anywhere, but after all other letters in Adaptive Server Enterprise.

- ◆ The upper-case Y-diaeresis (\x9F) is sorted with Y in Adaptive Server Anywhere, and case converts with lower-case Y-diaeresis (\xFF). In Adaptive Server Enterprise it is undefined and sorts after \x9E
- ◆ The lower case letter sharp s (\xDF) sorts with the lower case s in Adaptive Server Anywhere, but after ss in Adaptive Server Enterprise.
- ◆ Ligatures are two characters combined into a single character. The ligatures corresponding to **Æ** and **æ** (\xC6 and \xE6) sort after **A** and **a** respectively in Adaptive Server Anywhere, but after **Æ** and **æ** in Adaptive Server Enterprise.
- ◆ The ligatures corresponding to OE and oe (\x8C and \x9C) sort with O in Adaptive Server Anywhere, but after OE and oe in Adaptive Server Enterprise.
- ◆ The upper case and lower case letter S with caron (\x8A and \x9A) sorts with S in Adaptive Server Anywhere, but is undefined in Adaptive Server Enterprise, sorting after \x89 and \x99.

Supported OEM collations

The following table shows the built-in collations that correspond to OEM code pages. The table and the corresponding collations were derived from several manuals from IBM concerning National Language Support, subject to the restrictions mentioned above. (This table represents the best information available at the time of writing. Due to continuing rapid geopolitical changes, the table may contain names for countries that no longer exist).

Country	Language	Primary Code Page	Primary Collation	Secondary Code Page	Secondary Collation
Argentina	Spanish	850	850ESP	437	437ESP
Australia	English	437	437LATIN1	850	850LATIN1
Austria	German	850	850LATIN1	437	437LATIN1
Belgium	Belgian Dutch	850	850LATIN1	437	437LATIN1
Belgium	Belgian French	850	850LATIN1	437	437LATIN1
Belarus	Belarussian	855	855CYR		
Brazil	Portuguese	850	850LATIN1	437	437LATIN1
Bulgaria	Bulgarian	855	855CYR	850	850CYR

Country	Language	Primary Code Page	Primary Collation	Secondary Code Page	Secondary Collation
Canada	Cdn French	850	850LATIN1	863	863LATIN1
Canada	English	437	437LATIN1	850	850LATIN1
Croatia	Croatian	852	852LATIN2	850	850LATIN2
Czech Republic	Czech	852	852LATIN2	850	850LATIN2
Denmark	Danish	850	850DAN		
Finland	Finnish	850	850SVE	437	437SVE
France	French	850	850LATIN1	437	437LATIN1
Germany	German	850	850LATIN1	437	437LATIN1
Greece	Greek	869	869ELL	850	850ELL
Hungary	Hungarian	852	852LATIN2	850	850LATIN2
Iceland	Icelandic	850	850ISL	861	861ISL
Ireland	English	850	850LATIN1	437	437LATIN1
Israel	Hebrew	862	862HEB	856	856HEB
Italy	Italian	850	850LATIN1	437	437LATIN1
Mexico	Spanish	850	850ESP	437	437ESP
Netherlands	Dutch	850	850LATIN1	437	437LATIN1
New Zealand	English	437	437LATIN1	850	850LATIN1
Norway	Norwegian	865	865NOR	850	850NOR
Peru	Spanish	850	850ESP	437	437ESP
Poland	Polish	852	852LATIN2	850	850LATIN2
Portugal	Portuguese	850	850LATIN1	860	860LATIN1
Romania	Romanian	852	852LATIN2	850	850LATIN2
Russia	Russian	866	866RUS	850	850RUS
S. Africa	Afrikaans	437	437LATIN1	850	850LATIN1
S. Africa	English	437	437LATIN1	850	850LATIN1
Slovak Republic	Slovakian	852	852LATIN2	850	850LATIN2
Slovenia	Slovenian	852	852LATIN2	850	850LATIN2
Spain	Spanish	850	850ESP	437	437ESP

Country	Language	Primary Code Page	Primary Collation	Secondary Code Page	Secondary Collation
Sweden	Swedish	850	850SVE	437	437SVE
Switzerland	French	850	850LATIN1	437	437LATIN1
Switzerland	German	850	850LATIN1	437	437LATIN1
Switzerland	Italian	850	850LATIN1	437	437LATIN1
Turkey	Turkish	857	857TRK	850	850TRK
UK	English	850	850LATIN1	437	437LATIN1
USA	English	437	437LATIN1	850	850LATIN1
Venezuela	Spanish	850	850ESP	437	437ESP
Yugoslavia	Macedonian	852	852LATIN2	850	850LATIN2
Yugoslavia	Serbian Cyrillic	855	855CYR	852	852CYR
Yugoslavia	Serbian Latin	852	852LATIN2	850	850LATIN2

When creating a new database

A user creating a new database should find the line with the country/language that they wish to use, then pick either the primary or secondary collation, depending on which code page is in use in their computer. (The **chcp** command will display the current code page number.) If their particular combination is not present, then another line with a satisfactory combination may be used, or a custom collation may be required.

