Cisco Prime Service Catalog 12.1 Patch – Material Form Server ISF Library Method Signatures

First Published: December 26, 2018

This document contains information on the method signatures for the material Form Server ISF Library introduced in 12.1_Patch_v7.

Method Signatures and Comments

```
/**
 * Initialises _event_to_rules_map and _type_of_method_array
 * _event_to_rules_map: map between form/field specific event and js functions (form rules implementation) to execute for the event
 * _type_of_method_array: type for js functions (form rules implementation). Types : CR or DDR
   @param
   @return
*/
_isf_props = {}

/**
* To get _event_to_rules_map
* _event_to_rules_map: map between form/field specific event and js functions (form rules implementation) to execute for the event
   @param
   @return object
*/
```

```
isf_eventMap = () => {}


/**
* To get _type_of_method_array
* _type_of_method_array: type for js functions (form rules implementation). Types : CR or DDR
   @param
   @return object
*/
isf_methodArray = () => {}


/**
 * Object which contains namespace specific variables
*/
_isf_variables = {};


/**
   To resolve requisiton and service definition related namespace variables in a string
@param form: object - service form object which contains all form specific information
@param str: string - String to sanitize
@param separator: string - separator to identify namespace variables.
   Note: PSC specific separator is "#"
@return string
*/
isf_resolve_req_namespace = (form, str, separator)
```

/**

To resolve service form specific namespace variables in a string

@param form: object - service form object which contains all form specific information

@param str: string - String to sanitize

@param separator: string - separator to identify namespace variables.

@param nullIsZero: boolean - if null or blank found in string should be replace with 0

@param noCommas: boolean - if commas are to to removed from result string. Will return substring to first comma if set to true

@param isForDDR: boolean - called for ddr or not

@return string

*/

isf_cr_namespace = (form, str, separator, nullIsZero, noCommas, isForDDR)

/**

Figure out whether the distribution targets are of a valid combination or not.

1. Combination of the same grid for multiple times - Valid

2. Combination of more than one Flat - Valid

3. Combination of more than one grid - Invalid

4. Grid + Flat - Invalid

@param form: object - service form object which contains all form specific information

@param listOfTargets - list of field keys. A key is of format dictionaryName.fieldName

@return boolean

*/

isf_isDDRTargetCombinationValid = (form, listOfTargets)

/**

To check if dictionary is grid dictionary

@param form: object - service form object which contains all form specific information

@param dictionaryName: string

@return boolean

*/

isf_isGridDictionary = (form, dictionaryName)

/**

Escapes double quoatation marks in the string

@return string

*/

isf_fromEscString(escstr)

/**

DDR form rule handler

To execute a Data Retreiveval Rule (DDR)

@param form: object - service form object which contains all form specific information

@param dictionaryParameters: array - objects contains dictionary field values ddr is dependent on. To be passed in API request body for ddr execute rule api

@param encodedRuleId: string - url encoded form rule id (id is already encrypted) as in encodeURICompo-nent("${rule.ecryptedId}")

@return

*/

isf_ddr_call = (form, dictionaryParameters, encodedRuleId)

