Programming Notes

For details on any subjects in this section, refer to the Aras Innovator - Programmer's Guide. This section is only meant to be a reference to areas of interest for programmers. The Aras Innovator UI automatically accounts for language, time, and locale variables for the end user, but the programmer needs to know what to look out for when writing Solutions, Solutions Add-Ons, and Integration projects.

Important

Due to differences between Windows and Linux in culture and date and time formatting it is recommended to specify culture and format explicitly. For more information about cross-platform development please see section *Cross-platform development* in *the Aras Innovator - Programmer's Guide*.

AML

AML must be written to support the multi-lingual properties and the locale neutral data. When querying for AML, you must understand the format of the return. When applying AML, you must be sure to apply multi-lingual values to the correct language, apply DateTimes in the correct time zone, and properties with locale neutral data must have the correct formats.

Multi—Lingual Properties.

Multi-lingual properties should indicate the specific language being passed, if not the default (English). Multi-Lingual properties now contain the xml:lang attribute to indicate what language is returned.

Get Queries

Let us first consider a query with action='get' from an English client.

```
<Item type="ItemType" id="B88C14B99EF44982…" action="get" select="label"/>
```

This query returns the label in the language of the client session established at the time of connection value of the label property. If there is no locale matching the client at the time the session is established, the default (English) is returned.

```
<Item type="ItemType" id=" B88C14B99EF44982…">
<label xml:lang="en">Document</label>
</Item>
```

From a German client, the return would include the German label

```
<Item type="ItemType" id="B88C14B99EF44982…">
<label xml:lang="de">Dokument</label>
</Item>
```

Requests may specify the language(s) desired in the response by including the ‘language’ attribute on the Item tag. The value of this attribute may either be a comma-delimited list of language codes or an asterisk (*) to request all languages.

```
<Item type="ItemType" id="B88C14B99EF44982…" action="get" select="label" language="en,de"/>
```

To return

```
<Item type="ItemType" id=" B88C14B99EF44982…" >
<i18n:label xml:lang="de" xmlns:i18n="http://www.aras.com/I18N">Dokument</i18n:label>
<i18n:label xml:lang="en" xmlns:i18n="http://www.aras.com/I18N">Document</i18n:label>
</Item>
```

If the language attribute is not present, the AML should be interpreted to be requesting session-language values for all multilingual properties. The fallback behavior is to return the default-language value for any property with no session-language value.

Add/Edit/Update Queries

When adding or editing an Item through an AML query, the most important thing to keep in mind is that unless the namespace is defined, the session language is updated. This is either the language defined by the session locale, or the default(English) language for locales not defined in the Aras Innovator Database

```
<Item type="ItemType" id="B88C14B99EF44982…"action="edit">
<label>Document<label>
</Item>
```

The following is the result for an en-US session:

```
<Item type="ItemType" id=" B88C14B99EF44982…" >
<i18n:label xml:lang="de" is_null="1" xmlns:i18n=http://www.aras.com/I18N />
<i18n:label xml:lang="en" xmlns:i18n="http://www.aras.com/I18N">Document</i18n:label>
</Item>
```

While the same query displays the following result for a de-DE session:

```
<Item type="ItemType" id=" B88C14B99EF44982…" >
<i18n:label xml:lang="de" xmlns:i18n="http://www.aras.com/I18N">Document</i18n:label>
<i18n:label xml:lang="en" is_null="1" xmlns:i18n="http://www.aras.com/I18N" />
</Item>
```

In order to set a language value, other than the default, of a property you must specify the namespace as a property attribute.

```
<Item type="ItemType" id=" B88C14B99EF44982…" >
<i18n:label xml:lang="de" xmlns:i18n="http://www.aras.com/I18N">Dokument</i18n:label>
<i18n:label xml:lang="en" xmlns:i18n="http://www.aras.com/I18N">Document</i18n:label>
</Item>
```

If the namespace is not explicitly defined, then the value is assumed to be the value for the session language.

DateTime Properties

DateTime properties are passed to and from the Innovator Server in a neutral data format and the time zone may vary based on a fixed set of rules. DateTime values passed to\from the server don’t contain milliseconds as fractions of a second and are ignored by the Innovator. Milliseconds are saved on system properties.

