|  |  |  |
| --- | --- | --- |
| Subject: |  | ctDynamics™ SL API Defaults And Validation |
| Date: |  | 07/27/2012 |

# Overview

Catalina Technology's ctDynamicsSL™ web services for Dynamics SL provides a configurable schema for defaults and validation.

Contents

[Overview 1](#_Toc331402535)

[Table of Figures 1](#_Toc331402536)

[Table of Tables 1](#_Toc331402537)

[Programming Guide](#_Toc331402538) 1

[Creating Custom Data Defaults](#_Toc331402544) 2

[Setting the location of your custom default configuration files](#_Toc331402545) 2

[Custom default configuration files 2](#_Toc331402546)

[Enabling custom defaults in your web service 2](#_Toc331402547)

[Custom defaults configuration file XML Structure 3](#_Toc331402548)

[Custom Data Validation 5](#_Toc331402549)

[Setting the location of your custom validation configuration files 5](#_Toc331402550)

[Custom validation configuration files 5](#_Toc331402551)

[Custom validations configuration file XML Structure 5](#_Toc331402552)

## Creating Custom Data Defaults

When using the ctDynamicsSL™ web services, default values for fields are often needed to be configured based on the application needs. These defaults will be used if the programmer does not pass values in those fields.

### Setting the location of your custom default configuration files

Custom data defaults are stored in configuration files that are located a folder defined in the DSLCONFIGFILE.XML file under the key. These files contain the definitions for any of the classes you want to define defaults for.

|  |
| --- |
| <CONFIGITEM ID='DEFAULTCONFIGDIRECTORY'>  c:\inetpub\xctFiles\config\  </CONFIGITEM> |

Figure 6: DSLCONFIG.XML location for custom default files

*You would replace the value in this key with the location of your implementation.*

### Custom default configuration files

When you create a custom default configuration file, you would store it in the configuration directory defined in the DSLCONFIGFILE.XML key DEFAULTCONFIGDIRECTORY (see Setting the location of your custom default configuration files).

Each class you need to define defaults for, are stored in separate configuration files. These files have a naming convention of: ‘default.’ + namespace.class + ‘.xml’

Example:

If you are using the namespace.class of *ctDynamicsSL.fieldService.serviceContracts.maintenance.serviceContractEntry*

You would create a filename of:

*default.ctDynamicsSL.fieldService.serviceContracts.maintenance.serviceContractEntry.xml*

### Enabling custom defaults in your web service

You can turn on/off the custom defaults for your web service by setting the key ENABLEDEFAULTS in your DSLCONFIG.XML file to TRUE or FALSE (TRUE is on, FALSE is off).

Example:

If you want to turn custom defaults to be turned on, you would set the key to look like this:

<CONFIGITEM ID='ENABLEDEFAULTS'>TRUE</CONFIGITEM>

NOTE: if this field is not defined or included in your DSLCONFIG.XML file, no defaulting will occur.

### Custom defaults configuration file XML Structure

Custom defaults are configured in an XML file that defines which functions will have defaulted values, and the default value of parameters in that function. Only parameters defined in the configuration file will have defaults applied.

|  |
| --- |
| <?xml version=‘1.0’ encoding=‘utf-8’ ?>  <FUNCTIONS>  <FUNCTION ID='functionName'>  <VARIABLES>  <VARIABLE ID='variableName' DEFAULTTYPE=''/>  </VARIABLES>  </FUNCTION>  </FUNCTIONS> |

Figure 7: Custom defaults configuration file

#### Properties of the configuration file

The properties of the defaults configuration XML file are listed below.

Table 1: Defaults configuration XML file properties

|  |  |
| --- | --- |
| **Element** | **Description** |
| ID | Required   * The name of the parameter to the function * Case Sensitive * supports inner public properties of parameter objects * ex: myObject.myValue |
| DEFAULTTYPE | Required   * Sets the default value * Supports: TEXT, PROC (see Table 2: Custom Default Configuration Default Types supported for more information of the different types) |
| Inner Tag value | Required   * The value to default * <variable>VALUEHERE</variable> * If DEFAULTTYPE is PROC, then the name of the procedure to call |
| DEFAULT | Optional   * Param level validation setting * Supports: true/false |
| PARMS | Optional   * IF DEFAULTTYPE is PROC, this overrides the default parameter passed to the procedure with the list of variables listed here, comma delimited. Note: if set, then ID is not sent by default, must be specified in list. |
| ENCRYPTED | Optional   * values may be encrypted using Catalina's ctConfigEditor tool * Supports: true/false * siteKey encrypted value is supported |

Below are the different DEFAULTTYPE values that can be used in the custom defaults configuration file

