PWB Server API Programmers Guide

PDM Workbench Release 20.0 for Aras Innovator

User Manual

Version 1

T Systems

Copyright

Contact

T-Systems International GmbH
Business Unit PLM
Fasanenweg 5
70771 Leinfelden-Echterdingen
Germany

https://plm.t-systems.net/en-DE/pdm-workbench

☎ +49 (0) 40 30600 5544
✉ +49 (0) 3915 80125688

mail : *cmi_support@t-systems.com*

Manual History

Version	Date
19.0	Mar 2025
20.0	Dec 2025

This edition 20.0 of the manual obsoletes all previous editions.

Your Comments are Welcome

Please feel free to tell us your opinion; we are always interested in improving our publications. Mail your comments to:

T-Systems International GmbH
Business Unit PLM
Fasanenweg 5
70771 Leinfelden-Echterdingen
Germany

mail: *cmi_support@t-systems.com*

Preface

About this Manual

The PWB server API extends the Aras Innovator API. It provides additional methods used in context of the PWB CATIA and NX application.

This manual describes the application programming interface for the PWB server addin and provides basic information about how to extend the PWB CATA and NX application with own server methods.

Table of Contents

CHAPTER 1

PWB Server API

This chapter describes the PWB Server Application Programming Interface (API), which extends the Aras Innovator API.

The API is provided for the programming language C#.

Namespace PwbServerAddin

public class PwbServerAPI

public class **PwbServerApi**
Summary:
Main class for PWB server API functionality.

Methods:

Constructor:

public PwbServerApi **PwbServerApi** (
Aras.IOM.Item CalllingMethodItem,
string ServerConfigName = null,
string CadClient = null)
Summary:
Constructor for PwbServerApi class.
Parameters:
CalllingMethodItem: Item representing method from which method is called.
ServerConfigName: Server configuration to be used.
CadClient: Permitted values are CadClientCatia, CadClientNx, CadClientCreo

Aras Utilities:

public string **GetValueOfVariable**(
string Name)
Summary:
Get the value of Aras Server variable by name.
Parameters:
Name: Name of variable to search for.
Returns:
Value of variable

public bool **CurrentUserHasIdentity**(
string IdentityName)
Summary:
Checks whether the current user belongs to identity.
Parameters:
IdentityName: Name of variable to search for.
Returns:
Whether the user belongs to identity

public int **GetRelationshipCount**(
Aras.IOM.Item ParentItem
Aras.IOM.Item RelationshipType)
Summary:
Method to get the count of relationships. (Used for Quantity changes)
Parameters:
ParentItem: The source item of the relationship (Case Quantity: Use a Part BOM Item here)
RelationsshipType: The name of the relationship (Case Quantity: Use the name of the Part BOM to BOM Instance Relationship)

public string **GetThumbnailFieId**(
Aras.IOM.Item ItemObj)
Summary:
Gets the ID of the attached thumbnail file.
Parameters:
ItemObj: The item to get the link from.
Returns:
ID from related thumbnail file

public string **GetNextAutonameSequence**(
string SequenceName)
Summary:
Generates the next available sequence number for autonaming functionality.
Parameters:
SequenceName: The name of the sequence.
Returns:
Generated sequence number

public void **CheckItem**(
Aras.IOM.Item ItemObj)
Summary:
Check Aras.IOM.Item for errors and throws exception if an error was found.
Parameters:
ItemObj: The Aras.IOM.Item to analyse.

public void **CheckItem**(
Aras.IOM.Item ItemObjToCheck,
string ErrorStringIfFailed)
Summary:
Check Aras.IOM.Item for errors.
Parameters:
ItemObjToCheck: The Aras.IOM.Item to analyse.
ErrorStringIfFailed: Returns error string

public void **GetSpecificCadFilesInfo**(
string ItemId,
out List<IDictionary<string, string>> RelatedSpecificCadFiles)

Summary:
Gets infos about related specific cad files.
Parameters:
ItemId: ID from CAD Document to search related specific documents for.
RelatedSpecificCadFiles: List with related specific cad files

public void **GetRelatedItems**(
PwbApi.Obj StartObj,
string RelationType,
string RelationClassification,
bool IsLeftToRight,
out List<*PwbApi.Rel*> RelatedItems,
string InstanceRelationType = "")

Summary:
Get parent or child relations according to given relation type and classification.
Parameters:
StartObj: Obj to search relations for.
RelationType: Relation type
RelationClassification: Relation classification
IsLeftToRight: Whether to search for child or parent relations (true : parents, false: children)
RelatedItems: List of relations fitting the given input criteria
InstanceRelationType: Type of instance relations

public void **GetRelatedItems**(
string StartItemType,
string StartItemClassification,
string StartItemId,
string RelationType,
string RelationClassification,
bool IsLeftToRight,
out List<*PwbApi.Rel*> RelatedItems,
string InstanceRelationType = "")

Summary:
Get parent or child relations according to given relation type and classification.
Parameters:
StartItemType: Type of Obj to search relations for.
StartItemClassification: Classifacation of Obj to search relations for.
StartItemId: ID of Obj to search relations for.
RelationType: Relation type
RelationClassification: Relation classification
IsLeftToRight: Whether to search for child or parent relations (true : parents, false: children)
RelatedItems: List of relations fitting the given input criteria
InstanceRelationType: Type of instance relations

public void **SyncToBomApi**(
string *Type*,
string *Id*,
out System.Collections.Generic.List<string[]> *Messages*)
Summary:
Synchronizes a CAD Structure to BOM, that is, updates a structure of 'Part BOM' and 'BOM Instance' relation from a 'CAD Structure' and 'CAD Instance' structure as the input.
Parameters:
Type: Type of selected object. Currently only 'CAD' is supported.
Id: Aras id of selected object.
Messages: User messages: List of [MessageText][MessageType].

public bool **IsUserStandardPartAdmin**()
Summary:
Determines whether the user is member of the identity Standard Part Administrator.
Returns:
Whether user is member of the identity Standard Part Administrator.

public bool **ItemIsStandardPart**(Aras.IOM.Item ItemObj)
Summary:
Checks whether an item represents a standard part.
Parameters:
ItemObj: The item to check.
Returns:
Whether the item ItemObj represents a standard part.

public bool **IsUserTemplateFileAdmin**()
Summary:
Determines whether the user is member of the identity Template File Administrator.
Returns:
Whether user is member of the identity Template File Administrator.

public bool **ItemIsTemplateFile**(Aras.IOM.Item ItemObj)
Summary:
Checks whether an item represents a template file.
Parameters:
ItemObj: The item to check.
Returns:
Whether the item ItemObj represents a template file.

public bool **ItemIsSuperseded**(Aras.IOM.Item ItemObj)
Summary:
Checks whether an item is superseded, a newer version from the item exists.
Parameters:
ItemObj: The item to check.
Returns:
Whether the item ItemObj is superseded.

public void **CheckForEnvironmentAttributes**(
Aras.IOM.Item ItemObj,
ref bool IsEditAllowed,
ref bool IsCheckOutAllowed,
ref bool CheckOutAsNewGeneration)
Summary:
Determines behaviour of an item object.
Parameters:
ItemObj: The item to check.
IsEditAllowed: Whether edit is allowed for ItemObj
IsCheckOutAllowed: Whether check out is allowed for ItemObj
CheckOutAsNewGeneration: not implemented yet

public Aras.IOM.Item **UpdateAttrs**(
Dictionary<string, string> Attrs,
Aras.IOM.Item ItemToUpdate)
Summary:
Update attributes for a given item
Parameters:
Attrs: Dictionary with the new attribute values
ItemToUpdate: Item to update the attributes for.

```
public string GetFilteredStructureAml(
        string StartItemId,
        string StartItemType,
        string RelationType,
        string RelationClassification,
        string ConfigurationName,
        IDictionary<string, string> SelectAttrs,
        string Resolution,
        string loadWithLinks,
        Dictionary <string, Dictionary<string, string>> VariantConfigValues)
```

Summary:
Creates an AML string for an assembly structure or component. It can be used with an Structure Configuration for both CAD Structure an BOM Part mode.
Parameters:

StartItemId:	Item ID of the Item for which the AML String should be created
StartItemType:	Item Type of the Item for which the AML String should be created
RelationType:	class of relation with instance transformation matrix
RelationClassification:	Relation Type (CAD Structure / Part BOM).
ConfiguationName:	possible effectivity / Product Variant for Structure Config
SelectAttrs:	Attributes to select
Resolution:	Expand Info (Current, AsSaved, Released)
loadWithLinks:	if links should be loaded
VariantConfigValues:	possible values for the Structure Config which are used instead of a product variant / effectivity.