International aspects of case sensitivity

Adaptive Server Anywhere is **case preserving** and **case insensitive** for identifiers, such as table names and column names. This means that the names are stored in the case in which they are created, but any access to the identifiers is done in a case-insensitive manner.

For example, the names of the system tables are held in upper case (SYSDOMAIN, SYSTABLE, and so on), but access is case insensitive, so that the two following statements are equivalent:

```
SELECT *
FROM systable
```

```
SELECT *
FROM SYSTABLE
```

The equivalence of upper and lower case characters is enforced in the collation. There are some collations where particular care may be needed when assuming case insensitivity of identifiers.

Example

In the Turkish 857TRK collation, the lower case **i** does not have the character **I** as its upper case equivalent. Therefore, despite the case insensitivity of identifiers, the following two statements are *not* equivalent in this collation:


```
SELECT *
FROM sysdomain

SELECT *
FROM SYSDOMAIN
```

Using multibyte collations

Adaptive Server Anywhere provides collations using several multibyte character sets, including the following:

Multibyte character set	Description
SJIS	Japanese Shift-JIS Encoding
EUC_JAPAN	Japanese EUC JIS X 0208-1990 and JIS X 0212-1990 Encoding
EUC_CHINA	Chinese GB 2312-80 Encoding
EUC_TAIWAN	Taiwanese Big 5 Encoding
EUC_KOREA	Korean KS C 5601-1992 Encoding
UTF8	UTF8 is a variable-byte encoding of 4-byte Unicode (UCS-4). UTF8 characters up to 4 bytes in length are supported, while UTF8 may use up to 6 bytes.

 For a complete listing, see "Supplied collations" on page 300.

This section describes how multibyte character sets are handled. The description applies to the supported collations and to any multibyte custom collations you may create.

Variable length character sets

Adaptive Server Anywhere supports variable length character sets. In these sets, some characters are represented by one byte, and some by more than one, to a maximum of four bytes. The value of the first byte in any character indicates the number of bytes used for that character, and also indicates whether the character is a space character, a digit, or an alphabetic (alpha) character.

Adaptive Server Anywhere does not support fixed-length multibyte character sets such as 2-byte Unicode or 4-byte Unicode.

Example

As an example, characters in the Shift-JIS character set are of either one or two bytes in length. If the hex value of the first byte is in the range 81-9F or E0-EF (decimal values 129-159 or 224-239); the character is a two-byte character and the subsequent byte (called a **follow byte**) completes the character. If the first byte is outside this range, the character is a single-byte character and the next byte is the first byte of the following character.

- ◆ The properties of any Shift-JIS character can be read from its first byte also. Characters with a first byte in the (hex) range 09 to 0D, or 20, are space characters.
- ◆ Characters in the ranges 41 to 5A, 61 to 7A, 81 to 9F or E0 to EF are alphabetic (letters).
- ◆ Characters in the range 30 to 39 are digits.

In building custom collations, you can specify which ranges of values for the first byte signify single- and double-byte (or more) characters, and which specify space, alpha, and digit characters. However, all first bytes of value less than 40 (hex 28) must be single-byte characters, and no follow bytes may have values less than 40. This restriction is satisfied by all known current encodings.

Creating databases with custom collations

You can create a database using a collation different from the supplied collations. This section describes how to build databases using such a **custom collation**.

Steps to create a database

❖ To create a database with a custom collation:

- 1 **Decide on a starting collation** You should choose a collation as close as possible to the one you want to create as a starting point for your custom collation.

☞ For a listing of supplied collations, see "Supplied collations" on page 300. Alternatively, run *dbinit* with the `-l` (lower case L) option:

```
dbinit -l
```

- 2 **Create a custom collation file** You do this using the Collation utility. The output is a collation file.

For example, the following statement extracts the collation from the default database into a file named *mycol.col*:

```
dbcollat -c " uid=dba;pwd=sql" mycol.col
```

When you use the Collation utility to extract a collation from an existing database, the database does not need to contain any information. If you do not currently have a database using the collation on which you want to base a custom collation, you can create such a database using the Initialization utility.