Table 2: Custom Default Configuration Default Types supported

|  |  |
| --- | --- |
| **Element** | **Description** |
| TEXT | * Sets the variable to the value in this field. * <VARIABLE ID='inItem.Status' DEFAULTTYPE='TEXT' PARMS='' ENCRYPTED=‘False’>A</VARIABLE> |
| PROC | * Calls procedure specified in xml using just the id as the param, or all params listed in the PARMS tag. * <VARIABLE ID='inItem.StdCost' DEFAULTTYPE='PROC' PARMS='inItem.InvtId' ENCRYPTED='False'>xct\_spDSLdefaultInventory</VARIABLE> |

Example of a custom defaults definition file:

|  |
| --- |
| TODO: Put an example of a custom Data Defaults configuration file  <?xml version=‘1.0’ encoding=‘utf-8’ ?>  <FUNCTIONS>  <FUNCTION ID='functionName'>  <VARIABLES>  <VARIABLE ID='variableName' DEFAULTTYPE=''/>  </VARIABLES>  </FUNCTION>  </FUNCTIONS> |

Figure 8: Example of a custom defaults definition file.

## Custom Data Validation

Custom Data Validation allows the programmer to define additional validations for data passed to the web services.

### Setting the location of your custom validation configuration files

Custom data validations are stored in configuration files that are located a folder defined in the DSLCONFIGFILE.XML file under the key. These files contain the definitions for any of the classes you want to define defaults for.

|  |
| --- |
| <CONFIGITEM ID='VALIDATIONCONFIGDIRECTORY'>  c:\inetpub\xctFiles\config\  </CONFIGITEM> |

Figure 9: DSLCONFIG.XML location for custom validation files

*You would replace the value in this key with the location of your implementation.*

### Custom validation configuration files

When you create a validation default configuration file, you would store it in the configuration directory defined in the DSLCONFIGFILE.XML key VALIDATIONCONFIGDIRECTORY (See Setting the location of your custom validation configuration files).

Each class you need to define validations for, are stored in separate configuration files. These files have a naming convention of: ‘validate.’ + namespace.class + ‘.xml’

Example:

If you are using the namespace.class of *ctDynamicsSL.fieldService.serviceContracts.maintenance.serviceContractEntry*

You would create a filename of:

*validate.ctDynamicsSL.fieldService.serviceContracts.maintenance.serviceContractEntry.xml*

NOTE: if this field is not defined or included in your DSLCONFIG.XML file, no defaulting will occur.

### Custom validations configuration file XML Structure

Custom validations are configured in an XML file that defines which functions will have defaulted values, and the default value of parameters in that function. Only parameters defined in the configuration file will have defaults applied.

|  |
| --- |
| <?xml version=‘1.0’ encoding=‘utf-8’ ?>  <FUNCTIONS>  <FUNCTION ID='functionName'>  <VARIABLES>  <VARIABLE ID='variableName' DEFAULTTYPE=''/>  </VARIABLES>  </FUNCTION>  </FUNCTIONS> |

Figure 10: Custom validations configuration file

#### Properties of the configuration file

The properties of the validation configuration XML file are listed below.

Table 1: Validation configuration XML file properties

|  |  |
| --- | --- |
| **Element** | **Description** |
| ID | Required   * The name of the parameter to the function * Case Sensitive * supports inner public properties of parameter objects * ex: myObject.myValue |
| VALIDATETYPE | Required   * The type of validation we will perform on the parameter. * Supports: LIST, NUMBERRANGE, DATERANGE, PROC (See Table 2: Custom Validation Configuration Default Types supported) |
| Inner Tag value | Required   * <variable>VALUEHERE</variable> * The values to compare against. Comma delimited where necessary * IF VALIDATETYPE is PROC, then the name of the procedure to call. |
| ERRORMESSAGE | Optional   * override the system error message with content here |
| VALIDATE | Optional   * true/false * Param level validation setting. |
| PARMS | Optional   * IF VALIDATETYPE is PROC, this overrides the default parameter passed to the procedure with the list of variables listed here, comma delimited. Note: if set, then ID is not sent by default, must be |

Below are the different VALIDATETYPE values that can be used in the custom validation configuration file

Table 2: Custom Validation Configuration Default Types supported

|  |  |
| --- | --- |
| **Element** | **Description** |
| LIST | * compares against a comma delimited list of values. * Case-insensitive * <VARIABLE ID='actionType' VALIDATETYPE='LIST'>VALIDATEONLY,ADD,UPDATE,DELETE</VARIABLE> |
| DATERANGE | * compares against a date range using comma delimited minDate, maxDate * <VARIABLE ID='beginDate' VALIDATETYPE='DATERANGE'>1/1/2000,1/1/2014</VARIABLE> |
| NUMBERRANGE | * compares against a range using comma delimited: minValue, maxValue * <VARIABLE ID='priority' VALIDATETYPE='NUMBERRANGE'>1,10</VARIABLE> |
| PROC | * Calls procedure specified in xml using just the id as the param, or all params listed in the PARMS tag. * <VARIABLE ID='inItem.CpnyID' VALIDATETYPE='PROC' PARMS='inItem.CpnyID' ERRORMESSAGE='' |