Unicode Support - Migration Procedure

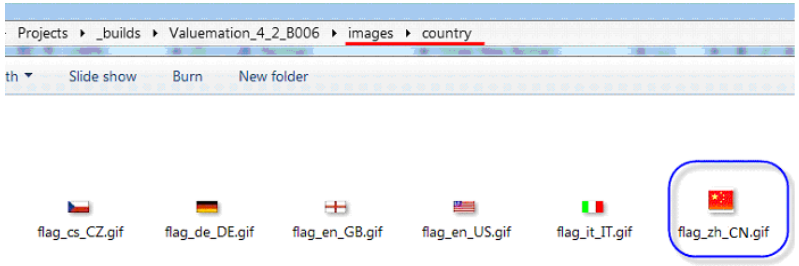
This document describes the steps that are necessary to take in order to make it possible to use languages with special characters in Valuemation (e.g. Chinese, Arabic etc.). The currently supported languages use Latin characters (e.g. German, English, Italian) and even those characters differ slightly language from language (e.g. umlauts in German language - e.g. ö) which require the use of a specific code page and collation in the database which makes it difficult to work with strings containing Latin characters that are part of another code page. The Unicode deals with the problem.

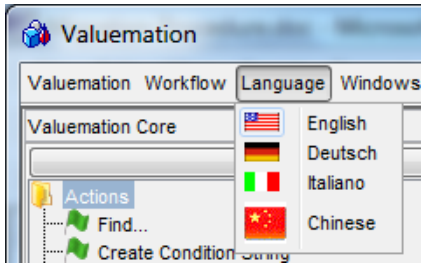
In the document we will show how to add the Chinese language but adding any other language would be very similar.

Adding a new language

To add a new language to Valuemation, it is necessary to create a new main parameter and add a new icon with the country flag.

- The path of the parameter is *languageDefinition* and the value is a combination of a Java locale (language code and country code) and name of the language which will appear in the Language menu, e.g. zh_CN_Chinese. Note that it is very important to use the correct Java locale because it influences also the format of numbers and dates/times. The **new_language.sql** contains an insert statement which inserts this new main parameter.
- The country flag needs to be an icon in the GIF format, approximately 20x11 pixels in size and the name has to be flag_<locale>.gif - e.g. **flag_zh_CN.gif**. The icon than needs to be placed in the *images/country* directory.





Creating translations for the new language

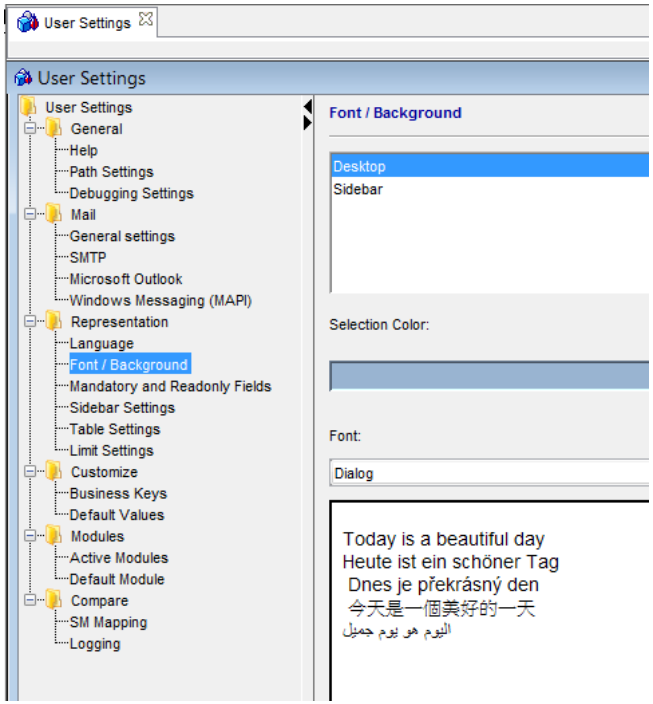
The first step is to populate the 3 translation tables (AMT_TRANSLATION, AMT_BOTRANSLATION, AMT_SIDEBARTRANS) with data copied from the English language – all you need to do is to execute the **new_language.sql** SQL script.

Translations for the new language must be created manually. What needs to be translated:

- names of object types and attributes (do this in the Translation Editor)
- names of valueset items (this can be done in Valueset Customizer for each valueset but it will be easier to do this in the normal editor from the catalog Translations of Application Text – just filter the catalog by Base Name = valueset/ValueSetmap)
- some translations of application text (do it in the catalog Translations of Application Text), esp.
  - messages (info, error, warning) presented to the end user (messages/Messages)
  - names of actions (including . tooltips) – (actions/Actionnames)
  - ....
- translations of sidebar catalogs (do this in the Catalog Customizer on the Translations tab for each catalog)

