Major New Features in TRIP 6

TRIPsystem
Product Documentation

End User License Agreement

All rights to this software, its documentation and logotypes of the TRIP product family and software (altogether “Software”) supplied by Smaser AG (Smaser) are exclusively owned by Smaser.

The transfer of this Software, solutions or parts thereof requires the prior written agreement of Smaser. Furthermore, the customer has the right to use licensed Software and / or process solutions supplied by Smaser to the extent specified in his contract with Smaser.

The free-to-use non-commercial version doesn’t require a prior written agreement with Smaser but such customers, organizations and/or third parties agree by using the software and / or solution of Smaser to be strongly obliged to keep all rights to this software, documentation and logotypes of the TRIP product family absolutely uninfringed and protected.

Table of Contents

Introduction

This document describes briefly the major new features of TRIP version 6. More information can be found in the specific documentation.

- Unicode support
- LDAP support
- Performance issues for indexing
- Enhanced fuzzy search
- Various kernel enhancements
- TRIPmgr enhancements
- XML based TRIPsdk products
- TRIPsql enhancements
- TRIPview enhancements

Unicode - overview

Background

Historically, there have been two independent attempts to create a single unified character set. One was the ISO 10646 project of the International Organization for Standardization (ISO), the other was the Unicode Project organized by a consortium of (initially mostly US) manufacturers of multi-lingual software. Fortunately, the participants of both projects realized in around 1991 that two different unified character sets is not what the world needs. They joined their efforts and worked together on creating a single code table.

Encodings

UCS and Unicode are first of all code tables that assign integer numbers to all characters. There exist several alternatives for how a sequence of such characters or their respective integer values can be represented as a sequence of bytes. The two most obvious encodings store Unicode text as sequences of either 2 or 4 bytes sequences. The official terms for these encodings are UCS-2 and UCS-4 respectively. Unicode supports three different encodings, UTF-8, UTF-16 and UTF-32.

UTF-8 is most common and is also the choice for TRIP

- Introduced to provide an ASCII backwards compatible multi-byte encoding.
- UTF-8 is made up of eight-bit code units, each of which is one byte.
- UTF-8 is a variable-length encoding, using 1 to 4 bytes.

Short Summary of Unicode

Unicode defines a 21-bit range of code points (U+00 to U+10FFFF). There are several encodings of the character values in Unicode. UTF-8 encodes the full range in 1 to 4 byte values. In multi-byte

sequences, the number of leading 1-bits in the first byte is identical to the number of bytes in the entire sequence.

The first 128 characters of UCS or Unicode are the same as in US-ASCII and the first 256 characters are the same as in ISO-8859-1 (LATIN-1).

Unicode support in TRIP

In order to implement Unicode support, TRIP has been integrated with ICU (International Components for Unicode), a software originating from IBM. ICU contains all the necessary definitions of character data, functions and methods to convert, normalize, compare, search and sort Unicode strings. The parts of ICU that TRIP is using handle data conversion and translation (normalization).

The Unicode enabled TRIP supports:

- Unicode data loaded via a TFORM file
- Unicode data entered via the TRIP api
- Commands in Unicode entered via the TRIP api
- Unicode data output via the TRIP api
- Unicode data output via the PRINT command
- Data stored in Unicode format in databases
- Global Update with Unicode data

TRIP handles a mixture of Unicode and non-Unicode databases, as long as there is only one type of encoding for the non-Unicode databases.

- A mix of Unicode and Latin-1 databases is ok
- A mix of Unicode and Latin-2 databases is ok
- A mix of Unicode, Latin-1 and Latin-2 databases is not accepted

TFORM files and export formats should contain a marker to indicate what encoding is used. TRIP will automatically write this marker into the file when exporting data. If the marker is absent, the value of TDBS_CHARS will be used. This optional marker has always been in use with TRIP but with the introduction of Unicode, the possible settings have got yet another member, UTF8.

A TRIP application can define at start-up what encoding to use, default setting is the value of TDBS_CHARS, which can not take a value indicating Unicode. The only way to run a Unicode enabled TRIP application is via methods defined in TRIPjxp/nxp/jtk/com/api.

Care should be taken when starting a Trip application in Latin-n mode and accessing Unicode databases.