/**

To make changes to service form after DDR is executed

```
@param form: object - service form object which contains all form specific information

@param response: object - response of ddr executeRule API

@return

*/

isf_ddr_callback_success = (form, response)


/**

   To extract values from DDR response in comma separated format

@return string: commas separated field values

*/

getValueStringFromList = (listOfValues)


/**

   To get array of options to be used to change options in single/multi select type fields. Used in DDR success
callback

@param values: array : values to be used to make new options

@return array

*/

getOptions = (values)


/**

   To handle ddr failure scenario

@param form: object - service form object which contains all form specific information

@param error: object - error response object from DDR execute API api

@return

*/

isf_ddr_callback_fail = (form, error)
```

```
/**
   To check if dictionary exists on service form
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@return boolean
*/
isf_checkDictionaryExistence = (form, fieldKey)

/**
   To check is field is grid field or not
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@return
*/
isf_isGridField = (form, fieldKey)

/*
   To get value from field. It returns a single value even in case of multi valued fields.
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@return string
*/
isf_getRealSafeValue = (form, fieldKey)
```

```
/**
   To get value from field. Returns array of values.
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@return string/array
*/
isf_getValue = (form, fieldKey)


/**
   To get value from field. Returns formatted values. Field specific format is used.
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@return string
*/
isf_getSafeValue = (form, fieldKey)


/**
 *  CR rule handler for set value action
 *  To set the value of the field
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@param value: string
@param valueIsField: boolean - value is a dictionary field or not
@param avs object: object : values to be used to set. check formrules_javascript_tpl.ftl
@return
*/
```

```
isf_cr_setValue = (form, fieldKey, value = "", valueIsField, avs = {})


/**
   To check if dictionary is a person based dictionary
@param form: object - service form object which contains all form specific information
@param dictionaryName: string
@return boolean
*/
isf_isDictionaryPersonBased = (form, dictionaryName)


/**
   To check if dictionary field is person based
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@return boolean
*/
isf_isFieldPersonBased = (form, fieldKey)


/**
   To check if dictionary is service item based
@param form: object - service form object which contains all form specific information
@return boolean
*/
isf_isDictionaryServiceItemBased = (form, dictionaryName)
```

```
/**
    To check if dictionary field is service item based
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@return boolean
*/
isf_isFieldServiceItemBased = (form, fieldKey)


/**
    To set field value in case or text type, person or service item identifier. Person (in person based dictionary) and service item identifier type fields should populate other dictionary fields
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@param id: string - id of person in case of person type. For other types it is equal to SERVICE_FORM_CONSTANTS.FORM_ID
@param value: string - field should be set to this value
@param isForDDR: boolean
@return
*/
setFieldValueWithAutoPopulate = (form, fieldKey, id, value, isForDDR)


/*
    To set value of a field
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@param sourceValue: string
@return
```

```
*/
setTargetFieldValue = (form, fieldKey, sourceValue)

/**
 * CR rule handler for set price action and set value action (when set value from expression result)
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@param priceAppliesTo: string - serivce names from the rules to which the price applies to
@param parameters: string - value
@param parametersIsField: boolean - value of parameter is a dictionary field name or not
@return
*/
isf_cr_setPrice = (form, fieldKey, priceAppliesTo, parameters, parametersIsField)

/**
   To set price of service form. Also invoked to set value of a field using expression result in a form rule.
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@param priceAppliesTo: string - serivce names from the rules to which the price applies to
@param parameters: string - value
@param parametersIsField: boolean - value of parameter is a dictionary field name or not
@return
*/
setPriceTask = (form, fieldKey, priceAppliesTo, parameters, parametersIsField)
```

```
/**
 * To set price for service form
@param response: object - reponse of evalute Expression API
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@param priceAppliesTo: string - serivce names from the rules to which the price applies to
@return
*/
setPriceHandler = (response, form, fieldKey, priceAppliesTo)

/**
 * To set price of service form or set value for set value action from expression
@param pricePerUnit: string - price per unit from CR action
@param priceAppliesTo: string - serivce names from the rules to which the price applies to
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@param form: object - service form object which contains all form specific information
@return
*/
setPriceField = (pricePerUnit, priceAppliesTo, fieldKey, form)

/**
 * To check is=f key is a dictionary name (contained in service form)
@param key: string
@return boolean
*/
isf_isDictionaryKey = (key)
```

```
/**
 * CR handler for action set mandatory
@param form: object - service form object which contains all form specific information
@param key: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@return
*/
isf_cr_setMandatory = (form, key, onoff = true)

/**
 * To set dictionary as mandatory. isMandatory attribute of all fields in this dictionary will set to onoff
@param form: object - service form object which contains all form specific information
@param dictionaryKey: string
@param onoff: boolean
@return
*/
isf_cr_dictionarySetMandatory = (form, dictionaryKey, onoff)

/**
 * To set dictionary field as mandatory. isMandatory attribute of field will set to onoff
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@param onoff: boolean
@return
*/
```

```
isf_cr_fieldSetMandatory = (form, fieldKey, onoff)


/**
 * CR handler for action set disabled
@param form: object - service form object which contains all form specific information
@param key: string
@param onoff: boolean
@return
*/
isf_cr_setDisabled = (form, key, onoff = true)


/**
 * To set dictionary as disabled. isEditable attribute of all fields in this dictionary will set to onoff
@param form: object - service form object which contains all form specific information
@param dictionaryKey: string
@param onoff: boolean
@return
*/
isf_cr_dictionarySetDisabled = (form, dictionaryKey, onoff)


/**
 * To set dictionary field as disabled. isEditable attribute of field will set to onoff
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@param onoff: boolean
@return
```

```
*/
isf_cr_fieldSetDisabled = (form, fieldKey, onoff)

/**
 * CR handler for action set hidden
@param form: object - service form object which contains all form specific information
@param onoff: boolean
@return
*/
isf_cr_setHidden = (form, key, onoff = true)

/**
 * To set dictionary field as hidden. isHidden attribute of all fields in this dictionary will set to onoff
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@param onoff: boolean
*/
isf_cr_dictionarySetHidden = (form, dictionaryKey, onoff)

/**
 * To set dictionary field as hidden. isHidden attribute field will set to onoff
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@param onoff: boolean
@return
```

```
*/
isf_cr_fieldSetHidden = (form, fieldKey, onoff)

/**
 * CR handler for action stop submission
@param form: object - service form object which contains all form specific information
@return
*/
isf_cr_stopSubmission = (form)

/**
 * CR handler for action show alert
@param form: object - service form object which contains all form specific information
@return
*/
isf_cr_alert = (form, msg)

/**
 * CR handler for action set focus
@param form: object - service form object which contains all form specific information
@return
*/
isf_cr_setFocus = (form, key)

/**
 * To check if field exists on the service form
```