Fonts

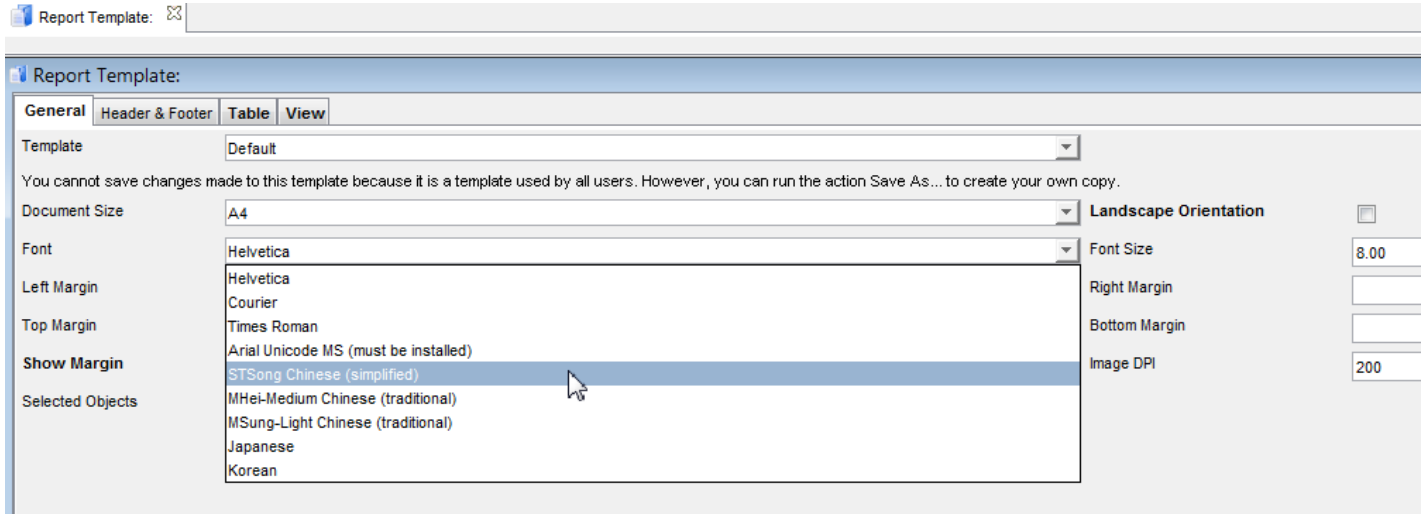
In order to display languages like Chinese in Valuemation rich client correctly, it is necessary to use one of the 5 standard Java fonts with full Unicode support – that is *Dialog*, *DialogInput* , *Monospaced*, *Serif*, *Sans-Serif*. Until VM 4.2, the default font was *Arial*, which doesn't contain Chinese characters. In VM 4.2, the default font has been changed to *Dialog*. However, the font setting is user-dependent (it is in the User Settings) so it is very likely that some Valuemation users already use an incorrect font (it is stored as a main parameter). Therefore, it is recommended to delete the user main parameters for fonts so that all users use the default font *Dialog*. The **new_language.sql** contains a DELETE statement which deletes those main parameters. Of course, each user can again select the font that they like (in User Settings/Representation/Font-Background) but if they decide to use a font different than the 5 mentioned fonts, they should make sure the font can display the language that they use – this can be tested easily by writing some sample text in the text area below the font selector. Note that since VM 4.2, there is only one font selector (not two) which makes it easier to change the font and keep the font consistent throughout the GUI.



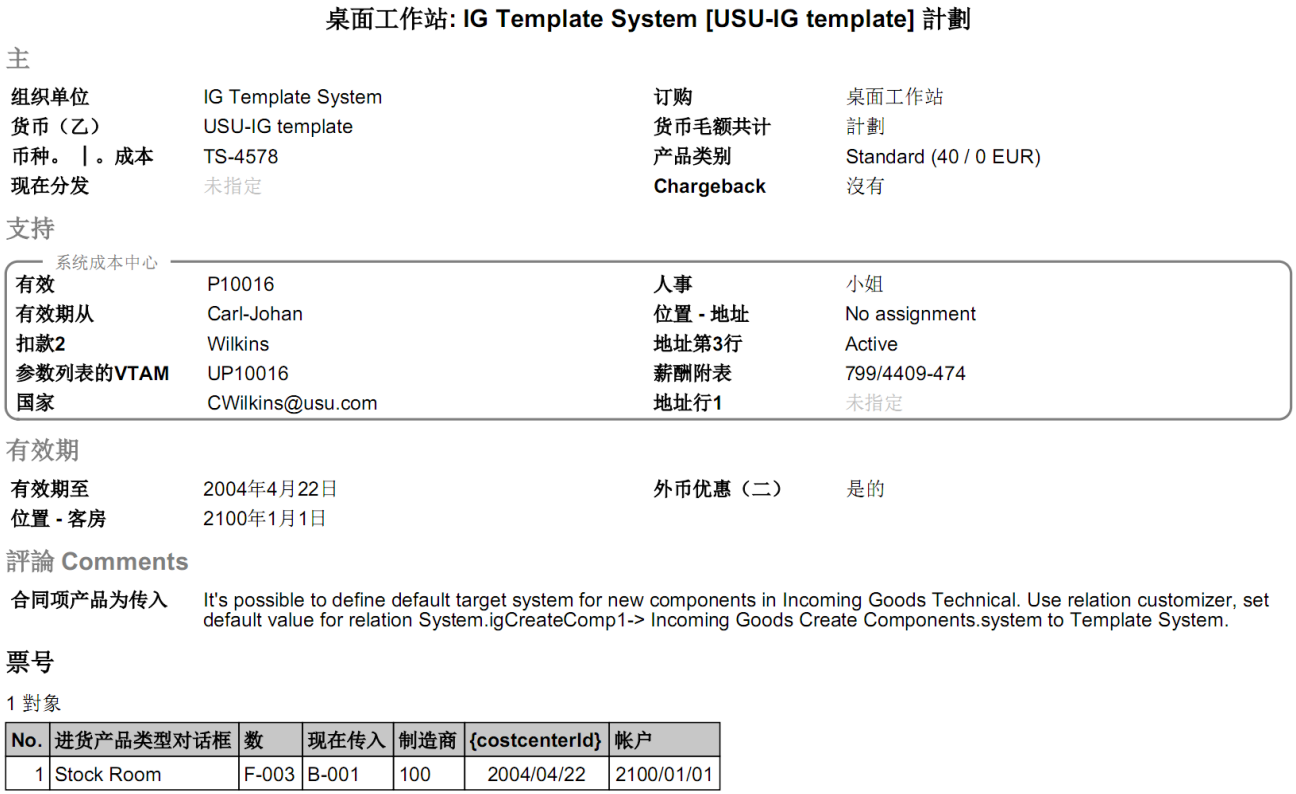
Fonts in Reports

PDF reports

If you want to generate PDF reports containing Chinese characters (when the language is switched to Chinese or the data contains Chinese strings), you need to change the font in the Report Template:



PDF documents use their own fonts (independent of the font settings in User Settings). There are 3 built-in fonts (*Helvetica*, *Courier*, *Times-Roman*) that each PDF reader has to be able to render no matter which OS is used and no matter which fonts are installed in the OS. However, these fonts contain only Latin characters, but not Chinese for example. Therefore, you need to choose one of the special Asian fonts (depending on the language), e.g. *STSong* for simplified Chinese. When the PDF is opened in a PDF viewer, the viewer detects that an Asian font is used and requests a download of the font (only the very first time). After confirmation, the PDF is shown:



桌面工作站: **IG Template System [USU-IG template]** 計劃

主

组织单位	IG Template System	**订购**	桌面工作站
货币（乙）	USU-IG template	**货币毛额共计**	計劃
币种。 | 。成本	TS-4578	**产品类别**	Standard (40 / 0 EUR)
现在分发	未指定	**Chargeback**	沒有

支持

系统成本中心

有效	P10016	**人事**	小姐
有效期从	Carl-Johan	**位置 - 地址**	No assignment
扣款2	Wilkins	**地址第3行**	Active
参数列表的VTAM	UP10016	**薪酬附表**	799/4409-474
国家	CWilkins@usu.com	**地址行1**	未指定

有效期

有效期至	2004年4月22日	**外币优惠（二）**	是的
位置 - 客房	2100年1月1日		

評論 Comments

合同项产品为传入	It's possible to define default target system for new components in Incoming Goods Technical. Use relation customizer, set default value for relation System.igCreateComp1-> Incoming Goods Create Components.system to Template System.