Get Queries

When reading the returns of a query with action="get" there are two main rules to keep in mind.

- If the CorporateTimeZone variable is set in the database, refer to section Time Zones , then all DateTime properties are returned in the corporate time zone.
- If the CorporateTimeZone is NOT set, then all DateTime properties are returned in the local time zone of the client session that applied the AML.

This includes AML applied by methods executed as a result of client actions in the UI.

Add/Edit/Update Queries

When applying AML to Aras Innovator, there are three main rules to keep in mind.

- If the CorporateTimeZone variable is set in the database, refer to section Time Zones , then all DateTime properties must be applied in the corporate time zone.
- If the CorporateTimeZone is NOT set, then all DateTime properties must be applied in the local time zone of the client session context that applied the AML.
- The DateTime applied in the AML must conform to the pattern yyyy-MM-dd[Thh:mm:ss]

> **Important**
>
> a) the optional portion is enclosed in "[]"; b) the format doesn't contain milliseconds as they are ignored by the server).

This includes AML applied by methods executed because of client actions in the UI.

There is one special case when applying updates to datetime properties in AML to Aras Innovator. It is possible to specify '__now()' as the value for properties of type datetime, instead of a specific value.

```
<Item type="Document" id="ACD123C46C1BFE8ED4E8BDDE0B009812">
<effective_date>__now()</effective_date>
</Item>
```

When a request similar to this is received by the server, it writes the value of the current moment of time into the database for the date property effective_date.

Locale Neutral Data Formats

Date, decimal, and float properties are displayed in the Aras Innovator UI based on the patterns in the client sessions Regional and Language settings, however the AML does not follow this rule. AML for these property types is sent in a neutral data format.

Get Queries

An AML query with action="get" returns date, decimal, and float properties in a data neutral format. Conversion of these values to a locale specific format should be done through client javascript.

```
<Item type="Test Item" id="D95C15B649AC46C1BFE8ED4E8BDDE0B0" action="get"/>
```

Returns

```
<Item type="Test Item" id="D95C15B649AC46C1BFE8ED4E8BDDE0B0">
<date_prop>2006-12-01T08:12:45</date_prop>
<decimal_prop>1234.56</decimal_prop>
<float_prop>-1234.56</float_prop>
</Item>
```

There are some external solutions where the AML returned must be in the patterned format, rather than the neutral data format. For these cases, you can use action='do_l10n' initial_action='get' to convert dates to the client session patterns. This is not recommended over the use of javascript on the client page, as it could have performance costs.

Example:

If you run the following query from a client with locale of ru-RU

```
<Item type="Test Item" id="D95C15B649AC46C1BFE8ED4E8BDDE0B0" action="do_l10n" initial_action="get/>
```

Aras Innovator returns

```
<Item type="Test Item" id="D95C15B649AC46C1BFE8ED4E8BDDE0B0">
<date_prop neutral_value="2006-12-01T08:12:45">01.12.2006</date_prop>
<decimal_prop neutral_value="1234.56">1234,56</decimal_prop>
<float_prop neutral_value="-1234.56">-1234,56</float_prop>
</Item>
```

Add/Update/Edit Queries

Date, decimal, and float properties must be in a locale neutral data format for any applied AML query to Aras Innovator.

```
<Item type="Test Item" id="D95C15B649AC46C1BFE8ED4E8BDDE0B0" action="add">
<date_prop>2006-12-01T08:12:45</date_prop>
<decimal_prop>1234.56</decimal_prop>
<float_prop>-1234.56</float_prop>
</Item>
```

IOM

Always keep the following statements in mind when dealing with locale specific types using IOM:

- The format of property values that are local specific (‘date’, ‘float’, ‘decimal’) are always presented in AML in the neutral format.
- Property values of type ‘date’ are always presented in AML in the time zone of the current session
- Dates are always stored in the database in UTC (no time zone)

To support conversion to\from neutral format as well as get information about the current session locale\language\time zone\etc. `IOM.Innovator` class has a new method `getI18NSessionContext()` that returns an instance of a class that implements `I18NSessionContext` interface. The interface allows you to get information about the session locale\language\etc. (`getLocale(), getLanguageCode(),` etc.) as well as containing methods for converting to\from neutral format (`ConvertToNeutral(..), ConvertFromNeutral(...)`) and methods specific to converting to\from UTC date-time `(ConvertUtcDateTimeToNeutral(..), ConvertNeutralToUtcDateTime(..)`).

