ActiveNet Multi-Lingual Support

# Overview Servlet: Embedded text

## General

Contains what would be a full scope for multi-lingual support, in all parts of the application. Some of this may be deferred to later releases or indefinitely.

## Design aims

Support any number of additional languages (English is default)

Allow Active or a contractor to provide a translation into another language, which becomes part of the deployed build.

Allow organizations to override the English or localized text, in such a way that these overrides are preserved when a new version is deployed.

Mobile support is only available if all CUI pages are implemented in ACM platform.

## Types of text

### Servlet static

All HTML/HTX/TXT static text

All error message text

All other text created by the servlet (e.g., portions of pages) containing static text

Applet static text

Flex static text

### CUI static

All CUI labels, headings, etc.

All error messages

### Localizable DB fields

Selected database fields will be language-specific.

When localized text is available for a data element, it will appear in:

* CUI pages
* AUI pages
* Applet pages
* Flex
* Selected reports

### Reports

For selected reports, labels and DB text will be localized

# Language configuration

## Global Configuration

Multi-lingual features turned on by a license key

Org can configure which languages other than English are supported by their site (standard ISO 639 language / culture keys)

If not licensed, or no other languages are configured, application behavior is default.

## AUI language selection

Display language is per session

For AUI, preferred language is stored at the system user level

Default language can be edited in system user edit page

User can change preferred language at login time (if >1 language configured)

User can change language from any page and optionally update their profile

## CUI language selection

For CUI, preferred language is stored at the customer level

Preferred language can be edited in customer edit page

User can change language at login time (if >1 language configured)

User can change language from any page and optionally update their profile

Language selection must be synchronized with servlet for screen-scraped pages

# Static text localization

## Static text conceptual model

HTML files, servlet code, reports, CUI custom controls will all contain token names, which get resolved to text.

Three-level design for static text: The process of actually generating a piece of programmatic text (html text, report labels, error message, etc.) should look in three places:

* The base token text provided by the developer (in English)
* Overridden by translated text for the token, provided by Active or a contractor in some target language,
* Overridden by org-provided text in a specific language.

Storage location for text:

* Text developed by Active or a contractor as part of a release would be in text resource files, easily deployed as part of the version image.
* Org override text would be in the database

This model might be enhanced to support language cultures; e.g., there could be a mapping for “fr” (default French) and “fr-ca” (Canadian French). I don’t think this resolution needs to be built in initially, but it should be easy for the model to support it.

## Token model

The structural change for the developer is that programmatic UI text, such as in an HTML page, no longer ends up containing actual text; instead, it contains "tokens" which identify the string.

There will be a single token space for all usages; i.e., a common place strings are stored whether they are in HTML pages, Flex pages, applets, reports, error strings, etc.

The servlet will provide the token resolution methods; CUI, applet and Flex will make requests to it.

For simplicity of conversion, the token name will be the current English text. So if a label says “Activity name”, this will be replaced with a lookup of the token named “Activity name”.

This also means by default, the token “Activity name” will automatically effect every usage in all UIs, unless there is a specific reason to display it differently, in which case the usage will be changed to a different token.

This default should allow the development of a utility to automatically tokenize at least the HTML/HTX files.

Token resolution should be recursive, so it is possible to have a token have a value like “Please enter the `activity name`”, to allow the surrounding text to be in different orders for different languages.

??? Honor existing org text customization

## CUI static text model

For custom controls which provide label text (say a label/text box combination), the label text will become the token name.

CUI pages which don’t use custom controls would need to be converted to use them.

The custom control will call a method to make servlet requests to resolve tokens to text per language.

The CUI must have a caching model for localized token text.

## Applet/Flex static text model

Similar to CUI, applet will make a request to a servlet method to resolve token.

Servlet will know language of current session

## Report static text model

ActiveReport code would behave like CUI …

Could possibly loop through controls and auto-convert any report

## Org localization UI

New user permission to turn on org customization mode

### Tabular editing

Ability to list and search for tokens from master list from default resource file.

Class application has a model.

### In-place editing in servlet

A new icon would be displayed next to text element.

Popup would allow show uncustomized and customized text for each supported language.

In CUI, would be built into custom control

In servlet, could be implemented as an HTX; in fact, the label/icon combination could be built as an HTX custom control.

Servlet-generated text like error messages may be out-of-scope

??? auto-refresh page

### In-place editing in CUI

Customization icon built into custom control

Popup for customization looked like servlet’s

WS API to get / update multi-lingual values

# Localizable DB fields

Because of the additional complexity in the UI, DB access, etc., thought should be given to strictly limiting which text attributes have a display usage which requires customization. E.g., activity name will definitely need to be localizable, but customer name is definitely not localizable.

## Database design

For backward compatibility and best performance in the default case, English text will be kept in-row.

Alternate language text can be stored in companion tables, with common primary key, a language key, and only the localizable fields.

For reports and anywhere else that direct queries are used (rather than cache access), a localization views could be used. The localization view for a localized table would have all the columns of the base table, plus the language key. For localized columns, it would return the localized value if present otherwise the base table value.

## Servlet modifications

For modifications within the servlet, the design must have a simple model for customizing the class containing that data element, with minimal or no modifications throughout the application:

* Internally to the class, the values should be stored in a collection keyed by language code.
* Cache loads all available companion records for each object.
* Access by other classes should be via a getter method; e.g., the Activity.activityName method. This method should return the appropriate value for the selected language of the session.
* Parameter puts and gets of the existing keys should get/put the value for the language of the current session.
* For support of configuration editing of multiple language values, new language-specific hash table keys should be put/got.

