SmartPlant 3D Programming II
TSMP4002

Ramon Him
Intergraph Corporation

INTERGRAPH

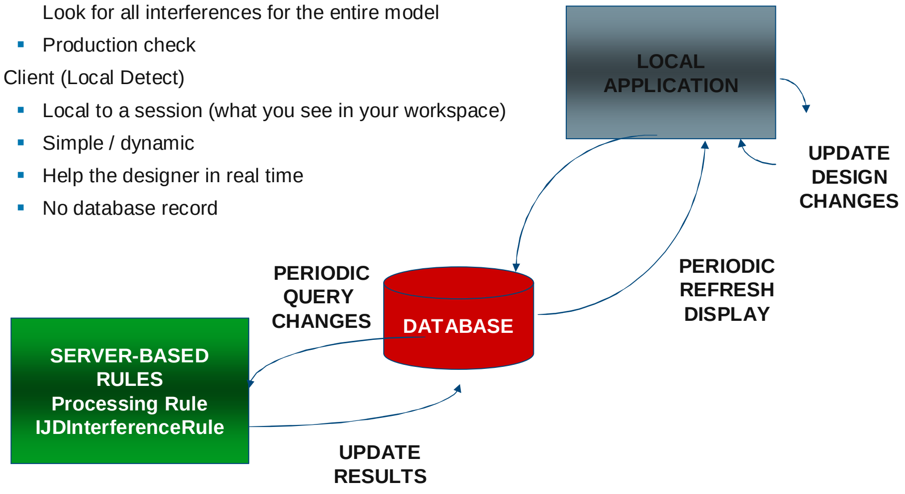
SmartPlant 3D

Agenda

- Monday - Tuesday
  - Interference checking service
    - Overview
    - Interference Entity Data Model
    - Processing Rules
      - Suppress interference between objects imported from CIMSteel
      - Assign a proper permission group for the interference object
      - Assign proper status for the interference object based on the interference type
      - Don't report Insulation and insulation interferences when 2 pipes are connected by a component
  - Tuesday - Friday
    - Equipment Data Model (Smart Occurrence)
    - Symbol Definition and Custom Assembly Definition
    - Nozzle Place Holders
    - Equipment Custom Assembly Definition using primitive shapes and control points
    - Equipment Custom Assembly Definition using equipment components
    - Equipment Custom Assembly Definition using other SP3D objects

Interference Service - Overview

- Two methods:
  - Server (Database Detect)
    - Run on a separate IFC machine as a windows service
      Look for all interferences for the entire model
    - Production check
  - Client (Local Detect)
    - Local to a session (what you see in your workspace)
    - Simple / dynamic
    - Help the designer in real time
    - No database record

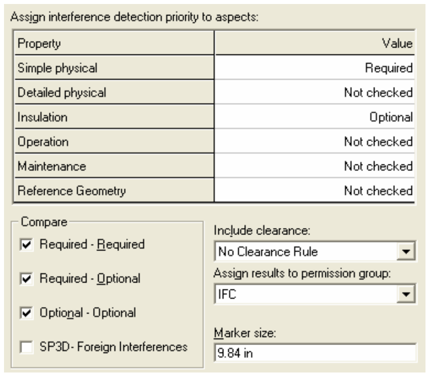


Interference Service - Overview

Database Detect	Local Detect
Runs all the time (System Admin. choice)	Works only within the current session
Minimizes impact on users and improves performance	Provides immediate graphical feedback (works in a dynamic mode)
Creates persistent interferences that are stored in the model database	Shows interferences when the pointer is idle for a brief amount of time; based on a hesitation approach
Based on administrator settings (controlled by permission groups)	Based on individual user settings
Provides feed back on how much has been checked	Checks only created and modified items
Users can visualize the interferences (persistent items)	Clears dynamic interferences after refreshing workspace

Interference Service - Overview

- Three type of checking (based on the object aspects):
  - Required (Hard)
  - Optional (Soft)
  - Not Checked
- Can process:
  - Required – Required (Hard – Hard)
  - Required – Optional (Hard – Soft)
  - Optional – Optional (Soft – Soft)
  - SP3D - Foreign Interferences
- A clearance can be used



Assign interference detection priority to aspects:

Property	Value
Simple physical	Required
Detailed physical	Not checked
Insulation	Optional
Operation	Not checked
Maintenance	Not checked
Reference Geometry	Not checked

Compare
- [✓] Required - Required
- [✓] Required - Optional
- [✓] Optional - Optional
- [ ] SP3D- Foreign Interferences

Include clearance: No Clearance Rule

Assign results to permission group: IFC

Marker size: 9.84 in

Interference Service - Overview

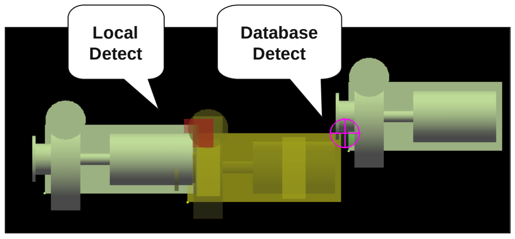
Permission Requirements:

- Only an user who is Plant Administrator (Full Control on a plant) can start and stop IFC process
- Create a permission group where results will be assigned (IFC PG)
- Grant Write permission or higher to the Plant Administrator login on IFC PG
- Administrator should have at least Read permission in all other permission groups
- Configure the IFC Windows Service to utilize the Plant Administrator Login, e.g. domain\sp3dadmin. If possible use login with password not required to change as often as corporate users'

Interference Service - Overview

Interference objects:

- Any persistent interference detected by the Database Detect process appears as a sphere
- Interference detected by the Local Detect process appears as a box



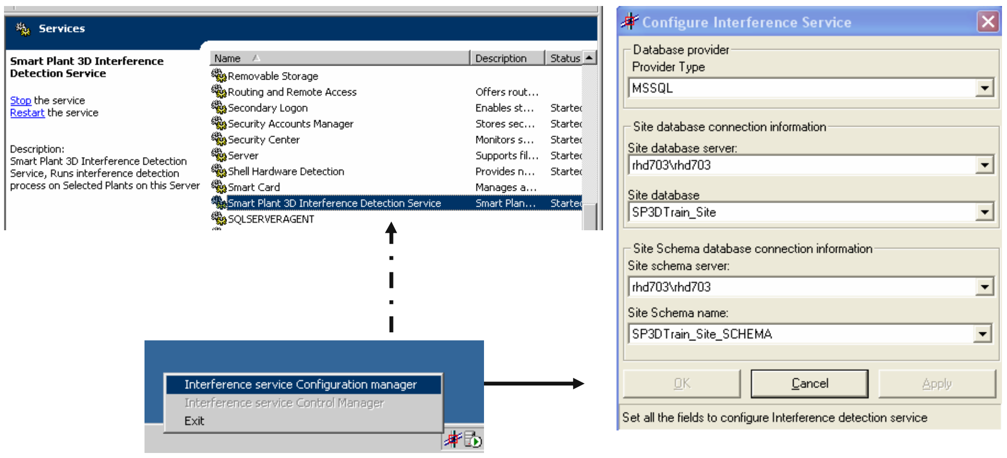
Interference Service - Overview

• The system administrator sets up the background IFC settings at the beginning of the project including configuring the Windows Service.

• If the system administrator want to change any settings for the background interference during the project execution, then all existing interferences will be deleted including any approvals and notes, and the background interference checking must be restarted. This ensures a recheck of all objects with the new settings for IFC.

Interference Service - Overview

Service Settings



Services

Smart Plant 3D Interference Detection Service

Stop the service
Restart the service

Description:
Smart Plant 3D Interference Detection Service, Runs interference detection process on Selected Plants on this Server

Name	Description	Status
Removable Storage		
Routing and Remote Access	Offers rout...	
Secondary Logon	Enables st...	Startec
Security Accounts Manager	Stores sec...	Startec
Security Center	Monitors s...	Startec
Server	Supports fil...	Startec
Shell Hardware Detection	Provides n...	Startec
Smart Card	Manages a...	
Smart Plant 3D Interference Detection Service	Smart Plan...	Startec
SQLSERVERAGENT		

Interference service Configuration manager
Interference service Control Manager
Exit