Here is the exact format of the conversion methods:

```
string ConvertToNeutral (string value, string type, string format )
```

where,

- ‘ `value` ’ is a string that has to be converted (e.g. ‘12/25/2003’)
- ‘ `type’` is name of an Aras Innovator type where representation is locale dependent; this parameter can be one of the following `‘date’ | ‘float’ | ‘decimal’`
- ‘ `format’` – format of the ‘ `value’` . If the parameter is either a null or empty string then an attempt is made to parse the passed ‘ `value’` assuming that the ‘ `value’` has a format that is valid for the session locale; if parsing failed then an exception is thrown.

```
string ConvertFromNeutral( string value, string type, string format )
```

where,

- ‘ `value’` is a string in neutral format that has to be converted (in most cases the string would be a property value taken from AML)

- ‘ `type’` is the name of an Aras Innovator type where representation is locale dependent; this parameter can be one of the following `‘date’ | ‘float’ | ‘decimal’`
- ‘ `format’` – the parameter is taken into account only `if ‘type’=’date’` in which case it’s one of the following: `‘short_date’ | ‘long_date’ | ‘short_date_time’ | ‘long_date_time’ | ‘long_time’ | ‘short_time’` .

Important

In each case, the string representation of the ‘value’ returned by the method depends on the locale of the session (in other words: conversion to, let’s say, ‘short_date’ of a particular ‘value’ results in different short date format representations of the ‘value’ depending on the session locale).

```
string ConvertUtcDateTimeToNeutral( string utcDateTime, string inFormat );
```

The method returns a string that represents ‘ `utcDateTime’` in the time zone of the session and in neutral format. Parameters are:

- ‘ `utcDateTime’` is a string that is interpreted either as UTC date-time if it doesn’t have an offset or has offset 0 or as the exact moment of time if it has an offset.
- ‘ `inFormat’` defines the format of the ‘ `utcDateTime’` string.

Important

If ‘inFormat’ is null then an attempt is made to parse passed ‘utcDateTime’ assuming that it has a valid invariant culture format.
If ‘inFormat’ is not null then it has to represent a valid invariant culture date\time format.

`string ConvertNeutralToUtcDateTime( string neutralDateTime, string outFormat );` the method returns a string that represents the corresponding UTC date-time in the specified format.

- ‘ `neutralDateTime’` is a string that represents a date-time value in the session time zone in neutral format
- ‘ `outFormat’` is the format in which the resulting UTC date-time is returned.

Let’s consider several specific use-cases in order to understand i) what the above statements mean in each situation; ii) how I18NSessionContext methods can help.

The User writes a client method that makes a request to the server and then converts the obtained value of the received property ‘ `modified_on’` of type ‘date’ into the ‘ `long_date’` format of the locale of the current session in order to display it in UI:

…

```
// Make request to the server and get the value of the "modified on" property
var request = this.newItem("Issue","get");
request.setProperty("id","4A6478F0C0CB46F1B24D9526223FA14F");
var response = request.apply();
var issue_date_str = response.getProperty("modified_on");
// At this point the value of “issue_date_str” is something like:
//   2007-12-25T15:23:30
// It has to be converted to the format that appropriate for the
// locale of the session:
var cntx = this.getInnovator().getI18NSessionContext();
var new_value =cntx.ConvertFromNeutral(issue_date_str,'date','short_date’);
// At this point the “new_value” would look as the following (let’s assume
// that the session locale is “en-US”; also remember that we used ‘short_date’
// format so time portion was cut):
//   12/25/2008
…
```

The User writes a client method that reads a form field that contains a date; the user wants to build an AML putting the value read from the field into the AML property ‘ `effective_date’` :