If both VariantConfigValues and ConfigurationName is used, the VariantConfigValues will have higher priority and will be used for the Structure Configuration. If Product Variants are used, VariantConfigValues should be null. The First Dictionary Key is the BOMConfiguration Classification Name and the second Dictionary contains the values which should be filtered.

```
public string GetNeighbourhoodAml(
        string ContextObjKey,
        string ContextObjectClass,
        string RelationClass,
        string Relationship,
        string AdditionalExpandInfo,
        IDictionary<string, string> SelectAttrs,
        string Clearance,
        string CompletlyIn,
        string sxMin,
        string syMin,
        string szMin,
        string sxMax,
        string syMax,
        string szMax,
        out string ErrorMessages)
```

Summary:
Creates AML string to search for components within given assembly lying within a boundingbox.
Parameters:

ContextObjKey:	Object ID of the assembly to filter the components for
ContextObjectClass:	Object class of the assembly to filter the components for
RelationClass:	class of relation with instance transformation matrix
Relationship:	relationship with expand direction (for example: CAD Structure/Structure_LeftToRight)
AdditionalExpandInfo:	Expand Info (Current, AsSaved, Released)
SelectAttrs:	Attributes to select
Clearance:	calculation tolerance
CompletlyIn:	whether component lies completely in bounding box (valid values are true or false)

sxMin, *syMin*, *szMin*: min bounding box point
sxMax, *syMax*. *szMax*: max bounding box point
ErrorMessages: error messages

public Aras.IOM.Item **CreateItem**(
string Type,
string Classification,
IDictionary<String, String> Attributes)
Summary:
Create Item.
Parameters:
Type: Item Type (for example 'CAD' or 'Part')
Classification: Item classification (for example 'Part/Component')
Attributes: Attributes for the item to create
Returns:
Created Aras.IOM.Item.

public Aras.IOM.Item **CreateRelation**(
Aras.IOM.Item LeftItem,
Aras.IOM.Item RightItem,
string PdmClass,
IDictionary<string, string> Attributes)
Summary:
Create Relation.
Parameters:
LeftItem: Parent item
RightItem: Child item
PdmClass Relation class (for example 'Part BOM')
Attributes: Attributes for the item to create
Returns:
Created Aras.IOM.Item representing relation.

public void **DeleteRelation**(
PwbServerAddin.PwbApi.Rel RelToDelete)
Summary:
Delete Relation.
Parameters:
RelToDelete: Relation to delete

public List<Item> **PerformQuery**(
string PdmClass,
string Classification,
string KeyAttrName,
string KeyAttrValue)
Summary:
Query for items.
Parameters:
PdmClass: Pdm class (for example 'CAD or 'Part)
Classification: Item classification (for example 'Assembly)
KeyAttrName: Attribute name to search for
KeyAttrValue: Attribute value to search for
Returns:
List of items fitting the input criteria.

public List<Item> **GetItemsFromIds**(
string TypeStr,
string IdListStr,

string SelectStr = null)
Summary:
Query items for input item IDs.
Parameters:
TypeStr: Pdm class (for example ‘CAD or ‘Part)
IdListStr: List of Ids to search for separated by ‘|’
SelectStr: selection string
Returns:
List of items fitting the input criteria.

public string **OrigCadPartnumberAttr**()
Summary:
Returns attribute storing the original part number of a document, before document was renamed by autonaming method. The attribute is defined in the PWB configuration.
Returns:
Name of the original part number attribute.

public void **SetRequestXml**(
string RequestName,
string RequestXml)
Summary:
Returns attribute storing the original part number of a document, before document was renamed by autonaming method. The attribute is defined in the PWB configuration.
Parameters:
RequestName: For example “PreProcessCadStructure”
RequestXml: Request xml to execute

public List<CadNodeInfo> **GetCadNodeInfoList**()
Summary:
Read result nodes from xml request, being performed before (see SetRequestXML)
Return:
Result nodes from xml request being executed before.

Manage Logging information:

public void **PdmClassToTypeInfo**(
string PdmClass,
out string Type,
out string Classification)
Summary:
Splits PdmClass into type and classification.
Parameters:
PdmClass: Pdm class to determine type and classification for
Type: Return type for given PdmClass
Classification: Return classification for given PdmClass

public void **LogDictionaryContent**(
IDictionary<string, string> DictToLog)
Summary:
Write dictionary content to log file.
Parameters:
DictToLog: Dictionary to write to log file

public void **AddLogLine**(
string logLine)
Summary:
Add string to Log File.

Parameters:
logLine: String to write to log file

public void **Log**(
string Str,
int TraceLevel = 1)
Summary:
Writes logging information to the pwb log file.
Parameters:
Str: The log information to write
TraceLevel: 1 – information is always written to log file
2 – information is only written if extended logging information is requested

public string **GetDictionaryString**(
IDictionary<string, string> StringDict)
Summary:
Converts the dictionary to a string for logging purpose.
Parameters:
StringDict: The dictionary
Returns:
A string containing the dictionary entry keys and related values.

public void **DisposeLog**()

Read information about calling context :

public string **GetPwbVersion**()
Summary:
Get version of current Pwb application.
Returns:
version string

public string **GetPwbCopyRight**()
Summary:
Get copyright information.
Returns:
copyright information string

public string **GetDrawingType**()
Summary:
Returns the connector drawing type string. By default it is "/CAD/Mechanical/Drawing", but it can be configured by defining the server setting "CATDrawingType".
Returns:
The connector drawing type string.

public void **GetGeneralDynDlgUpdInfo**(
out *PwbServerAddin.PwbApi.GeneralDlgInfo* GenlDlgInfo,
[System.Collections.Generic.List<*PwbServerAddin.PwbApi.CadLink*> CadLinks = null],
[System.Collections.Generic.IDictionary<string, string> InputObjectAttrDict = null])
Summary:
Class for general data which is not associated to one particular dialog (primary or secondary dialog).
Parameters:
GenlDlgInfo: General CAD attribute information.

CadLinks: CAD link types and target document ids from the CAD session.
InputObjectAttrDict: Attributes of a PDM object which may be passed to the method, like the item being duplicated. Does not always contain data, depending on the use case.

public *PwbServerAddin.PwbApi.Dlg* **GetDynDlgFromClient**(
PwbServerAddin.PwbApi.DlgOrdinal DialogOrdinal)
Summary:
Retrieves the primary or secondary dialog object which has been sent from the client. 'null' if it doesn't exist. The secondary dialog only exists in the 'PdmUpdate' context, if a Part item is being created in addition to the CAD item.
Parameters:
DialogOrdinal: 'Primary' or 'Secondary'.
Returns:
The dialog if it exists, or 'null' otherwise.

public *PwbServerAddin.PwbApi.Dlg* **CreateNewDynDlg**(
PwbServerAddin.PwbApi.DlgOrdinal DialogOrdinal,
in string ClassStr,
in string FormStr
in string contextString)
Summary:
Creates a new dialog object to be returned to the client.
Parameters:
DialogOrdinal: The primary dialog, or the secondary (the Create dialog of the Part item in 'Update PDM').
ClassStr: Corresponds to <object name="..." in the PWBSchema.xml file.
FormStr: Corresponds to <form name="..." in the PWBSchema.xml file.
contextString: Context

public void **SetDynDlgOutputInfo**(
PwbServerAddin.PwbApi.DlgOrdinal DialogOrdinal,
PwbServerAddin.PwbApi.Dlg DialogToReturn)
Summary:
Defines the dialog to be returned as the primary or secondary dialog to the client.
Parameters:
DialogOrdinal: Primary or secondary.
DialogToReturn: The dialog object.