票号

1 對象

No.	进货产品类型对话框	数	现在传入	制造商	{costcenterId}	帐户
1	Stock Room	F-003	B-001	100	2004/04/22	2100/01/01

Another alternative is the use of the *Arial Unicode MS* font (from Microsoft), which contains ALL characters (of all languages including Asian languages).
Since the font can display text in mixed languages (German+Czech+Arabic+Japanese for instance) it is ideal for multi-language environments. Notice the field showing Hallo world in several languages:

評論 Comments

合同项产品为传入

Hallo world
Ahoj světe
喂世界
餵世界
مرحبا العالم
привет мир

— —

However, the font is not free (it comes with MS Office if you also install additional language support, otherwise it is $100 for workstation use; the license for enterprises, web developers, for hardware & software redistribution or server installations needs to be enquired here. Therefore, the font is

not distributed with VM so it's up to the customer to get the font and install it if he wants to use it. Another (minor) disadvantage is that Asian characters don't look as nice in this font as in the special Asian fonts (the width of the strokes is constant).

In fact, it also possible to use any other font - you just have to add a new value to the *ReportTemplate.FontName* valueset with the file name of the font and the encoding - e.g. Arial.ttf/cp1250.

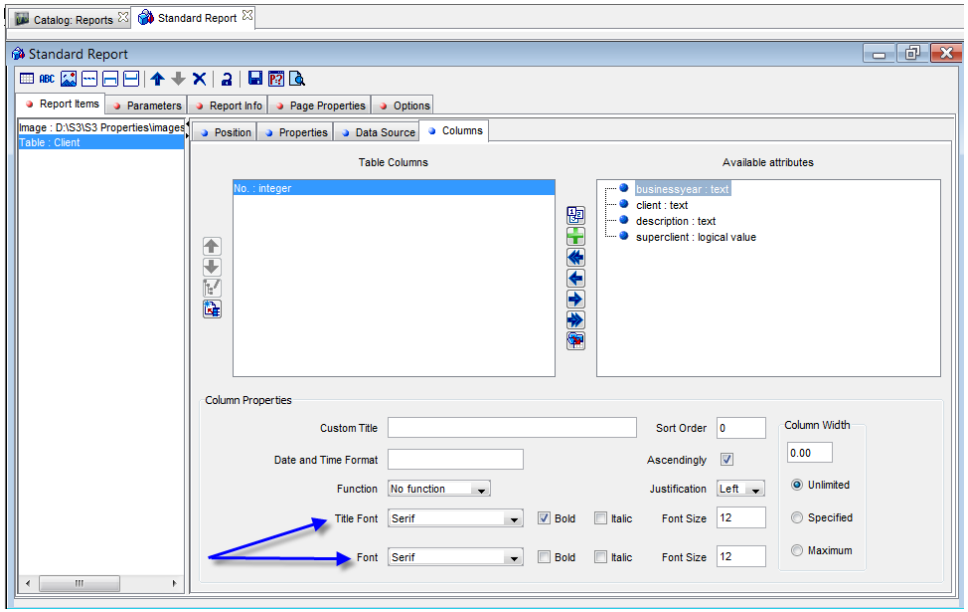
If you want to use the Arial Unicode MS font or any other TrueType font, you need to copy it to the root of Valuemation directory OR, leave it where it is (e.g. in C:\Windows\Fonts) but set the system property *fontdirectory* pointing to this directory,
e.g. –Dfontdirectory= C:\Windows\Fonts.

Old (deprecated) internal reports

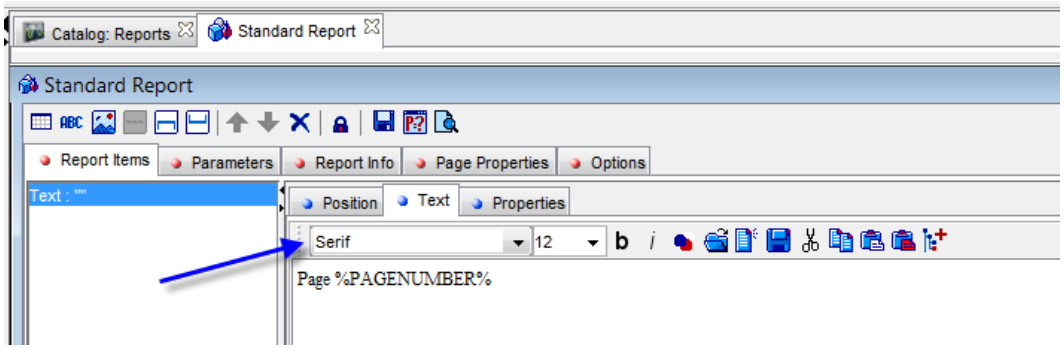
Internal reports are deprecated and should not be used in the first place, but in reality, they will still be used for some time before they get phased out.

Internal reports show Chinese characters correctly only if the correct font is set for all texts and table columns in the report (it has to be either *Dialog*, *Monospaced*,...see section Fonts).

The **default template report**, used when printing catalogs, usually contains only one table with no columns. Optionally, the table contains the Row Number column, in order to set various properties for the column, most importantly fonts. If this is the case, the font used for the column header and the column data needs to be one of the 5 standard fonts (this has to be changed by the customer).



If the default template report also contains any fixed texts (header/footer), it must be checked that the correct font is used and if not, it needs to be changed (again, by the customer).