Configure Interference Service

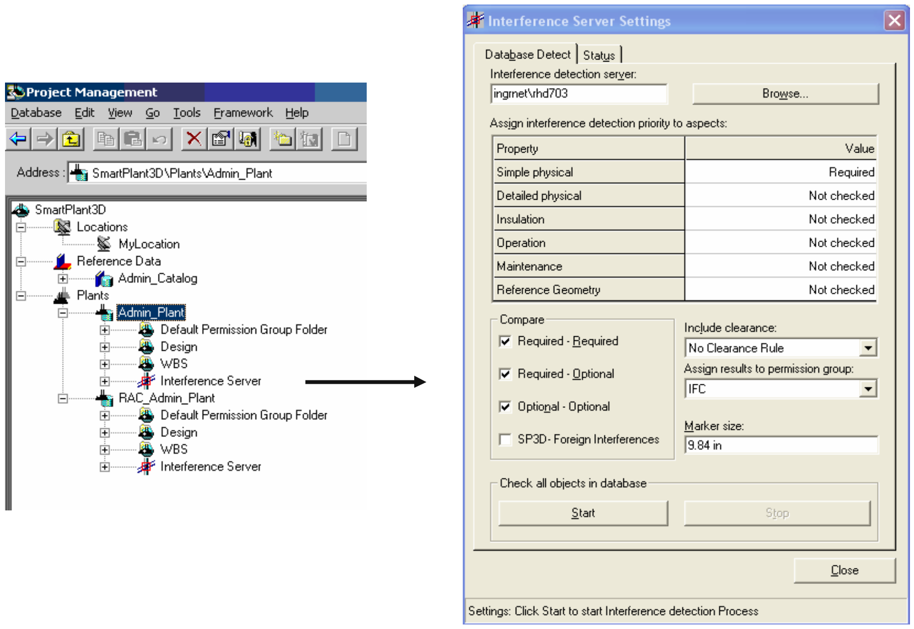
Database provider
Provider Type: MSSQL

Site database connection information
Site database server: rhd703\rhd703
Site database: SP3DTrain_Site

Site Schema database connection information
Site schema server: rhd703\rhd703
Site Schema name: SP3DTrain_Site_SCHEMA

Set all the fields to configure Interference detection service

Interference Service - Overview



Project Management

Address : SmartPlant3D\Plants\Admin_Plant

- SmartPlant3D
  - Locations
    - MyLocation
  - Reference Data
    - Admin_Catalog
  - Plants
    - Admin_Plant
      - Default Permission Group Folder
      - Design
      - WBS
      - Interference Server
    - RAC_Admin_Plant
      - Default Permission Group Folder
      - Design
      - WBS
      - Interference Server

Interference Server Settings

Interference detection server: ingrnet\rhd703

Assign interference detection priority to aspects:

Property	Value
Simple physical	Required
Detailed physical	Not checked
Insulation	Not checked
Operation	Not checked
Maintenance	Not checked
Reference Geometry	Not checked

Compare
- [✓] Required - Required
- [✓] Required - Optional
- [✓] Optional - Optional
- [ ] SP3D- Foreign Interferences

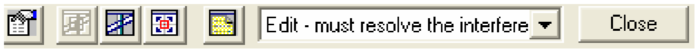
Include clearance: No Clearance Rule
Assign results to permission group: IFC
Marker size: 9.84 in

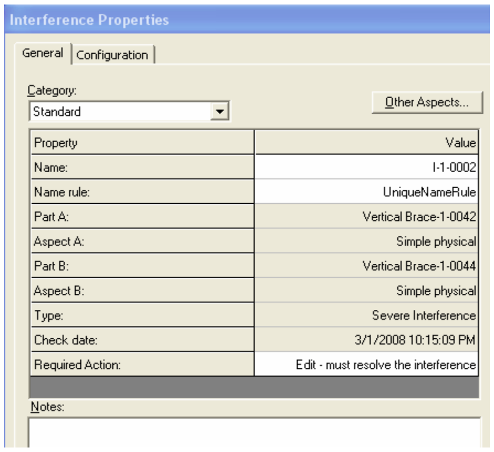
Check all objects in database

Settings: Click Start to start Interference detection Process

Interference Service - Overview

- Interference Check is available in all Task (Tools → Check Interference)
- Very simple and intuitive GUI
- Ribbon bar includes:
  - Settings
  - Visualization
  - Review & approval