- Reading data might get unwanted result as Trip will translate Unicode data to Latin-n. Any Unicode character without a Latin-n value will be returned as a fill character.
- Text modifications might thus cause data corruption.

Unicode enabling of databases can be done with TRIPmgr only. TRIPclassic stays Latin-n and GBK enabled but can access Unicode databases with the automatic Latin-n conversion.

Using Unicode with TRIP is optional and the current way of handling text data in TRIP will also be kept.

LDAP - overview

The Lightweight Directory Access Protocol, LDAP, is an application protocol for querying and modifying directory services running over TCP/IP. A directory is a set of objects with similar attributes organized in a logical and hierarchical manner. The most common example is the telephone directory, which consists of a series of names (either of persons or organizations) organized alphabetically, with each name having an address and phone number attached. Due to this basic design (among other factors) LDAP is often used by other services for authentication.

An LDAP directory tree often reflects various political, geographic, and/or organizational boundaries, depending on the model chosen. LDAP deployments today tend to use Domain name system (DNS) names for structuring the topmost levels of the hierarchy. Deeper inside the directory might appear entries representing people, organizational units, printers, documents, groups of people or anything else which represents a given tree entry (or multiple entries).

LDAP support in TRIP

Basically there are three systems of authenticating users in a system.

- A standalone login framework, e.g. the Trip default login. For a large number of sites the built-in Trip authentication system is great as it allows control over users in an independent manner.
- A Single Sign In (SSI) interface, e.g. LDAP. For sites with a large user community, the ability to use the same credentials to validate logins to various applications is essential.
- A Single Sign On (SSO) interface, e.g. LDAP and Kerberos. Once a user logs in, his credentials are used to validate him during that entire session. With one sign on he has validated himself.

The LDAP support in TRIP is providing SSI and is currently available for Windows, Linux and Solaris. Using LDAP removes the need for user's passwords to be maintained in TRIP as login requests are processed via the LDAP protocol.

LDAP support is configured using new environment variables residing in the [Privileged Section] of the TRIPrcs file. The new TRIPrcs file created by a TRIP6 installation contains documentation on how to setup TRIP to use an LDAP server.

By default, if a user provides a valid set of credentials for an LDAP authentication provider, but the user is unknown to TRIP, the user will be logged into TRIP as a guest user (under the BUILTIN_GUEST account). To disable this functionality set the following variable:

TDBS_DISALLOW_GUEST=True

To establish LDAP as the authentication provider, set the following variable:

TDBS_AUTH_PROVIDER=LDAP

The default behavior of the system in the absence of such a setting is to fallback to using CONTROL for all authentication requests.

The LDAP provider needs to know which servers are capable of authenticating. The following variable is a comma-separated list of servers and optional port numbers. For example:

TDBS_LDAP_SERVER=server1, server2:3030, server3

In the absence of port numbers, the default port for LDAP (or LDAP over SSL) will be provided by the system.

Communication with the LDAP server(s) can take place in two different ways, either insecure (the SIMPLE mechanism) or via an encrypted transmission (the SSL mechanism). Set the following variable accordingly:

TDBS_LDAP_MECHANISM={SIMPLE | SSL}

If using SSL for communication, the location of the local certificate database must be provided by setting the following variable. As TRIP uses the Mozilla LDAP SDK, the database in question is that used by the Mozilla and Firefox browser applications (amongst others), is entitled "cert8.db" and can normally be found within a user profile, for example:

TDBS_LDAP_SSL_CERT_DB=/home/tarzan/.mozilla/cert8.db

On Windows, the certificate database can be found inside a user's Application Data folder under Documents and Settings, like:

TDBS_LDAP_SSL_CERT_DB=C:\Documents and Settings\bjensen\Application Data\Mozilla\Firefox\...\cert8.db

All connections to the LDAP servers are synchronous and subject to timeouts to avoid an operation hanging the client. Set this variable to define a timeout in milliseconds that TRIP should wait for a response from the LDAP server(s).. For example:

TDBS_LDAP_TIMEOUT=1500

In order to find users, TRIP needs to be able to browse the LDAP repository. If the repository supports anonymous access for browsing, set the following variable to True, otherwise set it False.