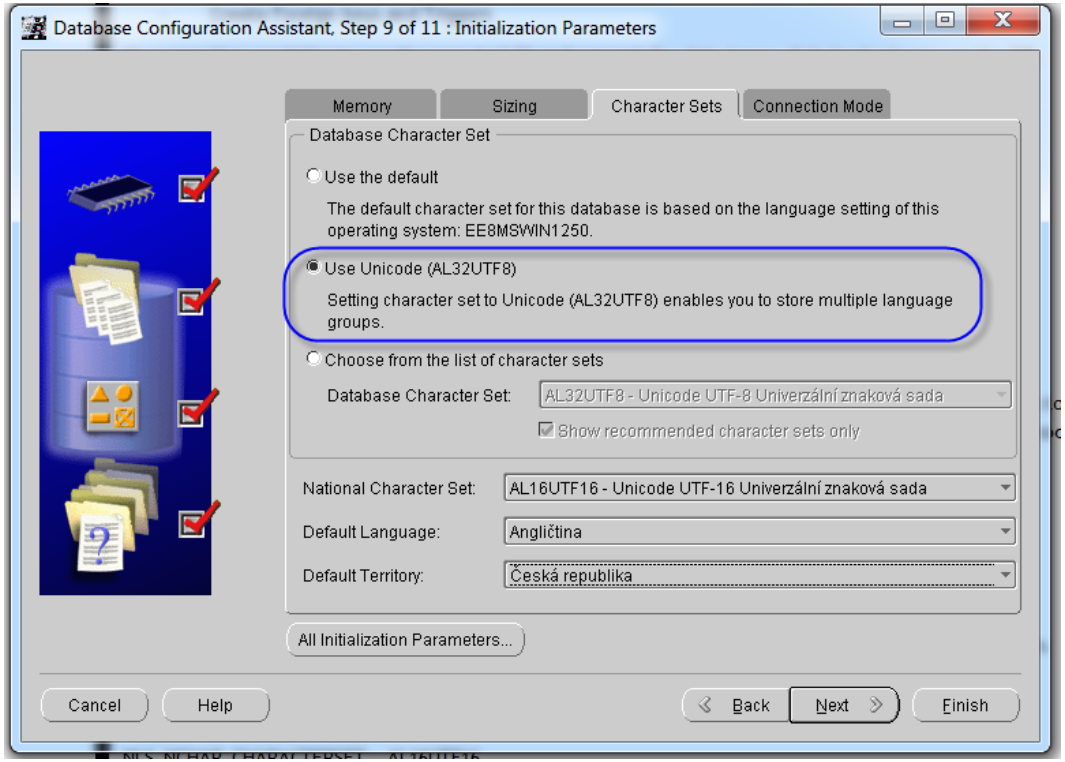
The correct font needs to be set also in other reports, which contain some texts and/or tables with particular columns. But, in reality it won't be necessary to change the fonts but to actually create new reports because the current reports contain hard coded English/German strings (usually names of attributes and titles) so they won't be used by Chinese customers.

Database

In order to make it possible to store and read strings containing other than Latin-based characters, it is necessary to install the database with the Unicode character set (if it's possible) or to change the data type of some character columns to a Unicode data type. This is database vendor specific, i.e. it is done differently on different database platforms.

Oracle

In general, Oracle has very good Unicode support – all that is needed is to select the Unicode database character set UTF-8 (called **AL32UTF8** by Oracle**)** when creating a new database:



Valuemation schema then needs to be created in this database using installation scripts.

Note: Oracle also supports **Unicode data types** (NVARCHAR2, NCHAR, NCLOB). This could be useful if you need to store Unicode characters only in new columns or new tables (added by the customer). However, if you want all character columns to support Unicode, using the UTF-8 character set is a much better and easier option.

If you have an existing Valuemation schema in a database that does not use the UTF-8 character set, then you need to

- export the Valuemation schema from the SOURCE database.
- if needed, create a new database with the UTF-8 character set (e.g. using the Database Configuration Assistant – see the screenshot above). This is the TARGET database.
- import the Valuemation schema into the TARGET database

It is important to realize that in UTF-8 database, one character may need more than 1 byte for storage (1 for ASCII characters, 2 for most other characters (e.g. accented characters), 3 for Asian characters, 4 for supplementary characters). Therefore the migration from single-byte character set to multi-byte character set may result in data truncation! Migrating from one database character set to another is a complex matter - it is necessary to perform additional steps that are explained in detail on this page.

Below you'll find a simplified version, which applies when migrating from a single-byte character set to UTF-8 :

Make sure you use the latest hotfix because the supporting tool in VM Console has been added only recently.

The following steps need to be done on the SOURCE database (with single-byte character set)

1) scan data to determine which columns need to be enlarged. Use the Oracle CSSCAN utility (see this page for detailed information)
   a. execute the `csminst.sql` script, which can be found in the Oracle installation directory. This script creates a DB schema CSMIG needed to collect the statistics. Run sqlplus from the command line:

   sqlplus sys/manager@orcl as sysdba

   @ `csminst.sql`

   b. run the CSSCAN utility from the command line:
   csscan 'sys/manager@orcl as sysdba'

   When prompted, use these options

- 1) full database 2) user 3) table 4) column: 2
- new database character set name: AL32UTF8
- enter user name to scan: <name of the Valuemation schema>
  otherwise use default options

  The result of the scan will be written into 3 files. The relevant files are scan.txt and scan.err. The scan.txt will most likely contain something like this:

[Application Data Conversion Summary]

Datatype	Changeless	Convertible	Truncation	Lossy
VARCHAR2	37.639.533	15.257	1.112	0
CHAR	18.264.372	0	0	0
LONG	0	0	0	0
CLOB	483.717	726.372	0	0
VARRAY	0	0	0	0
Total	56.387.622	741.629	1.112	0
Total in percentage	98,700%	1,298%	0,002%	0,000%

Application Data:

USER.TABLE|COLUMN	Convertible	Truncation	Lossy
VM_SQA_TEST.AMT_BPMFLOWOBJECT|SCRIPTCODE	68	0	0
VM_SQA_TEST.AMT_BPMFLOWOBJECT|VALIDATIONMSG	16	0	0
VM_SQA_TEST.AMT_BPMFLOWOBJECT|WORKITEMDESC	175	0	0
VM_SQA_TEST.AMT_BPMPROCESS|DESCRIPTION	4	0	0
VM_SQA_TEST.AMT_BPMVALUE|LONGPRESENTATION	62	0	0
VM_SQA_TEST.AMT_BPMVALUE|SHORTPRESENTATION	0	28	0
VM_SQA_TEST.AMT_BPMWORKITEM|DESCRIPTION	1.319	0	0
VM_SQA_TEST.AMT_CALENDAR_EVENT|DESCRIPTION	8.462	86	0
VM_SQA_TEST.AMT_CALENDAR_EVENT|GROUPNAME	21	0	0
VM_SQA_TEST.AMT_CALENDAR_EVENT|LOCATIONNAME	25	0	0
VM_SQA_TEST.AMT_CALENDAR_EVENT|SUBJECT	1.071	2	0

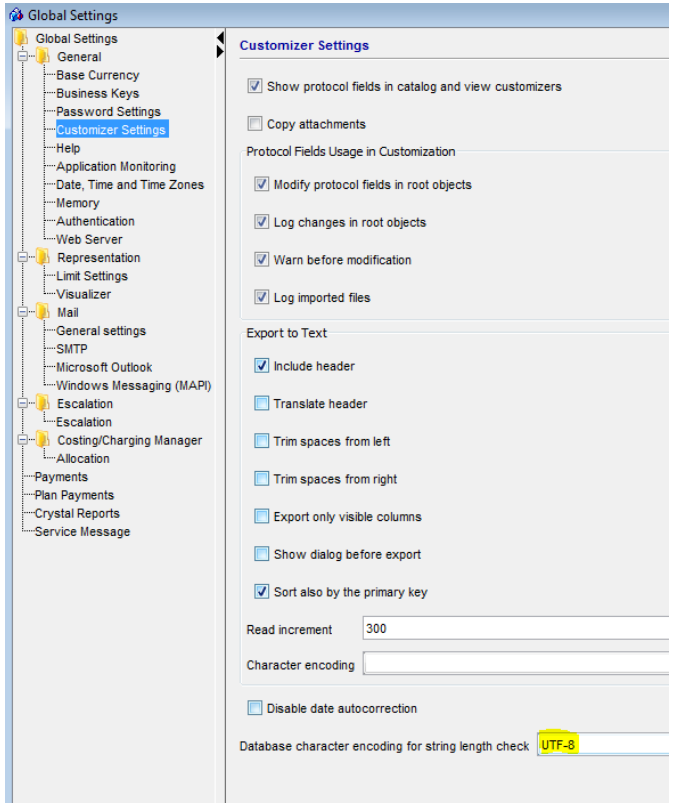
You will very likely see some positive numbers if the **Truncation** column. If so, some columns will need to be enlarged before exporting the schema.
Note: you should see only zeros in the Lossy column as the UTF-8 character set is a super set of all common single-byte character sets.

2) Generate DDL scripts using the VM Console:
    - copy the scan.txt and scan.err files to the VM directory
    - open VM Console
    - type 'analyse scan files' and press Enter
    - after a while, you should see 2 files in the VM directory:
        - source.sql – this files contains SQL statements that prepare the schema on the SOURCE database before it can be exported (essentially, it enlarges some columns and/or changes the data type of some column to CLOB). It should be executed on the SOURCE database.
        - target.sql - this files contains SQL statements that change the column type back to the original data type and length, but with the CHAR semantics, rebuilds some indexes and recompiles views. It should be executed on the TARGET database.
3) Execute the script source.sql on the Valuemation schema. You can do this directly in the VM Console by pressing Ctrl+F9 on the first line of the script.
4) Optionally, run the CSSCANNER utility again. Check the output in scan.txt – the column Truncation must contain only zeroes.
5) Export the Valuemation schema from the SOURCE database. Use the EXP or EXPDP utility to do so.

The following steps need to be done on the TARGET database (with UTF-8 character set):

6) execute ALTER SYSTEM SET BLANK_TRIMMING = TRUE SCOPE=SPFILE; Then restart the database server.
7) import the exported Valuemation schema created in step 5. Use the IMP or IMPDP utility to do so. You should see no exceptions related to exceeded data length when importing the schema!
8) execute the script target.sql created in step 2 on the Valuemation schema.
9) add system property to admin.bat on the line containing S3Application: -DcheckViews=true
10) run VM and start Metamodel Check for ‘Object Types’ with the repair option. This will detect the changed column length in bytes. Apply changes.

In VM 4.4 and older, it is then necessary to select the **UTF-8** code page in the Global Settings / Customizer Settings to let Valuemation know which code page is used by the database so that Valuemation can correctly check string lengths:



For more information on the String Length Check, read the document *String Length Check.doc*

DB2

The only option on DB2 is to create the database using the Unicode (UTF-8) code page – see this for information how to create such a database.

SQL Server

SQL Server does not support Unicode character set on instance nor database level. It only supports Unicode data types.

The character data types have to be changed to **Unicode data types**:

- char(n) → **nchar**(n) - fixed width string. n=max 4000 characters
- varchar(n) → **nvarchar**(n) - variable width string. n=max 4000 characters (or (max)=2^30)
- text → nvarchar(max) (text data type is deprecated)

These data types use **UCS-2** encoding, i.e. 1 character = 2 bytes

This needs to be done for most character columns in all or some generic tables (AMA_, AMS_, etc..) using a set of ALTER TABLE statements.
e.g.:

```
alter table AMA_SYSTEM alter column SHORTTEXT nvarchar(254)
go
alter table AMA_SYSTEM alter column NAME nvarchar(254)
go
......
.....
.....
```

But this is easier said than done. First of all, how do you create an DDL script with hundreds of similar DDL statements? The other problem is that the ALTER TABLE statement will fail if there is an index on the column or the column is constrained by any type of DB constraint (check/ default, foreign key, primary key). So what needs to be done is to drop all the indexes and constraints for the relevant columns and after changing the column data type, recreate those indexes and constraints. This makes creating such an DDL script even more difficult. You are free to write it manually or use any database tool to assist you in creating such a script.

But you can also use the **Valuemation Console** and its new feature, '**SQL Templates**', to create the migration script very easily!