public bool **IsCadDoc**(
string PdmClass)
Summary:
Returns whether the class string refers to a CAD document.
Parameters:
PdmClass: The class string, as defined in the PWBSchema.xml file, for example "/CAD/Mechanical/Assembly".
Returns:
'true' if it is a CAD, otherwise 'false'.

public bool **IsPart**(
string PdmClass)
Summary:
Returns whether the class string refers to a Part item.
Parameters:
PdmClass: The class string, as defined in the PWBSchema.xml file, for example "/Part/Assembly".
Returns:
'true' if it is a Part item, otherwise 'false'.

public IDictionary<string, string> **GetSessionInfoDictionary**()
Summary:
Determines information about the current session.
Returns:
Dictionary with session information.

public bool **IsInReconnectContext**()
Summary:
Checks whether the client calls (only used for synchronize calls) requests for checking if an item with the same item number exists in PDM and should be reconnected to the item to synchronize.
Returns:
Whether the reconnection with existing items in PDM is requested.

public HashSet<string> **GetDynDlgKeySet**()
Summary:
Get list of the main keys.
Returns:
List of main keys.

public List<string> **GetMainKeyList**()
Summary:
Get list of the main keys containing information about dynamic dialogs and on server defined context methods to be given back to client.
Returns:
List of main keys.

Write information to result item:

Server methods often require a result item to be given back. The result item creation is defined and created using the following methods.

public void **SetAutonameValue**(
    string autonameValue)
Summary:
Sets the string which is set as the automatically generated item number of a new CAD or Part item.
Parameters:
autonameValue: The item number of the new item to be created.

public void **AddMessageToUser**(
    string Message,
    MessageType Type = MessageType.Information)
Summary:
Writes messages to the result item, which should be given back to the client in the following.
Parameters:
Message: The message to return to the client
Type: possible values:
    MessageType.Information
    MessageType.Warning
    MessageType.Error

public void **AddInfoForClient**(
    string AttrName,

string AttrValue)
Summary:
Add information to be given back to the client.
Parameters:
AttrName: Name of attribute
AttrValue: Value of attribute

public void **SetPartNumberInfo**(
string PartNumber,
string Key,
string Value)
Summary:
Writes part number information to the result item.
Parameters:
PartNumber: The part number to return to the client
Key: The related object key
Value: The related value

public void **SetFileNameInfo**(
string PartNumber,
string Key,
string Value)
Summary:
Write file name information to the result item
Parameters:
PartNumber: The part number
Key: The related object key
Value: The related file information

public void **SetDictionaryInfo**(
string PartNumber,
string Key,
string Value,
ref IDictionary<string, IDictionary<string, string>> Dict)
Summary:
Writes information to the result item.
Parameters:
PartNumber: The part number
Key: The related object key
Value: The dictionary

public void **SetResultSuccess**(
bool ResultSuccess)
Summary:
Writes information to the result item.
Parameters:
ResultSuccess: The operation success flag

public Item **GetResultItem**()
Summary:
Get the result item.
Returns:
The result item, containing information set by the previously described methods.

public List<Aras.IOM.Item> **QueryItemsForCustomItemInfo**(
string CustomItemInfoInputStr)

Summary:
Query for items satisfying criteria, given by input string.
Parameters:
CustomItemInfoInputStr: String containing information about item ids and class to query for.
Input string example:
Part
||
616503DA9CDC4ABDA0DFCCDE0709606C|
7DC98B39DED247B0AE2E05ED9DC37973|
12EAEAE70036409B84BBB5933BC79757
|||
CAD
||
0B5A2004586A4EEABC22F0AA94846CF8|
E52430BADE164F809CF9880F2919447C|
FF0A560A5B6F49F294E4E7F28C042CEE
Returns:
Result item list, satisfying input criteria.

PWB Utilities

public bool **ParseBoolean**(
string BoolValue)
Summary:
Verifies if passed string represents a boolean value. It returns true for "TRUE", "true" and false for any other value (or null or empty string).
Parameters:
BoolValue: The string to analyse.
Returns:
Boolean value, which represents value of the passed string

public bool **IsNullOrEmpty**(
string Str)
Summary:
Verifies if passed string is null or empty
Parameters:
Str: The string to analyse.
Returns:
Whether passed string is null or empty

public string **GetApiVersion**()
Summary:
Version string, using semantic versioning of the API:
The major version is incremented when we make incompatible API changes (e.g. 1.2.3 -> 2.0.0),
the minor version is incremented when we add functionality in a backwards-compatible manner (e.g. 1.2.3 -> 1.3.0),
the patch version is incremented when we make backwards-compatible bug fixes (e.g. 1.2.3 -> 1.2.4).
Returns:
Version string.

public List<string> **StringToList**(
string StringToParse)
Summary:
Split string into substrings taking “|” as deliminator.
Parameters:
StringToParse: String to parse for split.

Returns:
List containing substrings

public string **StringListToString**(
List<string> StringList)
Summary:
Combine List of strings to strings taking “|” as deliminator.
Parameters:
StringList: Stringlist to combine.
Returns:
Combined string

public IDictionary<string, List<string>> **DialogStringListAttrsItemToDictionary**(
Aras.IOM.Item DialogItemObj)
Summary:
Write attribute from input dialog item to dictionary.
Parameters:
DialogItemObj: Input dialog item.
Returns:
Dictionary containing attribute names and values

public class CadNodeInfo

public class **CadNodeInfo**
Summary:
Class to manage PDM node information.

Methods:

Constructor:

public **CadNodeInfo**(
bool IsModified,
string PartNumber,
string FileName,
IDictionary<string, string> CadStdAttrs,
IDictionary<string, string> UserDefCadAttrs,
IDictionary<string, string> SpecialCadProperties,
IDictionary<string, string> PartPdmNodeData,
IDictionary<string, string> CadPdmNodeData,
List<IDictionary<string, string>> BomChildren)
Summary:
Constructor for CadNodeInfo class
Parameters:
IsModified: Whether node is modified
PartNumber: Item number
FileName: Name of related file
CadStdAttrs: Cad standard attributes
UserDefCadAttrs: User defined attributes
PartPdmNodeData: Data from related part node
CadPdmNodeData: Data from related cad node
BomChildren: Bom child nodes

Methods to read node data

public bool **GetIsModified**()

public string **GetPartNumber**()

public string **GetFileName**()

public IDictionary<string, string> **GetCadStdAttrs**()

public IDictionary<string, string> **GetCadUserDefAttrs**()

public IDictionary<string, string> **GetSpecialCadProperties**()

public IDictionary<string, string> **GetPartPdmNodeData**()

public IDictionary<string, string> **GetCadPdmNodeData**()

public int **GetBomChildrenCount**()

public IDictionary<string, string> **GetBomChildData**(int i)

Namespace PwbServerAddin.PwbApi

public class GeneralDlgInfo

PwbServerAddin.**PwbApi.GeneralDlgInfo**
Summary:
Class for general data which is not associated to one particular dialog (primary or secondary dialog).

Methods:

Constructor:

PwbServerAddin.**PwbApi.GeneralDlgInfo(**
Aras.IOM.Item CallingMethodItem)
Summary:
Constructor for GeneralDlgInfo class.
Parameters:
CallingMethodItem: Item containing information about the calling context.