@param form: object - service form object which contains all form specific information

@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.

@return

*/

isf_checkFieldExistence = (form, fieldKey)

/**

* CR handler for conditions where operator is 'exists on service form'

@param form: object - service form object which contains all form specific information

@param leftOperand: string - operand from the condition

@param leftOperandIsField: boolean

@return boolean

*/

isf_cr_exists = (form, leftOperand, leftOperandIsField)

/**

* To check if all the fields of a dictionary are disabled. Checks isEditable attribute for the field.

@param form: object - service form object which contains all form specific information

@param dictionaryKey: string

@return boolean

*/

isf_cr_isDictionaryDisabled = (form, dictionaryKey)

/**

* To check if all the fields of a dictionary are readonly. Checks controlType attribute for the field.

```
@param form: object - service form object which contains all form specific information

@param dictionaryKey: string

@return boolean

*/

isf_cr_isDictionaryReadOnly = (form, dictionaryKey)


/**

 * CR handler for condition where operator is 'is readonly'.

@param form: object - service form object which contains all form specific information

@param leftOperand: string - operand from the condition

@param leftOperandIsField: boolean

@return boolean

*/

isf_cr_isDisabled = (form, leftOperand, leftOperandIsField)


/**

 * CR handler for condition where operator is 'is read write'

@param form: object - service form object which contains all form specific information

@param leftOperand: string - operand from the condition

@param leftOperandIsField: boolean

@return

*/

isf_cr_not_isDisabled = (form, leftOperand, leftOperandIsField)


/**

 * To check if all fields of the dictioanry are hidden. Checks isHidden attribute of the field
```

@param form: object - service form object which contains all form specific information

@param dictionaryKey: string

@return boolean

*/

isf_cr_isDictionaryHidden = (form, dictionaryKey)

/**

* CR handler for condition where operator is 'is hidden'

@param form: object - service form object which contains all form specific information

@param leftOperand: string - operand from the condition

@param leftOperandIsField: boolean

@return boolean

*/

isf_cr_isHidden = (form, leftOperand, leftOperandIsField)

/**

* CR handler for condition where operator is 'is visible'

@param form: object - service form object which contains all form specific information

@param leftOperand: string - operand from the condition

@param leftOperandIsField: boolean

@return boolean

*/

isf_cr_not_isHidden = (form, leftOperand, leftOperandIsField)