Valuemation Console is a tool for developers, customizers, consultants and administrators (not for end users). It is used primarily for logging various events, most importantly SQL statements. But it can also be used to execute SQL/DDL statements and inspect or modify data in the DB. If you are not familiar with the Console, you should read the documentation first.

Now, let's look at the SQL Templates feature, which you can use to generate multiple SQL/DDL statements based on a template.

e.g. to select count from all tables, type

```
select count(*) from $TABLES$;
```

in the Console and press Ctrl+Shift+G to generate the SQL statements:

select count(*) from AMA_ACCESSADMIN;
select count(*) from AMA_ACCESSLEVEL;
select count(*) from AMA_ACTIVITYCOST;
select count(*) from AMA_ATTACH_HOLDER;
...
....

$TABLES$ is a *template tag* that is replaced in each SQL statement by the table name. There are other template tags ($COLUMNS$, $SCHEMAS$, $INDEXES$, $CONSTRAINTS$...).

Template tags can also be parameterized by using regular expressions. E.g. to select count from tables whose name starts with AMT_ , type:
select count(*) from $TABLES<name>AMT_.*$;

AMT_.* is a regular expression. Unfortunately, regular expressions in Java are quite complex and so the full explanation is beyond the scope of this document. I'm going to give just very short hints:
. means any character
* means repeated 0-N times
| means OR

Useful examples:
to match one specific name, use just the name: AMA_SYSTEM
to match specific names, separate them by | : AMA_SYSTEM|AMA_COMPONENT
to match names starting with a given prefix : AMA_.*
to match names ending with a given suffix : .*_ID
to match names containing a given word : .*WORD.*
to match names except specific ones: ^(?!(USERCHG|USERCRE|DATCRE|DATCHG|CLIENT)$).*
to match names except those that start with specific prefixes: ^(?!(AMA_ |AMS_)).*
to match names except those that end with specific suffixes: ^(?!.*(_ID|_IDX)$).*

When using $ in a regular expression, it needs to be escaped (doubled) in the template! E.g.

from $TABLES<name>^(?!.*(_ID|_IDX)$$).$;

More information on Java regular expressions can be found on many online sites. On this site you can test your regular expressions.

The complete documentation on SQL templates can be found on our Wiki.

The complete migration procedure:

WARNING: It is assumed and highly recommended to try this first on a COPY of the production database, **NOT ON THE PRODUCTION DATADABSE,** to test if there are any problems.

1) CHECK INDEXES

SQL Server has a maximum limit of 900 bytes in an index. For example, an index containing 2 varchar(254) columns occupies 508 bytes. Unicode data types use **UCS-2** encoding, which means that each character needs 2 bytes for storage. Thus, after converting character columns to Unicode data types, the same index need will now occupy 1016 bytes which exceeds the limit! When recreating the index, MS SQL Management Studio reports a warning: **Warning! The maximum key length is 900 bytes. The index 'I_COMPNETINFO_2' has maximum length of 1038 bytes. For some combination of large values, the insert/update operation will fail.** The index is created but with a warning. This means that if a combination of long strings is inserted into the column, such that it exceeds the limit in bytes, the operation will fail (e.g. inserting 2 strings of 226 characters each, which requires 902 bytes for storage).

To list such indexes, use this template:

```
--$TABLES<name>^(?!(AMT_|F\d\d)).*$ $INDEXES<sizeUnicode> >>900$
$INDEX.sql$
-- $INDEX.size$ --> $INDEX.sizeUnicode$
go
```

Copy this template to a console, select it and press Ctrl+Shift+g
You'll get something like this:

```
--AMA_FACILITY I_FACILITY
CREATE UNIQUE INDEX I_FACILITY ON VM_TEST.AMA_FACILITY (
  FACILITY_DESC,
  FACILITY_NO,
  CLIENT
)
-- 514 --> 1022

--AMA_IMPORTCSV I_IMPORTCSV_1
CREATE UNIQUE INDEX I_IMPORTCSV_1 ON VM_TEST.AMA_IMPORTCSV (
  IMPORTCSV,
  BOTYPE,
  CSV_SET_ID,
  CLIENT
)
-- 523 --> 1031
```

...

Below each index you'll see the original size in bytes and the size after the character columns in the index (except CLIENT) have been converted to Unicode data type (this new size will be greater than 900)

For each such index, you have the following options:

a) do nothing (i.e. the index will be recreated with a warning) – if the actual data in all the char/varchar columns is always shorter than the maximum column length, the insert/update will work just fine. But do not rely on this – before you decide for this option, you need to check the columns in the index and if you are not sure whether the data can be really as long as the column length, check the real values in the database – i.e. execute SELECT avg(len(<col1>)), max(len(<col1>)), avg(len(<col2>)), max(len(<col2>))... FROM <table>.

b) exclude one or more char/varchar columns in the index from converting to Unicode. This is recommended when such columns contain data, which never contain special characters (e.g. technical data like names of object types, attributes, statuses etc. – in general data not entered by end users).
To use this option, open the corresponding object type in the Object Type Customizer and for the column you wish not to convert to Unicode, set the extended property *UnicodeDisabled* to TRUE. Then, the generated DDL script from the templates will not contain such columns (because the template contains an expression tag with this property)
Note: the <sizeUnicode> expression tag takes this property into account.

c) shorten the length of one or more char/varchar columns. This is possible if the data in those columns are smaller than the newly set length – you need to check this - see the select statement in option a)

d) do not recreate the index again (only if it's NOT a unique index) – beware though, as this may negatively affect performance! If you choose this option, then comment out the CREATE INDEX statement from the generated DDL script created in the next section.

Standard indexes:

Even the standard database contains several indexes exceeding the limit (after conversion of columns to Unicode data type). So we have analyzed such indexes and the columns in them and chose the best option for each index. In most cases we chose option b), i.e. we set the *UnicodeDisabled* property for the corresponding columns because these column contains only technical data. These extended properties are part of the customization, delivered in a hotfix (VM 4.3.HF<XX>). This means that customers doing the migration will not have to analyze our standard indexes but only customer specific indexes.