TDBS_LDAP_ANONYMOUS={True | False}

If anonymous browse access is not supported, you must provide the DN and credentials (password) for the user that will be used to perform browse operations when searching for users to authenticate. This is done using the following variables:

TDBS_LDAP_USERNAME	is the fully qualified DN of the browse user
TDBS_LDAP_PASSWORD	is the plain text of the browse user's password

The user specified must have read access to the entire tree descending from the root node provided by TDBS_LDAP_BASE (described below).

TDBS_LDAP_USERNAME=cn=Manager,dc=bjensen,dc=com

TDBS_LDAP_PASSWORD=thx1139

When attempting to authenticate a user, that user's identity will typically be provided as an RDN rather than a fully specified DN. In order to turn that RDN into a DN for authentication, the following set of variables must be provided:

TDBS_LDAP_BASE	defines the base of the tree in which users can be found
TDBS_LDAP_SEARCH	defines an LDAP search string for finding users

For example, if the TRIP user community is collected in a subtree of the LDAP repository with a logical base of ou=tox/o=pharma/c=us, then the base of the search tree should be established as follows:

TDBS_LDAP_BASE=ou=tox,o=pharma,c=us

To find a user by RDN (for example by the UID or CN that the user presents as their typical login public key), specify an LDAP search string using the %u% substitution string to stand for the user's provided RDN. For example, when using the person structural schema (or some derivation, for example organizationalPerson, or inetOrgPerson) with the uidObject addon schema the search string would be:

TDBS_LDAP_SEARCH=(&(objectclass=person)(uid=%u%))

Any occurrence of the "%u%" pattern within the string will be replaced with whatever "username" is provided to TRIP during the login process.

Once the user has been found (i.e. their RDN has been dereferenced to a DN) their record must be turned into a TRIP username for use within the CONTROL database. The following variable is used to specify the field from the user record that will provide this mapping, for example the "uid" field in the case of most user-related schemas:

TDBS_LDAP_MATCH=uid

One specific username is never subject to the directory service provider model: SYSTEM. This is always validated against CONTROL. The Trip user BUILTIN_GUEST is added to CONTROL via the Trip6 installation. This user can be used by DBAs to provide “default” access to an installation in the absence of a valid Control record when provided with valid LDAP login credentials. This is a normal Trip user account and can be established with whatever level of user access is required by the DBA.

Performance issues for indexing

The index program will optionally run a re-index if that is deemed faster than an update. To enable this, set the value TDBS_FORCE_REINDEX=Y in the TRIPrcs file. This feature will be useful for situations like the following example:

- A database containing some 1000 records is indexed
- Another 100000 (or more) records are added and indexed

Memory usage of the index program can be limited by setting the value TDBS_MAX_ALLO_MEM, expressed in MB, in the TRIPrcs file. The program always checks the success of any memory allocation but setting this value ensures that an index run is not allocating more than up to a certain limit. The default value for this is 256 MB.

Other new features of the new index program:

- A better detection of modified parts of records decreases the disk I/O.
- Minimized use of temporary files, the TPO file no longer used.
- Various methods to minimize memory usage when indexing
  - Base/offsets and ”runs” for record numbers
  - E.g. Instead of storing record numbers 123456,123457,123470,123475 the sequence can be stored as Base:123456, RunValues: 0,1,13,5
  - Flags for same/close values
  - E.g. Subfield values are often 1 or the same as the previous. Instead of storing each value a flag is raised to indicate the situation
- Improved clean-up of old records after index
  - All records being modified are identified in a bit mask
  - Only marked records are read for clean-up

Non-Boolean searching

Improved Non-Boolean rank calculation:

- Weighting co-occurrence of terms, i.e. if > 1 term from the query is present in the document, this gives higher weight in order to bias multi-term hits.
- Definable search precision percentage (Define about=n)
  - Based on highest ranked record

- Define About=n, means that records with less relevance are discarded.
- Records with a weight of n% of the highest rank will be kept.

Stemming

Stemming is now based on a language setting in the database design. This value can be set with TRIPmgr only. This feature makes it possible to open a cluster of databases with different languages and get non-Boolean search with appropriate stemming.

Databases without a language setting will be using the default value defined in the TRIPrcs file. Stemming is currently only available with Non-Boolean search using the About() function.