/**

```
 * CR handler for condition is Order on behalf

@param form: object - service form object which contains all form specific information

@return boolean

*/

isf_isOOB=(form)


/**

 * CR handler for condition is Not Order on behalf

@param form: object - service form object which contains all form specific information

@return

*/

isf_isNotOOB=(form)


/**

 * CR handler for condition has unsubmitted requisitions

@param form: object - service form object which contains all form specific information

@return

*/

isf_hasUnsubmittedRequisitions=(form)


/**

 * CR handler for condition does not have unsubmitted requisitions

@param form: object - service form object which contains all form specific information

@return

*/

isf_doesNotHaveUnsubmittedRequisitions=(form)
```

```
/**
 * CR handler for condition customer role
@param form: object - service form object which contains all form specific information
@param roleId: string - a number
@return
*/
isf_isCustomerMemberOfRole = (form, roleId)



/**
 * CR handler for condition user role
@param form: object - service form object which contains all form specific information
@param roleId: string - a number
@return
*/
isf_isUserMemberOfRole = (form, roleId)


/**
 * To get value of a variable with name as leftOperand
@param form: object - service form object which contains all form specific information
@param leftOperand: string - operand from the condition
@return
*/
getServiceFormVariableValue = (form, leftOperand)
```

```
/**
 * compares values extracted using left operand and right operand.
@param form: object - service form object which contains all form specific information
@param leftOperand object : operand in the condition
@param leftOperandIsField
@param rightOperandIsField
@param rightOperand object: operand in the condition
@param literalComparer
@param arrayComparer
@return boolean
*/
isf_compareOperands = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand, literalComparer, arrayComparer)

/**
 * invokes comparer on left operand and right operand.
 *  > Literal comparer invoked if none is array
 *  > array comparer invoked if at least one is array
@param leftOperand object : operand in the condition
@param rightOperand object: operand in the condition
@param literalComparer
@param arrayComparer
@return boolean
*/
isf_cr_compare_invoker = (leftOperand, rightOperand, literalComparer, arrayComparer)
```

```
/**
 * To get comparable value from field with fieldKey leftOperand
@param form: object - service form object which contains all form specific information
@param fieldKey: string: key to identify a field on service form. It has value as dictionary_name.field_name.
@param leftOperandIsField
@param leftOperand object : operand in the condition
@return string
*/
isf_getComparableValueFromField = (form, fieldKey, leftOperandIsField, leftOperand)

/**
 * To get comparable value ltrl
 *  > resolve namespace variables in ltrl
 *  > ltrl formatted to type of dictionary field pointed to by leftOperand
@param form: object - service form object which contains all form specific information
@param ltrl: value
@param leftOperandIsField
@param leftOperand object : operand in the condition
@return string
*/
isf_getComparableValueFromLiteral = (form, ltrl, leftOperandIsField, leftOperand)

/**
 * To get comparable value from ltrl depending on type of field with name as leftOperand
@param form: object - service form object which contains all form specific information
```

```
@param ltrl: value

@param leftOperandIsField

@param leftOperand object : operand in the condition

@return string

*/

isf_getComparableValue = (form, ltrl, leftOperandIsField, leftOperand)

/**

 * To check is val matches number regular expression

@param val: strings

@return boolean

*/

isf_isNumberValue = (val)


/**

 * To check if val matches money regular expression

@param val: string

@return boolean

*/

isf_isMoneyValue = (val)


/**

 * To check if date is valid date

@param date: Date

@return

*/

isf_isValidDate = (date)
```

```
/**
 * CR handler for condition which has operator equals
 * Values extracted from leftOperand and rightOperand are compared. Compared on equals here.
@param form: object - service form object which contains all form specific information
@param leftOperand: value
@param leftOperandIsField
@param leftOperandIsField
@param rightOperand
@return boolean
*/
isf_cr_equals = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/**
 * To check if arrays a and b are equal
@param a: array
@param b: array
@return
*/
isf_equals_array_array = (a, b)

/**
 * CR handler for condition which has oprator equals ignore
 * Values extracted from leftOperand and rightOperand are compared. Compared on equals ignore case here.
@param form: object - service form object which contains all form specific information
```

```
@param leftOperand: value

@param leftOperandIsField

@param leftOperandIsField

@param rightOperand

@return boolean

*/

isf_cr_equals_nocase = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)


/**

 * To check if arrays a and b are equal ignore case

@param a: array

@param b: array

@return

*/

isf_equal_nocase_array_array = (a, b)



/**

 * CR handler for condition which has oprator not equals

 * Values extracted from leftOperand and rightOperand are compared. Compared on not equals here.

@param form: object - service form object which contains all form specific information

@param leftOperand: value

@param leftOperandIsField

@param leftOperandIsField

@param rightOperand

@return boolean
```

```
*/
isf_cr_not_equals = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/**
 * CR handler for condition which has oprator starts with
 * Values extracted from leftOperand and rightOperand are compared. Compared on starts with here.
@param form: object - service form object which contains all form specific information
@param leftOperand: value
@param leftOperandIsField
@param leftOperandIsField
@param rightOperand
@return boolean
*/
isf_cr_starts_with = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/**
 * To check if a[0] starts with at least one value in array b
@param a: array
@param b: array
@return
*/
isf_cr_starts_with_array_array = (a, b)

/**
 * CR handler for condition which has oprator ends with
```