Methods:

public Aras.IOM.Item **GetDialogInputItem**(
PwbServerAddin.PwbApi.InputItem *inputItemType*)
Summary:
Retrieves a specific item which has been passed as input. The input item type 'CorrespondingPdmObj' exists for the dialog context 'ContextAction', and 'DuplicateSourceFile' and possibly 'DuplicateSourcePart' exist for the dialog context 'Duplicate'.
Parameters:
inputItemType: Can be 'CorrespondingPdmObj', 'DuplicateSourceFile', or 'DuplicateSourcePart'.
Returns:
The item which corresponds to the input item type if it exists, 'null' otherwise.

public System.Collections.Generic.List<Aras.IOM.Item> **GetDialogInputItemList**(
PwbServerAddin.PwbApi.InputItemList *inputItemListType*)
Summary:
Returns a list of items. Currently the only valid input value is 'BomPartsInSession' for the dialog context 'Create'.
Parameters:
inputItemListType:Only 'BomPartsInSession' is valid.
Returns:
The list of Part items in the CAD session, if any exist.

public void **SetDialogInputItem**(
PwbApi.InputItem inputItemType,
Aras.IOM.Item item)
Summary:
Parameters:
inputItemType: Can be 'CorrespondingPdmObj', 'DuplicateSourceFile', or 'DuplicateSourcePart'.
item: The item which corresponds to the input item type.

public void **SetDialogInputItemList**(
PwbApi.InputItemList inputItemListType,
Aras.IOM.Item item)
Summary:
Parameters:
inputItemListType: Can be 'CorrespondingPdmObj', 'DuplicateSourceFile', or 'DuplicateSourcePart'.
item: The items corresponding to the input item type.

public System.Collections.Generic.IList<string> **GetUdpNames**()
Summary:
A list of the user-defined property names.
Returns:
The names in a string list.

public string **GetUdpValue(**
string *name*)
Summary:
The value of a specific user-defined property.
Parameters:
name: The UDP name.
Returns:
The UDP value corresponding to the name.

Attributes:

public string **CadComponentName** { get; }
Summary:
The CAD component name (Context=Create).

public string **CadDefinition** { get; }
Summary:
The CAD definition (Context=Create).

public string **CadDescriptionReference** { get; }
Summary:
The CAD reference description (Context=Create).

public string CadFileName { get; }
Summary:
The CAD file name (Context=Create).

public string **CadFileType** { get; }
Summary:
The CAD file type, for example "CATDrawing".

public string **CadNomenclature** { get; }
Summary:
The CAD nomenclature (Context=Create).

public string **CadPartNumber** { get; }
Summary:
The CAD part number, usually mapped to the Innovator item number (Context=Create).

public string **CadPsnSpecType** { get; }
Summary:
The internal CAD specification type (Context=Create).

public string **CadRevision** { get; }
Summary:
The CAD revision (Context=Create).

public PwbServerAddin.PwbApi.DlgContext **Context** { get; }
Summary:
The context in which the server method is called. Possible values are 'Create', 'PdmUpdate', 'Duplicate', and 'ContextAction'.

public string **ContextProduct** { get; }
Summary:
Context product, needed for specific use cases.

public string **ContextProductId** { get; }
Summary:
Context product ID, needed for specific use cases.

public string **OriginatedFromCad** { get; }
Summary:
For 'Context=Duplicate' the ID of the original CAD item.

public string **OriginatedFromPart** { get; }
Summary:
For 'Context=Duplicate' the ID of the original Part item, if it exists.

public string **CadId** { get; set; }
Summary:
The ID of the CAD document where the link points to.

public string **CadUDP** { get; set; }
Summary:
The CAD user-defined properties (Context=Create)

public string **CadType** { get; set; }
Summary:
The type, including the classification, of the CAD document the link points to.
Example: "/CAD/Mechanical/Part".

public string **Type** { get; set; }
Summary:
The type string, corresponds to the classification of the 'CAD Structure' relation.

public string **IsBom** { get; set; }
Summary:
Whether the CAD model node is BOM or non-BOM. 'true' is BOM. (Context=Create).

public class Dlg

public class **Dlg**
Member of namespace PwbServerAddin.PwbApi
Summary:
Class representing dialog information which is dynamically returned to the user.

Methods:

Constructor:

public **Dlg**(
PwbServerAddin.PwbServerApi PwbServerApiObj,
IDictionary<string,string> GeneralDialogInfoDict,
IDictionary<string,string> InputDialogValueDict = null,
IDictionary<string,string> InputDialogTypeDict = null,
string DlgDtxStr ="")
Summary:
Dlg class constructor.
Parameters:
PwbServerApiObj: PwbServerApi object
GeneralDialogInfoDict: Dictionary containing the dialog XML attributes.
InputDialogValueDict: Dictionary containing the dialog attribute values.
InputDialogTypeDict: Dictionary containing the dialog element types
DlgDtxStr:

Methods:

public void **BeginDialog**()
Summary:
Has to be called before adding information to the dialog.

public void **BeginDialog**(string title)
Summary:
Has to be called before adding information to the dialog.
Parameters:
title: Dialog title

public void **BeginDialog**(List<string> titleStrList)
Summary:
Has to be called before adding information to the dialog.
Parameters:
title: Dialog title list

public void **AddDlgAttrWidget**(
System.Collections.Generic.IDictionary<string, string> *xmlAttrs,*
[System.Collections.Generic.List<PwbServerAddin.PwbApi.Dlg.ListValueEntry> *values = null*])
Summary:
Adding information about one attribute widget to the dialog.

Parameters:
xmlAttrs: Dictionary containing the dialog XML attributes. Example (like in the PWBSchema.xml file): name="item_number" widgetType="ComboBox" mode="update" visibleLength="15" required="false" listViewRelevant="true" entryAllowed="true"
values: List containing the values of a list widget.

public void **AddActionDlgAttrWidget**(
System.Collections.Generic.IDictionary<string, string> *xmlAttrs*,
System.Collections.Generic.List<PwbServerAddin.PwbApi.Dlg.ActionListValueEntry> *actionValues*)
Summary:
Adds a widget performing a specific action to the dialog. Currently only "name"="PwbCorrespondingPart" is allowed.
Parameters:
xmlAttrs: Dictionary containing the dialog XML attributes
actionValues: Currently only the actions NoPart, CreateCorrespondingPart, QueryForCorrespondingPart, RelateCorrespondingPart, FirstPartInSession, and DisplayMessage are allowed.

public void **AddAttributeValues**(
System.Collections.Generic.IDictionary<string, string> *StringAttrValues*,
System.Collections.Generic.IDictionary<string,
System.Collections.Generic.List<string>> *StringListAttrValues*)
Summary:
Adding values to the dialog to be returned to the user.
Parameters:
StringAttrValues: Values of "String" type attributes.
StringListAttrValues: Values of "StringList" type attributes.

public void **AddControlSettings**(
System.Collections.Generic.Dictionary<PwbServerAddin.PwbApi.ControlSetting, string> *ControlSettings*)
Summary:
Adding control setting which trigger specific actions on the client. Currently only NoPart, CreateCorrespondingPart, QueryForCorrespondingPart, RelateCorrespondingPart, FirstPartInSession, and DisplayMessage are allowed.
Parameters:
ControlSettings: A dictionary containing control settings and an optional data string.

public void **EndDialog**()
Summary:
Has to be called after adding the last piece of information to the dialog.

Attributes:

public string **Key** { get; }
Summary:
The dialog key. Only "Primary" or "Secondary" are currently valid.

public string **Class** { get; }
Summary:
The dialog class. Corresponds to the object name in the PWBSchema.xml file. Example values are "/CAD/Mechanical/Part" or "/Part/Component".

public string **Form** { get; }

Summary:
The dialog form. Corresponds to the form name in the PWBSchema.xml file. Example values are "Create" or "Register".

public string **ChangedAttribute** { get; }
Summary:
The name of the changed attribute that triggered the call of the server method.

public string **NewAttributeValue** { get; }
Summary:
The new value of the changed attribute that triggered the call of the server method.

public PwbServerAddin.PwbApi.CorrespondingPartAction **Action** { get; }
Summary:
The 'CorrespondingPart' action.

public string **ActionData** { get; }
Summary:
Optional additional 'CorrespondingPart' action data, like for example a part type, or an item ID.

public System.Collections.Generic.IDictionary<string, string> **InputValueDict** { get; }
Summary:
Dictionary containing the dialog attribute input values.

public System.Collections.Generic.IDictionary<string, string> **InputTypeDict** { get; }
Summary:
Dictionary containing the dialog attribute input widget types, for example "String", "StringList" or "Boolean".

public class ListValueEntry

public class **ListValueEntry**
Member of namespace PwbServerAddin.PwbApi.Dlg
Summary:
Class handling the values of a widget.