- 3 **Edit the custom collation file** Make the changes you wish in the custom collation file, and provide a name for your collation. You can do this using any text editor.

The name of the collation is specified on a line near the top of the file, starting with *Collation*. You should edit this line to provide a new name.

- 4 **Convert the file to a SQL script** You can do this using the *dbcollat* command-line utility using the `-d` switch. This step was not required in previous releases.

For example, the following command line creates the *mycol.sql* file from the *mycol.col* collation file:

```
dbcollat -d mycol.col custmap.sql mycol.sql
```

- 5 **Add the custom collation to the custom.sql script** The *custom.sql* script is stored in the *scripts* subdirectory of your installation directory. This step was not required in previous releases.
- 6 **Create the new database** You do this using the Initialization utility, specifying the name of the custom collation. In previous releases, you would have specified the collation file name.

For example, the following command line creates a database named *newcol.db* using the custom collation sequence **newcol**.

```
dbinit -z newcol temp.db
```

First-byte collation orderings

A sorting order for characters in a multibyte character set can be specified only for the first byte. Characters that have the same first byte are sorted according to the hexadecimal value of the following bytes. If the two characters are the same up to the length of the shorter of the two, the longer character is greater than the shorter.

Editing the collation file

This section describes the collation file format. Collation files may include the following elements:

- ◆ Comment lines, which are ignored by the database.
- ◆ A title line.
- ◆ A collation sequence section.
- ◆ An Encodings section (multibyte character sets only).
- ◆ A Properties section (multibyte character sets only).

Comment lines

In the collation file, spaces are generally ignored. Comment lines start with either `%` or `--`.

The title line

The first non-comment line must be of the form:

```
Collation label (name)
```

In this statement:

Item	Description
Collation	A required keyword.
label	The collation label, which appears in the system tables as SYS.SYSCOLLATION.collation_label and SYS.SYSINFO.default_collation. The label must contain no more than 10 characters, and must not be the same as one of the built-in collations. (In particular, do not leave the collation label unchanged.)
name	A descriptive term, used for documentation purposes. The name should contain no more than 128 characters

For example, the Shift-JIS collation file contains the following collation line, with label SJIS and name (Japanese Shift-JIS Encoding):

```
Collation SJIS (Japanese Shift-JIS Encoding)
```

The collation sequence section

After the title line, each non-comment line describes one position in the collation. The ordering of the lines determines the sort ordering used by the database, and determines the result of comparisons. Characters on lines appearing higher in the file (closer to the beginning) sort before characters that appear later.

The form of each line in the sequence is:

```
[sort-position] : character
```

or

```
[sort-position] : character [lowercase uppercase]
```

where:

Descriptions of arguments

Argument	Description
sort-position	Optional. Specifies the position at which the characters on that line will sort. Smaller numbers represent a lesser value, so will sort closer to the beginning of the sorted item. Typically, the sort-position is omitted, and the characters sort immediately following the characters from the previous sort position
character	The character whose sort-position is being specified
lowercase	Optional. Specifies the lowercase equivalent of the character. If not specified, the character has no lowercase equivalent
uppercase	Optional. Specifies the uppercase equivalent of the character. If not specified, the character has no uppercase equivalent

Multiple characters may appear on one line, separated by commas (.). In this case, these characters are sorted and compared as if they were the same character. You need to specify all three forms of the first character, then a comma, then all three forms of the second character, and so on.

Specifying character and sort-position

Each character and sort position is specified in one of the following ways:

Specification	Description
<code>\dnnn</code>	Decimal number, using digits 0-9 (such as <code>\d001</code>)
<code>\xhh</code>	Hexadecimal number, using digits 0-9 and letters a-f or A-F (such as <code>\xB4</code>)
<code>'c'</code>	Any character in place of c (such as ',')
<code>c</code>	Any character other than quote ('), back-slash (\), colon (:), or comma (.). These characters must use one of the previous forms.

The following are some sample lines for a collation:

```
% Sort some letters in alphabetical order
: A a A
: a a A
: B b B
: b b B
% Sort some E's from code page 850,
% including some accented extended characters:
: e e E, \x82 \x82 \x90, \x8A \x8A \xD4
: E e E, \x90 \x82 \x90, \xD4 \x8A \xD4
% Sort some special characters at the end:
: ' '
:
: \xF2
: \xEE
: \xF0
: -
: ', '
: ;
: ': '
: !
```

Other syntax notes

For databases using case-insensitive sorting and comparison (no `-c` specified on the `DBINIT` command line), the lower case and upper case mappings are used to find the lower case and upper case characters that will be sorted together.