## Upsizer support

With appropriate naming convention on the companion table name and language key, Upsizer could:

* Create the localized views.
* Create all character columns in companion tables as nvarchar
* Validate companion table

## Display and editing

Display block of controls with all supported languages in standard editing page

## ACM support

Assumption is that ACM can only display, but not edit, localized DB text.

WS methods must return a language-keyed collection of text elements

Language-specific value displayed

## Applet/Flex support

Implementation will be similar to ACM

## Report modifications

For selected reports such as receipt, localizable fields should be displayed in appropriate language:

* Report system must be told language to use
* Report changed to use view

# Developer support tools for static text

## Tokenization utility for servlet HTML/HTX

It should be possible to build a utility which will scan at least HTML for untokenized strings, update the default text file, keeping it alphabetized by token name, and replace the strings with tokens. This will not only be useful in initial conversion, but also can allow a developer to do initial development of a page without separately managing the string table, then post-process it.

## Automatic generation of the default static text resource

In development mode, it would be possible to have the servlet, when receiving a token request not currently in the static file, update the resource file with an identity entry (token name = value), keeping it alphabetized by token name. Between this and the tokenization utility, there will be very little need for developers to directly edit the language file.

## Text in code

Some investment in tools or approaches to find display text embedded in the code would be useful.

* Error messages
* HTML which is build
* Enum values

# Translator support tools for static text

It must be possible for staff or contract translators to easily determine for a new release what new or modified text elements must be translated.

## Via SVN

SVN could easily provide basic capability with some simple conventions on the resource file:

* There should be a separate resource file for each language, with the ISO language code as part of the naming convention.
* The file entries should be alphabetized by token name.
* Two-column CSV file would make for easy management in Excel, and easy diffing.
* There should only be entries for tokens with localized values.

## In-place editing

While directly editing the source files is useful for an initial pass, during testing of translations, it would be useful to fix wrong translations in place.

* Only available on test sites
* Separate permission for translators
* Similar UI model with icon, and pop-up editor
* Directly rewrites the resource file

# Open items:

The above are thoughts about how to most easily fit localization into the existing application architecture. It should be reviewed against standard localization models.

Non-European languages: Are they out-of-scope?

??? Multi-lingual for asset service

# Estimates

|  |  |
| --- | --- |
| **WAG (hr)** | **Task** |
|  | **Framework proof of concept work** |
| 12 16 | Global configuration: License key, supported language key configuration |
| 12 16 | Language changer on HTML and CUI pages |
| 80 | HTML tokenization proof of concept: Tokenization of one page: Default language resource file, alternate language resource file, org override table, token resolution method and de-tokenization of HTML |
| 40 | Proof of concept of ACM static localization: Servlet token resolution WS API, custom control text resolution via API |
| 16 | Proof of concept of applet tokenization: One label customized |
| 16 | Proof of concept of Flex tokenization |
| 16 | Proof of concept of receipt static text tokenization: One label customized |
| 80 | Proof of concept of DB localization in servlet: Activity name: Companion table, caching, default language display; multi-lingual editing. |
| 40 60 | Proof of concept of DB localization in CUI: Activity name: WS method changes for localized text, language-specific text displayed. (Caching) |
| 16 | Proof-of-concept of DB localization in applet |
| 32 | Proof of concept for receipt DB localization: sending language to report, using view |
| 8 | Proof of concept of code-resident text resolution (e.g., error messages) |
|  | **Framework buildout** |
| 32 | Servlet session language control: customer login, system user login, |
| 32 | ACM session language control: customer login, system user login, language sync with servlet |
| 120 | Tokenization utility for HTML |
| 16 | Servlet automatic creation of default language resource entries |
| 32 40 | All existing CUI custom controls re-route to servlet tokenization |
| 32 | Upsizer support for DB localization |
| 24 | Tabular editor for org override text |
| 40 80 | Servlet in-place editor for org override text (static text) |
| 40 80 | CUI in-place editor for org override text (static text) |
| 60 | Servlet in-place editor for static translation (optional?) |
| 60 | CUI in-place editor for static translation (optional?) |
| 80 | CUI caching of tokenized static text by org/token/language |
| 80 | Allocation to invest in tools for locating / tokenizing text in code. |
| ??? | Multi-lingual asset service support |
|  | **Bulk work** |
| 60 | Servlet pages: test and fix any problems resulting from tokenization utility  2786 files  264890 lines |
| 80 | Conversion of all existing CUI pages to use custom controls |
| 80 | Complete tokenization of static text in applets |
| 80 | Complete tokenization of static text in Flex |
| 480 | Servlet modifications for all DB localization  120 fields \* 4h = 480 |
| 208 | CUI modifications for all DB localization  (for displaying French activity name etc)  100 fields \* 2h = 200  10 types of assets, change data type to list of String: 8 |
| 80 | Flex modifications for all DB localization |
| 60 | Applet modifications for all DB localization |
| 240 | Servlet: Embedded text  p.put(String) 3286 usages  p.put(StringBuilder) 189 usages |
| 80 | CUI embedded text |
| 8 | Receipt static text localization |
| 8 | Permit static text localization |
| 8 | Statement static text localization |
| 8 | ? Pass printing report change |
| 40 | ? DB localization of 4 reports (using views in reports) |
| 400 | ? DB localization of all reports (using views in reports) |
| 0~400 | ? Static localization of all reports |
| TBD | Initial translation of resource file into French |
| TBD | Initial translation of resource file into Spanish |
| 10000 | Implement All CUI pages natively in ACM |