Methods:

Constructor:

public **ListValueEntry**(
string name,
string displayName)
Summary:
Constructor for class ListValueEntry
Parameters:
name: Widget name
displayName: Widget display name

Attributes:

public string **Name** { get; set;}
Summary:
The type string corresponds to a widget name.

public string **DisplayName** { get; set;}
Summary:
The type string corresponds to a widget display name.

public class ActionListValueEntry

public class **ActionListValueEntry:ListValueEntry**
Member of namespace PwbServerAddin.PwbApi.Dlg
Summary:
Class handling an action associated with a widget.

Methods:

Constructor:

public **ActionListValueEntry**(
string name,
CorrespondingPartAction action,
string actiondata ="",
string displayName = "")
Summary:
Constructor for class ListValueEntry
Parameters:
name: Widget name
action: Possible action type values are:
Invalid = -1,
NoPart,
CreateCorrespondingPart,
QueryForCorrespondingPart,
RelateCorrespondingPart,
FirstPartInSession,
DisplayMessage,
CustomNoPart
actiondata: Required data for the action
displayName: Widget display name

Attributes:

public string **Action** { get; set;}
Summary:
The type string corresponds to the classification of the CAD Structure relation.

public string **ActionData** { get; set;}
Summary:
Required data for the action.

public class CadLink

public class **CadLink**
Member of namespace PwbServerAddin.PwbApi
Summary:
Class representing link information.

Attributes:

public string **Type** { get; set;}
Summary:
The type string corresponds to the classification of the CAD Structure relation.

public string **CadId** { get; set;}
Summary:

ID of the referenced CAD document.

public string **CadType**{ get; set;}
Summary:
Type of the referenced CAD document (Example: "/CAD/Mechanical/Part").

public class Obj

public class **Obj**
Member of namespace PwbServerAddin.PwbApi
Summary:
Class representing CAD document and part objects.

Methods:

Constructor:

public **Obj** (
Aras.IOM.Item ItemObj)
Summary:
Constructor for Obj class.

public **Obj** (
PwbServerAddin.Base.ExpandResult.Obj PdmObj)
Summary:
Constructor for Obj class.

public string **GetStringInfo**(
string AttrName = "")
Summary:
Returns information string for logging purpose.

public Dictionary<string, string> **GetAttrs**(
HashSet<string> AttrNames)
Summary:
Returns attribute values for requested attributes.

public Aras.IOM.Item **GetItem**()
Summary:
Returns corresponding Aras.IOM.Item.

Attributes:

public Dictionary<string, string> **Attr**
Summary:
List of to Obj associated attributes.

public class Rel

public class **Rel**
Member of namespace PwbServerAddin.PwbApi
Summary:
Class representing relations.

Methods:

Constructor:

public **Rel** (
Aras.IOM.Item RelItemObj)
Summary:
Constructor for Rel class corresponding to Aras.IOM.Item relation item.

public string **GetLeftObjID** ()
Summary:
Gives back object ID of parent object.

public string **GetRightObjID** ()
Summary:
Gives back object ID of child object.

public string **GetRightObjClassification** ()
Summary:
Gives back classification of child object.

public string **GetRelType** ()
Summary:
Gives back type of relation.

public string **GetStringInfo**(
string AttrName = "")
Summary:
Returns information string for logging purpose.

Attributes:

public Dictionary<string, string> **Attr**
Summary:
List of to Rel associated attributes.

public Aras.IOM.Item **LeftObjItem** { get; set;}
Summary:
Returns the associated relation parent.

public Aras.IOM.Item **RightObjItem** { get; set;}
Summary:
Returns the associated relation child.

public Aras.IOM.Item **RelItem** { get; set;}
Summary:
Returns the associated relation.

CHAPTER 2

Extent PDM Workbench by own Server Methods

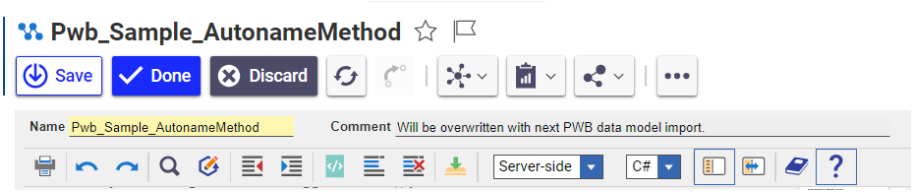
A powerful instrument to adapt the PDM Workbench to own needs is to implement server methods.

Own server methods might be implemented for the PWB in different use cases:

- Pre- or Postprocessing during item update to PDM.
- Autonaming.
- Reconnecting an item during update to PDM to an existing item in PDM.
- Adding and implementing additional context functions in PDM.
- Defining the input parameters for the create dialog.
- …

Define own Server Method

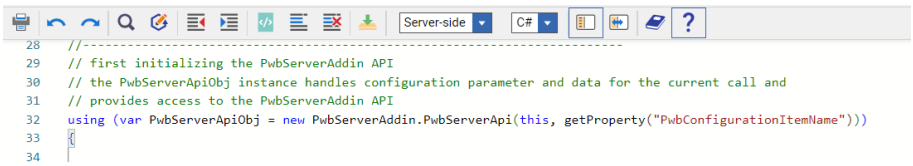
The user defined server methods need to be defined as Server-side C# method and the execution should be allowed for all suitable identities.



The PwbServerApi object handles configuration parameter and data for the current call.

Therefore the PwbServerApi is instantiated first.

The current configuration is retrieved from the PwbConfigurationItemName property, which is used for CATIA and NX.



Input

The calling item is an Aras.IOM.Item. It contains information about the calling purpose, which are stored as item properties.

For example “reading some properties for an autonaming method”:

```
// Preparing the input information
string Autoname = this.getProperty("Autoname");
string PdmType = this.getProperty("Type");
string PdmClassification = this.getProperty("Classification");
```

Further input data is retrieved from the PwbServerApi.object (see: Read information about calling context)

Output

The own server method always needs to return an Aras.IOM.Item.

Depending on the scope of the server method a result Item might be requested, which is created by the GetResultItem method of the PwbServerApi object.

To write information to the result item see: Write information to result item.

Be aware that sometimes other Aras.IOM.Items are requested as return value. For example the server method called for the search of in PDM existing items to be reconnected to an item during update from CAD, needs to give back the suitable item found on server.

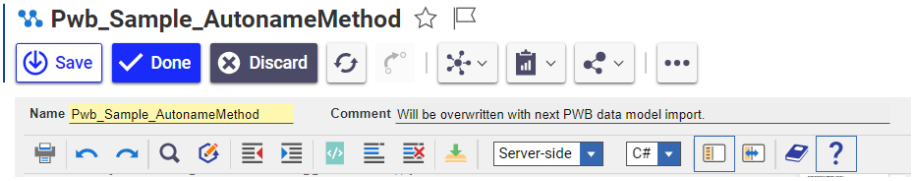
CHAPTER 3

Implementing Autonaming Methods

The autoname functionality might be implemented by a server method. Therefore, the function has to be configured in the file “PWBSchema.xml” (see Admin manual for the desired CAD system).

Example method:

The autoname method needs to be on the server-side as C# method and the execution needs to be allowed to for the suitable identities.