For multibyte character sets, the first byte of a character is listed in the collation sequence, and all characters with the same first byte are sorted together, and ordered according to the value of the second byte. For example, the following is part of a Shift-JIS collation file:

```
: \xfb
: \xfc
: \xfd
```

In this collation, all characters with first byte `\xfc` come after all characters with first byte `\xfb` and before all characters with first byte `\xfd`. The two-byte character `\xfc \x01` would be ordered before the two-byte character `\xfc \x02`.

Any characters omitted from the collation will be added to the collation at the position equal to their binary value. DBINIT issues a message for each omitted character. It is recommended that any collation contain all 256 characters (first bytes).

The Encodings section

The Encodings section is optional, and follows the collation sequence. It is not useful for single-byte character sets.

The Encodings section lists those combinations of bytes which are valid characters. The format of the section may be described by example.

The Shift-JIS Encodings section is as follows:

```
Encodings:
[\x00-\x80, \xa0-\xdf, \xf0-\xff]
[\x81-\x9f, \xe0-\xef] [\x00-\xff]
```

The first line following the section title lists valid single-byte characters. The square brackets enclose a comma-separated list of ranges. Each range is listed as a hyphen-separated pair of values. In the Shift-JIS collation, values `\x00` to `\x80` are valid single-byte characters, but `\x81` is not a valid single-byte character.

The second line following the section title lists valid multibyte characters. Any combination of one byte from the second line followed by one byte from the first is a valid character. Therefore `\x81\x00` is a valid double-byte character, but `\xd0 \x00` is not.

The Properties section

The Properties section is optional, and follows the Encodings section. It is not useful for single-byte character sets.

If a Properties section is supplied, an Encodings section must be supplied also.

The Properties section lists values for the first-byte of each character that represent alphabetic characters, digits, or spaces.

The Shift-JIS Properties section is as follows:

```
Properties:  
space: [\x09-\x0d, \x20]  
digit: [\x30-\x39]  
alpha: [\x41-\x5a, \x61-\x7a, \x81-\x9f, \xe0-\xef]
```

This indicates that characters with first bytes \x09 to \x0d, as well as \x20, are to be treated as space characters, digits are found in the range \x30 to \x39 inclusive, and alphabetic characters in the four ranges \x41-\x5a, \x61-\x7a, \x81-\x9f, and \xe0-\xef.

Character set translation details

This section describes in more detail how Adaptive Server Anywhere carries out character set translation. Most users do not need this information. It is provided for advanced users, such as those who may be deploying applications or databases in a multi-character-set environment.

Locales

The database server and the client library recognize their language and character set environment using a **locale definition**. The application local is used by the client library when making requests, to determine which character set it wants results in. The server compares its own locale with the application locale to determine whether character set translation is needed. The locale consists of the following components:

- ◆ **Language** The language is a two-character string: FR for French and so on.
- ◆ **Character set** The character set is the code page in use. It uses a Sybase identifier.
- ◆ **Collation label** The collation label is the Adaptive Server Anywhere collation.

Each of these is discussed in the following sections.

Determining the locale language

The language is a two-letter identifier; EN specifies English, JA specifies Japanese, and so on. It is used to determine the following behavior:

- ◆ Which language library to load by the database server and by the client library.
- ◆ For the client, which language to request from the database.

The way the language component of the locale is determined depends on the operating system:

- 1 If a SQLLOCALE environment variable exists, it is used.
 - ☞ For more information, see "The SQLLOCALE environment variable" on page 316.
- 2 On Windows and Windows NT, get the language from the registry, as described in "Registry settings on installation" on page 10 of the book *Adaptive Server Anywhere Reference Manual*.
- 3 On other operating systems, get the language from the operating system.

Determining the locale character set

Both application and server locale definitions have a character set. The application uses its character set when requesting character strings from the server. The database server compares its character set with that of the application to determine whether character set translation is needed.

The character set is also used to determine which code page to use when initializing a database, if none was specified by the application and no collation label is available.

The character set is determined as follows:

- 1 If a `SQLLOCAL` environment variable exists, it is used.
↪ For more information, see "The `SQLLOCAL` environment variable" on page 316.
- 2 Get the character set from the operating system:
 - ◆ On Windows NT, use the `GetACP` system call. This returns the ANSI character set.
 - ◆ On UNIX, default to ISO8859-1.
 - ◆ On other platforms, use code page 850.