…

```
// Read the value from the field
var duedate = document.forms[0].new_due_date.value;
// The “duedate” looks as the following (let’s assume that session locale is
// “fr-FR” (French(France)):  25/12/2007
// Convert to neutral format because according to the statement a.
// all dates in AML must be in neutral format:
var cntx = this.getInnovator().getI18NSessionContext();
var dd_neutral = cntx.ConvertToNeutral( duedate, ‘date’, ‘’);
// The value of “dd_neutral” would look like: 2007-12-25
// Now create item and set its property ‘effective_date’
var newPart = this.newItem("Part","add");
newPart.setProperty("item_number","xxx");
newPart.setProperty("effective_date",dd_neutral);
```

…

The user writes a stand-alone application using IOM to access Aras Innovator and needs to form a correct AML for properties of type `‘date’:`

```
...
// Instance of IOM.Innovator is available as ‘inn’
I18NSessionContext cntx = inn.getI18NSessionContext();
Item my_part = inn.newItem( ‘Person’, ‘add’ );
// Let’s assume that user of the application entered the date-time value
// in a text field; this value is a) in the time zone of the machine which is also
// the time zone of the current Aras Innovator session; b) in the ‘long_date_time’
// format of the session locale (let’s assume it’s “en-US”). Before the conversion
// the date looks like “Friday, October 10, 2008 01:00:00 PM”; after the convertion –
// “2008/10/10T13:00:00”
string dob_val = , cntx.ConvertToNeutral(inputText.Text, ‘date’, ‘’);
// Finally set the property
my_part.setProperty( ‘date_of_birth’, dob_val );
…
```

The user writes a stand-alone application using IOM to access Aras Innovator and needs to show dates obtained in AML in the short-date-time format of the machine locale (which is also the locale of the current session):

```
...
// Instance of IOM.Innovator is available as ‘inn’
I18NSessionContext cntx = inn.getI18NSessionContext();
// Make the request to get data
Item request = this.newItem( ‘Part’, ‘get’ );
request.setProperty( ‘item_number’, ‘xxx’ );
Item response = request.apply();
string part_date_str = response.getProperty( ‘modified_on’ );
// ‘modified_on’ date is in the time zone of the current session (according
// to the statement b) and in neutral format (according to the statement a).
// So before the conversion the date would like: “2005/09/17T11:22:35”,
// after the conversion (let’s assume that the locale is “fr-FR”):
// “17/09/2005 11:22:35”
inputText.Text = cntx.ConvertFromNeutral( part_date_str, ‘date’, ‘short_date’ );
```

The user writes a server `‘onBefore’` method in which the user needs to compare the created_on property in the AML with some other date that needs to be obtained from the database:

```
…
// The value is the date in the session time zone (according to the statement b)
// which is presented in neutral format (per statement a).
string created_on_str = this.getProperty( ‘created_on’);
// Convert this to a date-time object
DateTime created_on_dt = DateTime.Parse(created_on_str);
// Build request to the database and get another date with which we need to compare
// Note that interested us property value of type ‘date’ is also in the neutral
// format (according to statement a) and in the time zone of the session (statement b)
Item request = this.newItem( ‘Part’, ‘get’ );
request.setProperty( ‘item_number’, ‘xxx’ );
Item response = request.apply();
string part_date_str = response.getProperty( ‘modified_on’ );
DateTime part_dt = DateTime.Parse( part_date_str);
// Because both created_on_dt and part_dt are DateTime objects
// in the same time zone they could be compared
if( DateTime.Compare( created_on_dt, part_dt ) != 0 )
…
```

Let’s say the user writes a server method that must set property `closed_on` of type ‘ `date’` . The format of the property value must be in neutral format and the value must be in the time zone of the session in which the method is called. Here is a problem though: an attempt to create an instance of `DateTime` on the server would return a date\time object in the time zone of the server which might be different from the time zone of the current session; from the other side methods for getting `DateTime` in a particular time zone and\or converting from a time zone to another time zone are available only in .NET 3.5. In this case the best option is to use method `ConvertUtcDateTimeToNeutral(...)` available in `I18NSessionContext` class:

```
…
// First, get session context
I18NSessionContext cntx = this.getInnovator().getI18NSessionContext();
// Get ‘now’ moment as date-time in UTC in SortableDateTimePattern
// which is: “yyyy'-'MM'-'dd'T'HH':'mm':'ss”
String utc_now = DateTime.UtcNow.ToString( “s”);
// Get the same moment of time in the time zone of the session
// and in neutral format.
String pval = cntx.ConvertUtcDateTimeToNeutral( utc_now, null );
// Finally set the property value
this.setProperty( ‘closed_on’, pval );
…
```