First the PwbServerApi class is instantiated, which provides information about the calling context and provides additional methods required in the PWB context.

```
using (var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this))
{
```

Information is read from the calling Aras.IOM.Item object for example about the object Type and Classification of the object to be named or the data structure mode.

```
// Preparing the input information
string Autoname = this.getProperty("Autoname");
string PdmType = this.getProperty("Type");
string PdmClassification = this.getProperty("Classification");

string UseBomPartStructure = this.getProperty("UseBomPartStructure");
string NonBomInBomMode = this.getProperty("NonBomInBomMode");
```

The item name is created for example through getting it from a sequence.

```
// Getting the actual autoname value
string AutonameValueItemnumber = "";
AutonameValueItemnumber = PwbServerApiObj.GetNextAutonameSequence("CAD Document");
```

Finally, the result Aras.IOM.Item object is retrieved from the PwbServerApi object and given back as return value.

```
// Preparing the output
PwbServerApiObj.SetAutonameValue(AutonameValueItemnumber);

return PwbServerApiObj.GetResultItem();
```

Possibility to use CATIA User-Defined Properties in Autoname and PreProcDlgAttrs Custom Methods

It is possible to use user-defined properties in the autoname method and the customization method defined by the setting ‘CustomMethod_PreProcCreDlgAttrs’.

Here is how to retrieve them in the autoname method (Pwb_Sample_AutonameMethod):

```
// Standard CATIA Properties
Item CadStdPropsItem = this.getPropertyItem("CadStdProps");
IDictionary<string, string> CadStdPropsDict = null;
if (CadStdPropsItem != null)
{
	CadStdPropsDict =
		PwbServerApiObj.DialogAttrsItemToDictionary(
			CadStdPropsItem);
}

// NEW: User-defined CATIA Properties
Item CadUserDefPropsItem = this.getPropertyItem("CadUserDefProps");
IDictionary<string, string> CadUserDefPropsDict = null;
if (CadUserDefPropsItem != null)
{
	CadUserDefPropsDict =
		PwbServerApiObj.DialogAttrsItemToDictionary(
			CadUserDefPropsItem);
}
```

Here is how to retrieve them in Pwb_Sample_PreProcDlgAttrs:

```
// CATIA user-defined properties
IDictionary<string, string> CatiaUserDefPropsDict = null;
Item CatiaUserDefPropsItem =
	this.getPropertyItem("CadUserDefProps");
if (CatiaUserDefPropsItem != null)
{
	CatiaUserDefPropsDict =
		PwbServerApiObj.DialogAttrsItemToDictionary(
			CatiaUserDefPropsItem);
}
if (CatiaUserDefPropsDict != null)
{
	foreach (KeyValuePair<string, string> Attribute in CatiaUserDefPropsDict)
	{
		PwbServerApiObj.Log(
			"CATIA user-defined property '" + Attribute.Key +
			"'/'" + Attribute.Value + "'");
	}
}
```

CHAPTER 4

Server-side Create dialog definition

Dialogs can also be defined on the server side. How to configure the create dialog on the server side is described within this example.

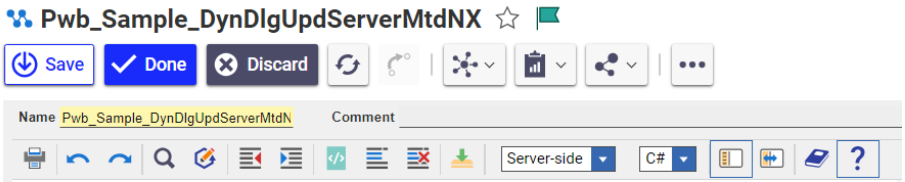
Main method for dynamic dialog creation

First the entry point for the dialog definition needs to be defined within schema file as dynDlgServerMethod:

<PWBSchema ... dynDlgServerMethod="Pwb_Sample_DynDlgUpdServerMtdNX">

Example method

The server method for the dialog definition is implemented as server-side C# method.



First the PwbServerApi class is instanciated.

```
//--------------------------------------------------------------------
// first initializing the PwbServerAddin API
// the PwbServerApiObj instance handles configuration parameter
// and data for the current call and
// provides access to the PwbServerAddin API
using (var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this))
{
```

Then the GeneralDlgInfo object is retrieved, which contains information about the calling client function.

```
// GetGeneralDynDlgUpdInfo
// General information - get current input parameter from client
//
// OUTPUT parameter:
// generalDlgInfo - PwbServerAddin.PwbApi.GeneralDlgInfo.Context - provides information
//                                                    about the calling client function :
//          PwbServerAddin.PwbApi.DlgContext.PdmUpdate - Call from update function
//          PwbServerAddin.PwbApi.DlgContext.Create    - Call from create item function
//          PwbServerAddin.PwbApi.DlgContext.ContextAction  - Call from context function
//          PwbServerAddin.PwbApi.DlgContext.Duplicate - not used for NX
//          PwbServerAddin.PwbApi.GeneralDlgInfo.CadFileType  - is always prt
// cadLinks     - not yet used for NX
// inputObjectAttrDict - current dialog values from calling dialog
PwbServerApiObj.GetGeneralDynDlgUpdInfo(
  out PwbServerAddin.PwbApi.GeneralDlgInfo generalDlgInfo,
  cadLinks,
  inputObjectAttrDict);
```

The ‘create’ dialog is given back for the suitable context. First the attributes of the currently on client shown attributes are determined and then the new dialog is created taking into account the current dialog values.

Here the new ‘create’ dialog for CAD Document objects is created with the ProcessDlgForCad method, which is described later.

```
//-----------------------------------------------------------------
// create dialog with respect to the calling NX method context
//

if (Context == PwbServerAddin.PwbApi.DlgContext.Create)
{
  //-----------------------------------------------------------------
  // create dialog is requested in context of pdm update action
  //-----------------------------------------------------------------

  //-----------------------------------------------------------------
  // First dialog (for cad document item)
  // get the previously shown dynamic dialog from client
  var primaryReturnDialog =
    PwbServerApiObj.GetDynDlgFromClient(
      PwbServerAddin.PwbApi.DlgOrdinal.Primary);

  // add dialog elements to the new primary dialog
    ProcessDlgForCad(
      PwbServerApiObj,
      cadLinks,
      inputObjectAttrDict,
      CadFileType,
      Context,
      primaryReturnDialog);
```

The new dialog is given back to the PwbServerApi object, and it is checked whether a dialog for a related Part object is required.

The new ‘create’ dialog for Part objects is created with the ProcessDlgForCad method.

An example for dialog creation is given within the next paragraph see “create dialog creation”.

```
// return dialog to the server addin
PwbServerApiObj.SetDynDlgOutputInfo(
  PwbServerAddin.PwbApi.DlgOrdinal.Primary,
  primaryReturnDialog);

var cadItemClass = primaryReturnDialog.Class;

//------------------------------------------------------------------
// Possibly return a secondary dialog (for Part item)

// get the previously shown dynamic dialog from client (second dialo
PwbServerAddin.PwbApi.Dlg secondaryReturnDialog =
  PwbServerApiObj.GetDynDlgFromClient(
    PwbServerAddin.PwbApi.DlgOrdinal.Secondary);

// if Part creation is required, create second dialog for the part ite

var CreatePart_Mode =
PwbServerApiObj.ParseBoolean(
PwbServerApiObj.GetSessionInfoDictionary()["NX_CreatePartMode"]);

if (!CreatePart_Mode)
{
  // "NoPart" switches from a possibly existing
  // secondary (== Part) dialog to 'no dialog'.
  secondaryReturnDialog = null;
}
else
{
  // Creating a new dialog
  string partClass = "/Part/Component";

  // Creates a new dialog object to be returned to the client.
  secondaryReturnDialog =
    PwbServerApiObj.CreateNewDynDlg(
      PwbServerAddin.PwbApi.DlgOrdinal.Secondary, partClass, "Create"
}

if (secondaryReturnDialog == null)
{
  // No secondary dialog, returning.
  return PwbServerApiObj.GetResultItem();

}

// add dialog elements to the new secondary dialo
ProcessForGeneralItem(
  inputObjectAttrDict,
  secondaryReturnDialog,
  PwbServerApiObj);

// return second dialog to the server addin
PwbServerApiObj.SetDynDlgOutputInfo(
  PwbServerAddin.PwbApi.DlgOrdinal.Secondary,
  secondaryReturnDialog);
}
```

```
  return PwbServerApiObj.GetResultItem();
}
```