Determining the locale collation label

Each database has its own collation. The collation label taken from the locale definition is used by the database server to determine which code page to use when initializing a database.

The collation label is determined as follows:

- 1 If a `SQLLOCAL` environment variable exists, it is used.
↪ For more information, see "The `SQLLOCAL` environment variable" on page 316.
- 2 Get the language and character set from the operating system as described in the previous two sections, and use an internal table to find a collation label corresponding to that language and character set.

The `SQLLOCAL` environment variable

The `SQLLOCAL` environment variable is a single string that consists of three semi-colon-separated assignments. It has the following form:

Charset=*cslabel*;Language=*langlabel*;CollationLabel=*colabel*

The values that the labels can take are set out in the following tables.

Character set label values

The following table shows the valid character set label values, together with the equivalent IANA labels and a description:

Character set label	IANA label	Description
iso_1	iso_8859-1:1987	ISO 8859-1 Latin-1
cp850	<N/A>	IBM CP850 - European code set
cp437	<N/A>	IBM CP437 - U.S. code set
roman8	hp-rpman8	HP Roman-8
mac	macintosh	Standard Mac coding
sjis	shift_jis	Shift JIS (no extensions)
euclj	euclj	Sun EUC JIS encoding
deckanji	<N/A>	DEC Unix JIS encoding
eucln	<N/A>	EUC CNS encoding: Traditional Chinese with extensions
euclb	<N/A>	EUC GB encoding = Simplified Chinese
cp932	windows-31j	Microsoft CP932 = Win31J-DBCS
iso88592	iso_8859-2:1987	ISO 8859-2 Latin-2 Eastern Europe
iso88595	iso_8859-5:1988	ISO 8859-5 Latin/Cyrillic
iso88596	iso_8859-6:1987	ISO 8859-6 Latin/Arabic
iso88597	iso_8859-7:1987	ISO 8859-7 Latin/Greek
iso88598	iso_8859-8:1988	ISO 8859-8 Latin/Hebrew
iso88599	iso_8859-9:1989	ISO 8859-9 Latin-5 Turkish
iso15	<N/A>	ISO 8859-15 Latin1 with Euro, etc.
mac_cyr	<N/A>	Macintosh Cyrillic
mac_ee	<N/A>	Macintosh Eastern European
macgrk2	<N/A>	Macintosh Greek
macturk	<N/A>	Macintosh Turkish
greek8	<N/A>	HP Greek-8
turkish8	<N/A>	HP Turkish-8
koi8	<N/A>	KOI-8 Cyrillic

Character set label	IANA label	Description
tis620	<N/A>	TIS-620 Thai standard
big5	<N/A>	Traditional Chinese (cf. CP950)
eucksc	<N/A>	EUC KSC Korean encoding (cf. CP949)
cp852	<N/A>	PC Eastern Europe
cp855	<N/A>	IBM PC Cyrillic
cp856	<N/A>	Alternate Hebrew
cp857	<N/A>	IBM PC Turkish
cp860	<N/A>	PC Portuguese
cp861	<N/A>	PC Icelandic
cp862	<N/A>	PC Hebrew
cp863	<N/A>	IBM PC Canadian French code page
cp864	<N/A>	PC Arabic
cp865	<N/A>	PC Nordic
cp866	<N/A>	PC Russian
cp869	<N/A>	IBM PC Greek
cp874	<N/A>	Microsoft Thai SB code page
cp950	<N/A>	PC (MS) Traditional Chinese
cp1250	<N/A>	MS Windows 3.1 Eastern European
cp1251	<N/A>	MS Windows 3.1 Cyrillic
cp1252	<N/A>	MS Windows 3.1 US (ANSI)
cp1253	<N/A>	MS Windows 3.1 Greek
cp1254	<N/A>	MS Windows 3.1 Turkish
cp1255	<N/A>	MS Windows Hebrew
cp1256	<N/A>	MS Windows Arabic
cp1257	<N/A>	MS Windows Baltic
cp1258	<N/A>	MS Windows Vietnamese
utf8	utf-8	UTF-8 treated as a character set

Language label values

The following table shows the valid language label values, together with the equivalent ISO 639 labels:

Language label	Alternate label	ISP_639
us_english	english	EN
french	<N/A>	FR
german	<N/A>	DE
spanish	<N/A>	ES
japanese	<N/A>	JA
korean	<N/A>	KO
portugese	portugue	PT
chinese	<N/A>	ZH

Collation label values

The collation label is a label for one of the supplied Adaptive Server Anywhere collations, as listed in "Supplied collations" on page 300.