Important

Like the above case in many cases it’s required to set ‘now’ moment of time as the value of a property. In this cases an alternative simpler approach would be to put __now() into the property closed_on. Note that can be done only in request AML as the __now()is interpreted by the Aras Innovator server and replaced on the equivalent date-time representation of ‘this moment of time’. So that can be done in let’s say a method that is called onBeforeUpdate as the resulting AML is processed by

the update action of server and the __now()is interpreted correctly. At the same time if the method is invoked on, let's say, onAfterUpdate putting __now() would be a wrong thing to do because this syntax is allowed only in AML requests (not responses (!) and onAfterUpdate event works on a response AML that is about to be sent back to the client) of add/update type (i.e. action="add" | "edit" | "merge" | "update" | etc.) – refer to section DateTime Properties for more details.

Let's say a user triggers a Server Event from a Client machine in the United States locale (en-US) but the server is located on a German locale with a German operating system (de-DE). This German server by default handles decimals in the format 123.456,78 whereas the US client handles decimals 123,456.78. Because of this mismatch in decimal handling on the client and server the server method must be able to handle an invariant culture for decimals. Here is some sample code written in C# to parse a string (returning a decimal property) and convert that string to Decimal:

string dval_str_enUS = "123,456.78"; // US Decimal String

string dval_str_deDE = "123.456,78"; // German Decimal String

```
//Convert US Decimal string to Decimal value
decimal dval_enUS = Decimal.Parse( dval_str_enUS, CultureInfo.InvariantCulture );
//Convert German Decimal string to Decimal Value
decimal dval_deDE = Decimal.Parse( dval_str_deDE, CultureInfo.InvariantCulture );
…//Perform some computations on decimal values…
this.setProperty("my_prop",cntx.ConvertToNeutral(dval_enUS.ToString(),"decimal","");
…
```

Date Related SQL Queries

AML is the language of Aras Innovator, but some external add-ons do connect to Aras Innovator on a direct SQL level. Because Aras Innovator stores DateTimes in SQL Server as UTC, we have provided an add-on function for Microsoft SQL Server that allows the easy conversion of DateTime to and from UTC.

To convert DateTime values from UTC to a specific time zone, use the function ConvertToLocal ({value},{Time Zone}). The time zone should be specified according to the registry key name of the time zone desired. (Refer to section Corporate Time Zone .) To use the DEFAULT parameter for Time Zone it is required that a value be set for CorporateTimeZone.

```
select item_number, created_on
from innovator.Document
```

Would be written as

```
select item_number, innovator.ConvertToLocal(created_on,DEFAULT) as CreadtedOn
from innovator.Document
```

Or

```
select item_number, innovator.ConvertToLocal(created_on,'Eastern Standard Time') as CreadtedOn
from innovator.Document
```

To convert DateTime values from a specific time zone to UTC, use the function `ConvertFromLocal({value},{Time Zone})` . The time zone should be specified according to the registry key name of the time zone desired. (Refer to section Corporate Time Zone) To use the DEFAULT parameter for Time Zone it is required that a value be set for CorporateTimeZone.

```
update innovator.Document
set effective_date = '1/1/2007 00:00:00.00'
Would be written as:
update innovator.Document
set effective_date = innovator.ConvertFromLocal('1/1/2007 00:00:00.00',DEFAULT)
```

Or

```
update innovator.Documnet
set effective_date = innovator.ConvertFromLocal('1/1/2007 00:00:00.00','Eastern Standard Time')
```

This allows users to write queries in a familiar time zone context.

```
select item_number, innovator.ConvertToLocal(release_date,DEFAULT) as ReleaseDate
from innovator.Document
where created_on >= innovator.ConvertFromLocal('1/1/2007 00:00:00.00',DEFAULT)
```

Important

For guidance on SQL date formatting in custom server-side code, see Appendix A: Custom Server-Side Code and SQL Date Formatting in the Upgrading to 40 from Aras Innovator 14+ guide.