Create dialog creation

The following paragraph presents a method for a create dialog creation taking as example the CAD Documents:

```
//--------------------------------------------------------------------------------
// create dialog PwbServerAddin.PwbApi.Dlg for cad document item
//
// INPUT parameter
//  PwbServerApi PwbServerApiObj - PwbServerApi contains current configuration and
//  provides API for current client call
//  List<PwbServerAddin.PwbApi.CadLink> cadLinks - not yet used
//  IDictionary<string, string> inputObjectAttrDict - values from previous dialog
//  string CadFileType -
//  PwbServerAddin.PwbApi.DlgContext - calling function from client
//
// OUTPUT parameter
//  PwbServerAddin.PwbApi.Dlg CadReturnDialog - the created dialog PwbServerAddin.PwbApi.Dlg
//  List<Item> bomPartsInSession -
// RETURN parameter
      private static void ProcessDlgForCad(
        PwbServerApi PwbServerApiObj,
        List<PwbServerAddin.PwbApi.CadLink> cadLinks,
        IDictionary<string, string> inputObjectAttrDict,
        string CadFileType,
        PwbServerAddin.PwbApi.DlgContext Context,
        PwbServerAddin.PwbApi.Dlg CadReturnDialog)
      {

        CadReturnDialog.BeginDialog();
```

The dialog is created as PwbServerAddin.PbwApi.Dlg object. First the dialog is initialized by calling the BeginDialog() method. Then the dialog elements are added step by step by the AddDlgAttrWidget().

```
// "item_number"
CadReturnDialog.AddDlgAttrWidget(
  new Dictionary<string, string>
  {
  {"name", "item_number"},
  {"displayName", "Doc. Number"},
  {"widgetType", "SingleLineEditor"},
  {"mode", "update"},
  {"visibleLength", "15"},
  {"allowedLength", "32"},
  {"required", "true"}
  });
```

```
// "name"
CadReturnDialog.AddDlgAttrWidget(
  new Dictionary<string, string>
  {
  {"name", "name"},
  {"displayName", "Name"},
  {"widgetType", "SingleLineEditor"},
  {"mode", "update"},
  {"required", "false"},
  {"visibleLength", "15"},
  {"allowedLength", "256"}
  });

// "description"
CadReturnDialog.AddDlgAttrWidget(
  new Dictionary<string, string>
  {
  {"name", "description"},
  {"displayName", "Description"},
  {"widgetType", "MultiLineEditor"},
  {"mode", "update"}
  });

// "is_template"
CadReturnDialog.AddDlgAttrWidget(
  new Dictionary<string, string>
  {
  {"name", "is_template"},
  {"displayName", "is template"},
  {"widgetType", "SingleCheckBox"},
  {"mode", "update"},
  {"updateFormIfModified", "true"}
  });
```

If the dialog is already shown and should now be updated, the dialog values, already keyed in on the client side, are entered to the dialog elements.

Finally the dialog creation is finished by calling the EndDialog() method.

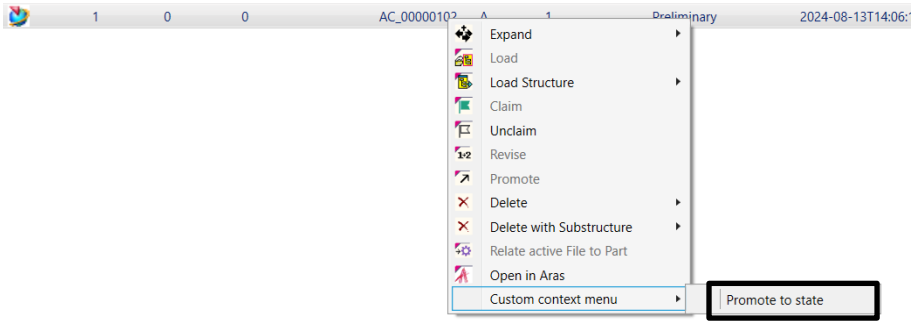
```
// 2. The dialog attribute values
// Simple string attributes
IDictionary<string, string> PrimaryDlgStringAttrValues =
  new Dictionary<string, string>();
```

```
// Then, if they exist, the values of the previous dialog, which may
// have been changed by the user.
foreach (var Entry in CadReturnDialog.InputValueDict)
{
  if (PrimaryDlgStringAttrValues.ContainsKey(Entry.Key))
  {
    PrimaryDlgStringAttrValues[Entry.Key] = Entry.Value;
  }
  else
  {
    PrimaryDlgStringAttrValues.Add(Entry.Key, Entry.Value);
  }
}

  // String list attributes
  IDictionary<string, List<string>> PrimaryDlgStringListAttrValues =
    new Dictionary<string, List<string>>();

  CadReturnDialog.AddAttributeValues(
      PrimaryDlgStringAttrValues,
      PrimaryDlgStringListAttrValues);


  CadReturnDialog.EndDialog();

}
```

CHAPTER 5

Server-side Context dialog definition for “Promote to state” action

The context dialog of an Aras object might be extended with help of methods implemented on the aras server.



Schema file extension

Therefore, the schema file has to be extended on the client side. The customContextAction is added to the selected object:

```
<object name="/CAD/Mechanical/Part" displayName="Model" icon="NXPart">
  ...
  <customContextAction name="Pwb_Sample_DynContextPromote" usedIn="PdmWindow|QueryDialog"
  confirm="false" dialog="Pwb_Sample_ContextPromoteDlg" displayName="Promote to state"/>
```

The customContextAction includes the following attributes:

name: Name of the OK Action

usedIn: context of the method call

confirm: whether the user has to confirm the action after context action selection

dialog: name of the related dialog

displayName: title of the related form

Moreover, the entry point for the dialog definition needs to be defined within the schema file as dynDlgServerMethod (see also Main method for dynamic dialog creation).

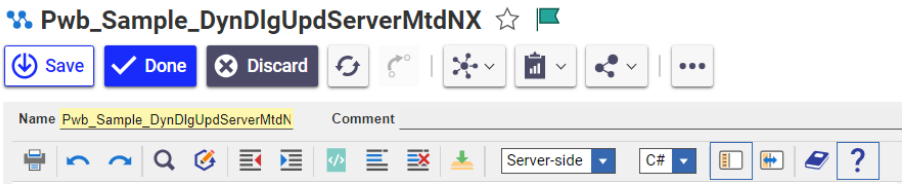
<PWBSchema ... dynDlgServerMethod="Pwb_Sample_DynDlgUpdServerMtdNX">

Main method for context dialog definition

Example method

Also, for context dialog definition, the main method for dialog definition is used as entry point. In the following the adaptions of this method for the context dialog definition is explained.

As explained at the beginning of this chapter the paragraph ‘the server method for the dialog definition is implemented as server-side C# method.



First the PwbServerApi class is instantiated.

```
//--------------------------------------------------------------------
// first initializing the PwbServerAddin API
// the PwbServerApiObj instance handles configuration parameter
// and data for the current call and
// provides access to the PwbServerAddin API
using (var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this))
{
```

Then the GeneralDlgInfo object is retrieved, which provides information about the calling client function context.

```
// GetGeneralDynDlgUpdInfo
// General information - get current input parameter from client

PwbServerApiObj.GetGeneralDynDlgUpdInfo(
  out PwbServerAddin.PwbApi.GeneralDlgInfo generalDlgInfo,
  cadLinks,
  inputObjectAttrDict);
```

The PwbServerAddin.PwbApi.Dlg object is received from the GetDynDlgFromClient method. It represents one dialog.

The PwbServerAddin.PwbApi.Dlg object provides the name of the requested dialog.