However, for some indexes we chose the option a). These are:

I_FACILITY
I_PAY_AUDIT_COMP
I_SYSTEM_5
I_USAGETYPE

Before proceeding with the next step, if you selected options b) or c), make the change now – i.e. b) set the extended property c) shorten column length

2) GENERATE THE DDL SCRIPTS
Paste the following templates into the Console:

```
--
$TABLES<name>^(?!(AMT_|F\d\d)).*$ $CONSTRAINTS<type>(?!primary$$).*<column.type>varchar|char<column.name>^(?!(USERCHG|USERC
RE|DATCRE|DATCHG|CLIENT|MANDANT)$$).*<column.JavaType>java.lang.String|-$
alter table $SCHEMA$.$TABLE$ drop constraint $CONSTRAINT$
go

--
$TABLES<name>^(?!(AMT_|F\d\d)).*$ $CONSTRAINTS<type>primary<column.type>varchar|char<column.name>^(?!(USERCHG|USERCRE|DAT
CRE|DATCHG|CLIENT|MANDANT)$$).*<column.JavaType>java.lang.String|-$
alter table $SCHEMA$.$TABLE$ drop constraint $CONSTRAINT$
go

--
$TABLES<name>^(?!(AMT_|F\d\d)).*$ $INDEXES<name>^(?!PK_).*<column.type>varchar|char<column.name>^(?!(USERCHG|USERCRE|DATC
RE|DATCHG|CLIENT|MANDANT)$$).*<column.JavaType>java.lang.String|-$
drop index $INDEX$ on $SCHEMA$.$TABLE$
go

alter table $SCHEMA$.$TABLES<name>^(?!(AMT_|F\d\d)).*$ alter column
$COLUMNS<type>varchar|char<name>^(?!(USERCHG|USERCRE|DATCRE|DATCHG|CLIENT|MANDANT)$$).*<JavaType>java.lang.String|-
<property.UnicodeDisabled>false|-$ n$COLUMN.type$($COLUMN.size$) $COLUMN.notnull$
go

--
$TABLES<name>^(?!(AMT_|F\d\d)).*$ $INDEXES<name>^(?!PK_).*<column.type>varchar|char<column.name>^(?!(USERCHG|USERCRE|DATC
RE|DATCHG|CLIENT|MANDANT)$$).*<column.JavaType>java.lang.String|-$
$INDEX.sql$
go

--
$TABLES<name>^(?!(AMT_|F\d\d)).*$ $CONSTRAINTS<type>primary<column.type>varchar|char<column.name>^(?!(USERCHG|USERCRE|DAT
CRE|DATCHG|CLIENT|MANDANT)$$).*<column.JavaType>java.lang.String|-$
$CONSTRAINT.sql$
go
```

```
--
$TABLES<name>^(?!(AMT_|F\d\d)).*$ $CONSTRAINTS<type>(?!primary$$).*<column.type>varchar|char<column.name>^(?!(USERCHG|USERC
RE|DATCRE|DATCHG|CLIENT|MANDANT)$$).*<column.JavaType>java.lang.String|-$
$CONSTRAINT.sql$
go

EXECUTE sp_refreshview '$VIEWS$'
go

--$TABLESANDVIEWS<name>^(?!(AMT_|F\d\d)).*$
update $SCHEMA$.AMT_ATTRCOLMAP set IS_UNICODE = 'Y' where COLUMNNAME =
'$COLUMNS<type>varchar|char<name>^(?!(USERCHG|USERCRE|DATCRE|DATCHG|CLIENT|MANDANT)$$).*<JavaType>java.lang.String|-
<property.UnicodeDisabled>false|-$' and exists(select 1 from $SCHEMA$.AMT_GENERICBO gb where gb.GENERICBO_ID =
AMT_ATTRCOLMAP.GENERICBO_ID and TABLENAME = '$TABLE$')
go
```

There are 2 template tags that need explanation:

- $TABLES<name>^(?!(AMT_|F\d\d)).*$ means: : iterate over all tables whose name does NOT starts with AMT_ and it doesn't start with F<digit><digit>. By specifying more prefixes, you may exclude more tables. In any case, **technical tables (starting with AMT_) and tables populated by external systems which cannot handle Unicode data types must NOT be processed**!

- $COLUMNS<type>varchar|char<name>^(?!(USERCHG|USERCRE|DATCRE|DATCHG|CLIENT|MANDANT)$$).*<JavaType>java.lang.String|-<property.UnicodeDisabled>false|-$ means: iterate over all columns whose type is varchar or char and whose name is not USERCHG, USERCRE... and whose java type is String (or none – the column exists in the DB but it is not mapped to any attribute in VM) and which don't have a property UnicodeDisabled set to true (see this section for more information on this property).

Now, select each template (including the separator 'go'), one by one, and press Ctrl+Shift+G:

Worksheet

```
--$TABLES<name>AMA_.*|AMS_.*$ $CONSTRAINTS<column.type>varchar|char<column.name>^(?!.*(USERCHG|USERCRE|DATCRE|DATCHG|CLIENT)).*<column.size>(?!1).*$
alter table $SCHEMA$.$TABLE$ drop constraint $CONSTRAINT$
go

--$TABLES<name>AMA_.*|AMS_.*$ $INDEXES<column.type>varchar|char<column.name>^(?!.*(USERCHG|USERCRE|DATCRE|DATCHG|CLIENT)).*<column.size>(?!1).*$
drop index $INDEX$ on $SCHEMA$.$TABLE$
go

alter table $SCHEMA$.$TABLES<name>AMA_.*|AMS_.*$ alter column $COLUMNS<type>varchar|char<name>^(?!.*(USERCHG|USERCRE|DATCRE|DATCHG|CLIENT)).*<size>(?!1).*$ n$COLUMN.type$($COLUMN.size$) $COLUMN.notnull$
go

--$TABLES<name>AMA_.*|AMS_.*$ $INDEXES<column.type>varchar|char<column.name>^(?!.*(USERCHG|USERCRE|DATCRE|DATCHG|CLIENT)).*<column.size>(?!1).*$
create $INDEX.unique$ index $INDEX$ on $TABLE$ ($INDEX.columns$)
go

--$TABLES<name>AMA_.*|AMS_.*$ $CONSTRAINTS<column.type>varchar|char<column.name>^(?!.*(USERCHG|USERCRE|DATCRE|DATCHG|CLIENT)).*<column.size>(?!1).*$
$CONSTRAINT.sql$
go
```