Term list enhancements

Term lists can now include field content from up to 10 fields including the actual field being searched. The additional field names can be entered together with the Display command or by a previous Define command.

- Display f1=abc# field=f2,f3,…
- Define display field=f2,f3,…

Class names assigned by a categorization of records with TRIP can now be viewed using the Display command. To display all class names associated with the currently open database(s) and show the record count for each one, do like this:

- Display class(#)

The performance for a search set restricted term lists with large databases and small restriction sets is much improved. A Display command like this will execute almost instantly even with a very large database if the restriction set is reasonably small.

- Display s=n f1=#

Thesaurus lists will now be shown without any duplicates. The example below shows the difference between the result in TRIP5 and earlier and TRIP6.

- Trip5
  - All terms from a Display down() list will show up with its tree structure, thereby duplicating information for terms that are NTs of other terms.
  - E.g. Display down(#) in the demo thesaurus Thesali results in a forest comprising 139 root terms, all together 461 terms, many of which are duplicated.
- Trip6
  - All term from a Display down() list will show up only once, in its proper place within the thesaurus forest.
  - E.g. Display down(#) in the demo thesaurus Thesali results in a forest comprising 4 root terms, all together 139 terms, none of which are duplicated.

Find command enhancements

Mapped search can now be done with numeric fields. The following example finds all records which have the same value in the numeric field num1 as those in the numeric field num2 in records from search set n.

Find num1=n.num2

Searches can now be made using the in-equal operator. This following example finds all records not containing the string abc.

Find fld <> abc

Enhanced <link> output format filter

Link databases are no longer required to have record name fields. It is now possible to specify the field name or field type and how to search, exact or non-exact.

- TEXT, PHRASE or TEXT+PHRASE
- Exact match in PHRASE fields
- A specific PHRASE field
- Exact match in a specific PHRASE field

The field to be retrieved from a link database can be of type TEXT.

- Needs a dummy TEXT field in the database to store the data
- Use standard formatting features to show the contents of the dummy field

Retrieving data from several record hits.

- Needs a dummy field to store the data
- If the source field is TEXT or has several subfields, the target dummy field must be a part field of the same type.
- If the target dummy field is not a part field, the data from each record found is stored in each subfield
- If the target dummy field is a part field, the data from each record found is stored in each part
- Use standard formatting features to show the contents of the dummy field, either looping over subfields or both parts and subfields

Fuzzy search

Fuzzy search, performed by the Fuzz() function is a way to find misspelled terms. Note that this is not a phonetic search method like Soundex, Metaphone and others, rather a way of matching a search term with a selection of candidates, based on textual similarity.

The new FUZZ() function is implemented with a choice of two different algorithms where each one has some pros and some cons.

- The S-gram method. Based on an idea from University of Tampere, Finland, where not only traditional n-grams are used but also n-grams in which characters are skipped.
  - TRIP uses 2-grams and skips one character for the s-grams.
    - E.g. 2-grams of the term ”wonderland” are wo,on,nd,de,er,rl,la,an,nd
    - E.g. S-grams of the term ”wonderland” are wn,od,ne,dr,el,ra,ln,ad
  - Selection of terms based on the positions of these 2/S-gram within the terms

- Very high frequency terms are removed if relevant.

- Sliding mask
  - E.g. the term “wonderland” generates these masked searches :!onderland, w::!nderland, wo::!derland, won::!erland,…
  - Locates only one spelling mistake

Similarity measure used for ranking term candidates found by 2/s-grams is the Levenshtein distance. The measure is named after the Russian scientist Vladimir Levenshtein, who devised the algorithm in 1965.

The Levenshtein distance (or edit distance) is a measure of the similarity between two strings. The distance is the number of deletions, insertions, or substitutions required to transform one string into the other. The greater the Levenshtein distance, the more different the strings are.