* Values extracted from leftOperand and rightOperand are compared. Compared on ends with here.

@param form: object - service form object which contains all form specific information

@param leftOperand: value

@param leftOperandIsField

@param leftOperandIsField

@param rightOperand

@return boolean

*/

isf_cr_ends_with = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/**

* To check if a[0] ends with at least one value in array b

@param a: array

@param b: array

@return boolean

*/

isf_cr_ends_with_array_array = (a, b)

/**

* CR handler for condition which has oprator contains

* Values extracted from leftOperand and rightOperand are compared. Compared on contains here.

@param form: object - service form object which contains all form specific information

@param leftOperand: value

@param leftOperandIsField

@param leftOperandIsField

@param rightOperand

@return boolean

*/

isf_cr_contains = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/**

* To check if values is b are contained in at least one value in a

@param a: array

@param b: array

@return boolean

*/

isf_contains_array_array = (a, b)

/**

* CR handler for condition which has oprator does not contain

* Values extracted from leftOperand and rightOperand are compared. Compared on does not contain here.

@param form: object - service form object which contains all form specific information

@param leftOperand: value

@param leftOperandIsField

@param leftOperandIsField

@param rightOperand

@return boolean

*/

isf_cr_not_contains = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/**

```
 * CR handler for condition which has oprator greater than

 * Values extracted from leftOperand and rightOperand are compared. Compared on greater than here.

@param form: object - service form object which contains all form specific information

@param leftOperand: value

@param leftOperandIsField

@param leftOperandIsField

@param rightOperand

@return boolean

*/

isf_cr_gt = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)


/**

 * returns result of a > b

@param a: array

@param b: array

@return boolean

*/

isf_cr_gt_array_array = (a, b)


/**

 * CR handler for condition which has oprator greater than equal to

 * Values extracted from leftOperand and rightOperand are compared. Compared on greater than equal to here.

@param form: object - service form object which contains all form specific information

@param leftOperand: value

@param leftOperandIsField

@param leftOperandIsField
```

@param rightOperand

@return boolean

*/

isf_cr_gte = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/**

* returns result of a >= b

@param a: array

@param b: array

@return boolean

*/

isf_cr_gte_array_array = (a, b)

/**

* CR handler for condition which has oprator less than

* Values extracted from leftOperand and rightOperand are compared. Compared on less than here.

@param form: object - service form object which contains all form specific information

@param leftOperand: value

@param leftOperandIsField

@param leftOperandIsField

@param rightOperand

@return boolean

*/

isf_cr_lt = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

```
/**
 * returns result of a < b
@param a: array
@param b: array
@return boolean
*/
isf_cr_lt_array_array = (a, b)

/**
 * CR handler for condition which has oprator less than equal to
 * Values extracted from leftOperand and rightOperand are compared. Compared on less than equal to here.
@param form: object - service form object which contains all form specific information
@param leftOperand: value
@param leftOperandIsField
@param leftOperandIsField
@param rightOperand
@return boolean
*/
isf_cr_lte = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/**
 * returns result of a <= b
@param a: array
@param b: array
@return boolean
*/
```

```
isf_cr_lte_array_array = (a, b) => {

    return a <= b;// we want avoid strict comparision here
```