Ctrl+Shift+g

On separate tabs, you should get several sets of DDL statements:

- to drop some constraints

```
--AMA_ACTIVITYCOST I_ACTIVITYCOST
drop index I_ACTIVITYCOST on dbo.AMA_ACTIVITYCOST
go

--AMA_BILLTERM I_BILLTERM_1
drop index I_BILLTERM_1 on dbo.AMA_BILLTERM
go

--AMA_BUDGET I_BUDGET
drop index I_BUDGET on dbo.AMA_BUDGET
go

--AMA_BUSINESSPART I_BUSINESSPART
drop index I_BUSINESSPART on dbo.AMA_BUSINESSPART
go
```

- to drop some indexes

```
--AMA_PLAN DF__AMA_PLAN__PRIORI__2E70E1FD
alter table dbo.AMA_PLAN drop constraint DF__AMA_PLAN__PRIORI__2E70E1FD
go
```

- to change the data type of some columns

```
alter table dbo.AMA_ACCESSLEVEL alter column ACCESSLEVEL nvarchar(254)
go

alter table dbo.AMA_ACCESSLEVEL alter column ENVIRONMENT nvarchar(254)
go

alter table dbo.AMA_ACCESSLEVEL alter column DESCRIPTION nvarchar(254)
go

alter table dbo.AMA_ACCESSLEVEL alter column SHORTTEXT nvarchar(254)
go

alter table dbo.AMA_ACTIVITYCOST alter column ACTIVITYTYPE nvarchar(254)
go
```

- to recreate the indexes

```
--AMA_ACTIVITYCOST I_ACTIVITYCOST
create UNIQUE index I_ACTIVITYCOST on AMA_ACTIVITYCOST (ACTIVITYTYPE, CLIENT)
go

--AMA_BILLTERM I_BILLTERM_1
create UNIQUE index I_BILLTERM_1 on AMA_BILLTERM (BILLTERM, CLIENT)
go

--AMA_BUDGET I_BUDGET
create UNIQUE index I_BUDGET on AMA_BUDGET (ACCOUNT_NO)
go

--AMA_BUSINESSPART I_BUSINESSPART
create UNIQUE index I_BUSINESSPART on AMA_BUSINESSPART (BUSINESSPART_NO, CLIENT)
go
```

- to recreate the constraints

```
--AMA_PLAN DF__AMA_PLAN__PRIORI__2E70E1FD
ALTER TABLE dbo.AMA_PLAN ADD CONSTRAINT DF__AMA_PLAN__PRIORI__2E70E1FD DEFAULT ('1') FOR PRIORITY
go
```

- to refresh DB views
- to update meta information (in AMT_ATTRCOLMAP) – VM needs to know that the column has a Unicode data type (nchar or nvarchar)

Now save the content of each tab to a file (copy and paste), make sure the name contains a number indicating the order!
Example:
migration1.sql
migration2.sql
migration3.sql
....

3) **EXECUTE THE GENERATED DDL SCRIPTS**

Now you need to execute all the generated DDL scripts **in the same order**. We recommend to execute them directly in the **Microsoft SQL Management Studio** – simply open all the DDL scripts and execute them one by one (button Execute (F5)).

Make sure there are no errors before you proceed to another DDL script. If for any reason a constraint or an index could not be dropped, the column data type change fill fail! In such a case, you have to resolve those problems first before proceeding otherwise you will get more errors caused by the previous errors. Note that the DDL script that actually changes the data type takes a long time to execute (15-30 minutes) so be patient.

If you were able to execute all the DDL scripts without an error, you can combine them into one DDL script (in the same order!). This script can be then executed on the production database.

Note: when testing, you can execute the scripts also directly in the Console – on each tab, position the cursor on the first line of the first generated script and press Ctrl+F9. Then commit the change (F11) – for some reason this is needed!

F56 table:

This table contains incoming emails and is populated by the O-server. A later version now supports the Unicode data types. To change the data types in this table, it is necessary to create a temporary table F56x having Unicode data types, insert data from the original F56 table to this temporary table, drop the F56 table and rename F56x to F56. This is the complete script:

```
CREATE TABLE F56x (
        F56000 decimal(9) NOT NULL,
        F56001 nvarchar(12),
        F56002 decimal(9),
        F56003 nvarchar(max),
        F56004 nvarchar(256),
        F56005 datetime,
        F56006 nvarchar(256),
        F56007 nvarchar(48),
        F56008 decimal(9),
        F56009 nvarchar(24),
        F56010 nvarchar(1000),
        F56011 nvarchar(1000),
        F56012 nvarchar(2000),
        F56013 nvarchar(250),
        F56014 nvarchar(48),
        F56015 datetime,
        F56016 nvarchar(6),
        F56017 nvarchar(6),
        F56018 nvarchar(6),
        F56019 nvarchar(1),
        F56020 datetime,
        F56021 decimal(9),
```

```
        F56022 nvarchar(1),
        F56023 datetime,
        F56024 datetime,
        F56025 decimal(6),
        F56026 decimal(6),
        F56027 nvarchar(250),
        F56028 nvarchar(250),
        F56029 nvarchar(250),
        F56030 decimal(9),
        F56090 nvarchar(48),
        F56091 datetime,
        F56092 decimal(9),
        DATCRE datetime,
        DATCHG datetime,
        USERCRE varchar(254),
        USERCHG varchar(254),
        CLIENT char(6),
        CONSTRAINT PK_F56x PRIMARY KEY (F56000)
)
go

insert into dbo.F56x select * from dbo.F56
go

drop table dbo.F56
go

EXECUTE sp_rename F56x, F56
go

CREATE INDEX SK2_F56 ON dbo.F56 (
  F56001,
  F56009
)
go
```

4) **SETTINGS**

If you are using Valuemation version 4.3 or 4.4:

Make sure the **sessions_MSSQL2005.properties** file contains this line:

SessionName.VALUEMATION.database=%DBASE%;SelectMethod=cursor;**sendStringParametersAsUnicode**=**true**

i.e. the parameter **sendStringParametersAsUnicode** has to be set to **true** (this is actually the default value).

If you are using Valuemation version 4.5 or newer:

the parameter **sendStringParametersAsUnicode** should be set to **false** for optimal performance!

Now you should be able to store strings to database in any language (including Asian languages).