For example, the Levenshtein distance between "kitten" and "sitting" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits

- kitten → sitten (substitution of 's' for 'k')
- sitten → sittin (substitution of 'i' for 'e')
- sittin → sitting (insert ‘g' at the end)

The new Fuzz() function takes four optional values to tune the processing:

- Def fuzz=a,b,c,d
  - a=precision percentage (1-100), default=75
  - b=number of terms to include when used with Find, default=2
  - c=number of top ranked terms to be searched for “best” match term, default=5
  - d=algorithm: 1=grams, 2=SlidingMask, default=1

Spelling corrections

The new Fuzz() function can be used for simple spell-checking like this:

- Use the Display command with fuzz() to show a term list and choose among the suggestions
- This is not really a spell-checker, rather a spell-guesser as it’s not using any dictionary or lexical analysis, just displaying similar terms

If the Fuzz() options has been set quite restrictive using “Define fuzz”, it can also be used to search for possible misspelled words without too much noise.

Crash handling

Whenever a server crash happens, TRIP6 will write out the stack trace plus any additional information provided by the operating system into a log file. Stack traces are saved in the TDBS_LOG directory in files named backtrace_nnn.log, where nnn is the process identification number. As the log file is stored in the process working directory if TDBS_LOG is defined, it is a good practice always to define this value.

The crashing session is gently terminated, returning a message about a crash to the application.

Chinese word segmentation enhancements

TRIP6 supports three different methods for Chinese Word Segmentation.

- M – Maximum
  - This method was introduced with TRIP5 and is now enhanced to handle both cross-ambiguities and continuous-cross-ambiguities correctly.
- W – Word
  - This method is introduced with TRIP6 and works as method M and performs a re-segmentation of all words longer than three Chinese characters.
- A – All
  - This method was introduced with TRIP5 and is the same for TRIP6.

TRIPmgr

Detailed information about TRIPmgr is found in the help file TRIPmmc.chm found in the TRIPmgr installation directory.

- Unicode support – choice of database character set
- Language setting for stemming
- Grid configuration
- List of licensed products for a server
- Improved Thesaurus listing
- Compress BAF files (same as the server utility Packit)
- Check BAF files status (same as the server utility Doctor)
- Show database modification/index time and number of records
- Load/Index databases and get a notification when ready
- Multi-database settings - Select a group of databases, right-click and choose properties
- Chinese character handling on non-Chinese Windows/XP. Requires TDBS_CHARS=GBK in the TRIPrcs file

TRIPsdk

TRIPxpi

Three new SDK products, formerly known by the name of the underlying network protocol, TRIPxpi have been added to TRIPsdk.

- TRIPnxp for .NET development
- TRIPjxp for Java development
- TRIPaxp for Javascript-based development in AJAX-style

These products are all XML based and defines operations that normally would require a large number of networked function calls using the old TRIPsdk products TRIPclient/jtk. The operations defined by XPI are exposed via TRIPjxp and its sibling products through their classes and methods and can be tunneled through other protocols, such as HTTP or TRIPnet.

All of these products can also be set up using grid technology.

- Execute searches on more than one TRIP server with the same query.
- Uses a servlet-based ”grid router” software to route requests and responses.
- Configured with TRIPmanager.
- Read-only access.

For a full description of the new TRIPsdk products, please read the separate documentation that comes with each product.

TRIPclient, TRIPjtk

These two products have been upgraded to support Unicode and the new TRIPview (described below).

TRIPview

New features include:

- A new out-of-process server design to isolate the TRIP server process from possible extraction and conversion malfunctions.
- Extraction of document properties
- HTML conversion with highlighting.
- Multipart MIME (.mht) as alternative output format for HTML.

TRIPsql

TRIPsql exposes TRIP in the guise of a relational database and supports a subset of the SQL92 grammar via server-side SQL engine, fronted by an ODBC driver and a JDBC driver.

New features include:

- Support for sub-select statements
  - Can often be replaced by joins, but are a so essential part of the SQL syntax, that we feel that this must be supported.
- Multipart MIME (.mht) as alternative output format for HTML.

Supported server platforms

- 32bit processors
  - Windows (X86) 2000,2003,XP,Vista
  - Linux (X86) kernel 2.4 and later
  - Solaris (SPARC) version 2.8 and later
  - HP-UX (PA-RISC) version 11.0 and later
  - IBM-Aix (Power) version 5.1 and later

- 64bit processors
  - HP-Tru64 (Alpha) version 4.0 and later
  - Linux (X86-64)
  - Solaris (X86-64)
  - Windows (X86-64) server 2003, 2008, XP, Vista
  - zLinux IBM Mainframe s390 (available on request)