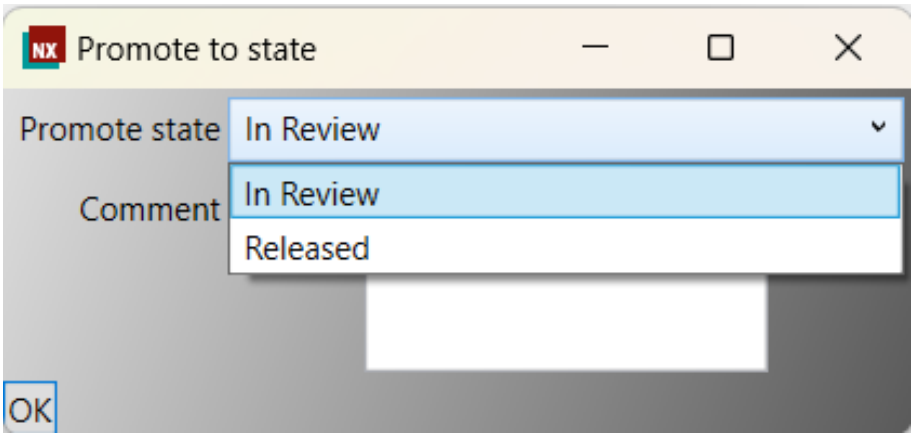
The dialog elements have to build up here and then the object is returned to the PwbServerApi.

```
if (Context == PwbServerAddin.PwbApi.DlgContext.ContextAction)
{
  //-----------------------------------------------------------------
  // dynamic context action dialog is requested
  //-----------------------------------------------------------------

  // get the previously shown dynamic dialog from client
  var primaryReturnDialog =
    PwbServerApiObj.GetDynDlgFromClient(
      PwbServerAddin.PwbApi.DlgOrdinal.Primary);

  if (primaryReturnDialog.Form == "Pwb_Sample_ContextPromoteDlg")
```

```
  {
    ProcessForPromoteCad(inputObjectAttrDict, correspondingPdmObjItem,
      primaryReturnDialog, PwbServerApiObj);
  }

  PwbServerApiObj.SetDynDlgOutputInfo(
    PwbServerAddin.PwbApi.DlgOrdinal.Secondary,
    primaryReturnDialog);
}

return PwbServerApiObj.GetResultItem();
```

Promote to state dialog creation

The following paragraph presents a method for the Pwb_Sample_ContextPromoteDlg dialog creation:



First the PwbServerAddin.PwbApi.Dlg ReturnDialog dialog object is initialized.

```
//-------------------------------------------------------------------
// dialog for promote call
//
// INPUT:
//  PwbServerApi PwbServerApiObj - PwbServerApi contains current configuration
//  and provides API for current client call
//  IDictionary<string, string> InputObjectAttrDict - values from previous dialog
//  Item correspondingPdmObjItem - item to promote
//
// OUTPUT:
//  PwbServerAddin.PwbApi.Dlg CadReturnDialog - the created dialog
//                                              PwbServerAddin.PwbApi.Dlg
private static void ProcessForPromoteCad(
  IDictionary<string, string> InputObjectAttrDict,
  Item correspondingPdmObjItem,
  PwbServerAddin.PwbApi.Dlg ReturnDialog,
  PwbServerAddin.PwbServerApi PwbServerApiObj)
{
  ReturnDialog.BeginDialog();
```

The next promote states for the corresponding item are determined.

```
// get id of item to promote
var id = correspondingPdmObjItem.getProperty("OBID");
string pdmClass = correspondingPdmObjItem.getProperty("Class");

var actionItem = PwbServerApiObj.InnovatorObj.newItem(pdmClass)
actionItem.setID(id);
var promoteStates = actionItem.apply("getItemNextStates");
```

And the dialog elements are created.

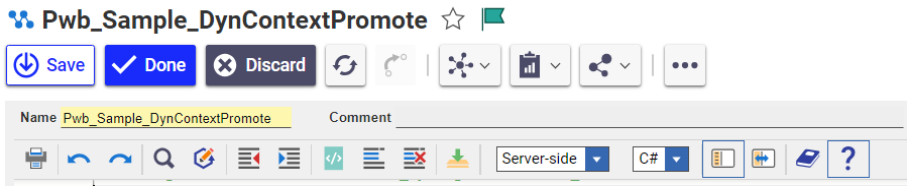
```
ReturnDialog.AddDlgAttrWidget(
  new Dictionary<string, string>
  {
  {"name", "PromoteState"},
  {"displayName", "Promote state'
  {"widgetType", "ComboBox"},
  {"mode", "update"},
  {"visibleLength", "25"},
  {"required", "true"},
  {"dataSource", "States"},
  {"updateFormIfModified", "true'
  },
  promoteValues);
ReturnDialog.AddDlgAttrWidget(
  new Dictionary<string, string>
  {
  {"name", "Comment"},
  {"displayName", "Comment"},
  {"widgetType", "MultiLineEditor"},
  {"mode", "update"},
  {"visibleLength", "15"},
  {"required", "false"}
  });

ReturnDialog.EndDialog();
```

Context method Promote to state – OK Action

Also, the OK action needs to be implemented on server. The name of the OK Action is defined within the schema file (See:

Schema file extension). It's Pwb_Sample_DynContextPromote in this case.

The OK action needs to be implemented as server side C# action.



The OK Action is called from an Aras.IOM.Item object:

- Aras object from which the context action is called on the client side

- Dialog attributes and values being returned from the dialog (Pwb_Sample_ContextPromoteDlg).

Get Type, ID and Classification from the Aras object:

```
using (var PwbServerApiObj = new PwbServerAddin.PwbServerApi(this))
{
  string PdmType = this.getProperty("Type");
  string PdmKey = this.getProperty("Id");
  string PdmClassification = this.getProperty("Classification");

  string InputDialogStr = this.getProperty("Attributes");
```

Get dialog attributes and values being returned from the dialog:

```
IDictionary<string, string> InputDialogDict = null;

if (!PwbServerApiObj.IsNullOrEmpty(InputDialogStr))
{
  InputDialogDict = PwbServerApiObj.DialogAttrsStringToDictionary(InputDialogStr
}

string InputObjectStr = this.getProperty("ObjectAttributes");

// Dialog attributes
IDictionary<string, string> InputObjectDict = null;

if (!PwbServerApiObj.IsNullOrEmpty(InputObjectStr))
{
  InputObjectDict = PwbServerApiObj.DialogAttrsStringToDictionary(InputObjectStr
}
```

Retrieve the Aras.IOM.Item corresponding to the object, for which the context action is called.

```
Item ItemObject = this.getInnovator().getItemById(PdmType, PdmKey);
```

Get dialog promote state and comment from the passed dialog attributes.

```
if (InputDialogDict != null)
{
  if (InputDialogDict.ContainsKey("PromoteState"))
  {
    var targetPromoteState =InputDialogDict["PromoteState"];
    if (!String.IsNullOrEmpty(targetPromoteState))
    {
      string comment = "";
      if (InputDialogDict.ContainsKey("Comment"))
      {
        comment = InputDialogDict["Comment"];
      }
```

Perform promote action to the promote state passed from the Pwb_Sample_ContextPromoteDlg dialog:

```
try
{
  ResultItem = ItemObject.promote(targetPromoteState, comment);
  if (ResultItem.isError())
  {
    PwbServerApiObj.AddMessageToUser(
    "Promote to "+ targetPromoteState + " failed",
    PwbServerAddin.Base.MessageType.Error);
  }
  else
  {
    ItemObject = this.getInnovator().getItemById(PdmType, PdmKey);
    PwbServerApiObj.AddMessageToUser(
    itemNumber + " promoted to "+ targetPromoteState,
    PwbServerAddin.Base.MessageType.Information);
  }
}
catch(Exception)
{
  PwbServerApiObj.AddMessageToUser(
  "Promote to "+ targetPromoteState + " failed",
  PwbServerAddin.Base.MessageType.Error);
}
```

Finally, the proceeded item is added to the PwbServerApi object and the Aras.IOM.Item result item is retrieved and returned.

```
  var itemsToReturn = new List<Item>();
  itemsToReturn.Add(ItemObject);

  PwbServerApiObj.SetItemsToReturn(itemsToReturn);

  return PwbServerApiObj.GetResultItem();
}
```