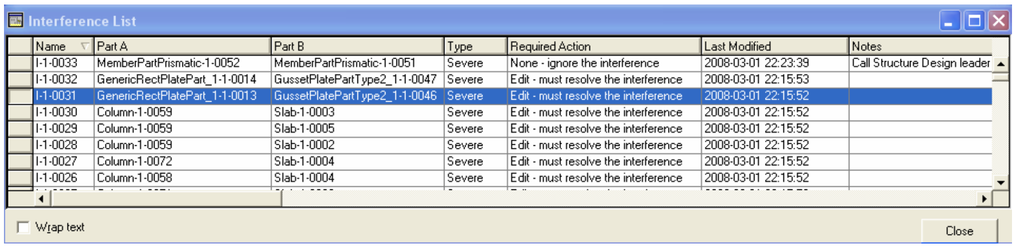


Category: Standard

Property	Value
Name:	I-1-0002
Name rule:	UniqueNameRule
Part A:	Vertical Brace-1-0042
Aspect A:	Simple physical
Part B:	Vertical Brace-1-0044
Aspect B:	Simple physical
Type:	Severe Interference
Check date:	3/1/2008 10:15:09 PM
Required Action:	Edit - must resolve the interference

Notes:

Interference Service - Overview



Name	Part A	Part B	Type	Required Action	Last Modified	Notes
I-1-0033	MemberPartPrismatic-1-0052	MemberPartPrismatic-1-0051	Severe	None - ignore the interference	2008-03-01 22:23:39	Call Structure Design leader
I-1-0032	GenericRectPlatePart_1-1-0014	GussetPlatePartType2_1-1-0047	Severe	Edit - must resolve the interference	2008-03-01 22:15:53	
I-1-0031	GenericRectPlatePart_1-1-0013	GussetPlatePartType2_1-1-0046	Severe	Edit - must resolve the interference	2008-03-01 22:15:52	
I-1-0030	Column-1-0059	Slab-1-0003	Severe	Edit - must resolve the interference	2008-03-01 22:15:52	
I-1-0029	Column-1-0059	Slab-1-0005	Severe	Edit - must resolve the interference	2008-03-01 22:15:52	
I-1-0028	Column-1-0059	Slab-1-0002	Severe	Edit - must resolve the interference	2008-03-01 22:15:52	
I-1-0027	Column-1-0072	Slab-1-0004	Severe	Edit - must resolve the interference	2008-03-01 22:15:52	
I-1-0026	Column-1-0058	Slab-1-0004	Severe	Edit - must resolve the interference	2008-03-01 22:15:52	

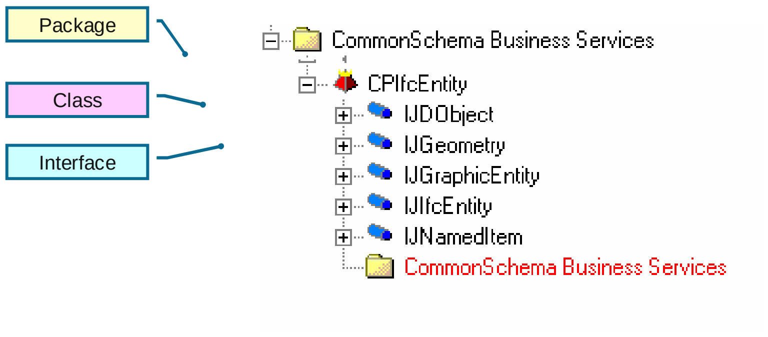
☐ Wrap text

Interference List Dialog Box

- Resizable
- Sort the listed interferences by Columns
- Highlight an interference in the workspace
- Double Click Interference entry to access property page for the interference
- Copy/Paste functionality from list to Excel

IFC Entity Data Model

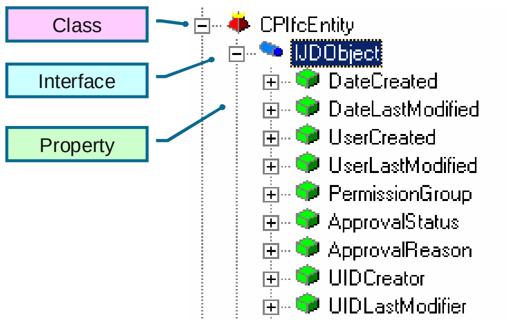
CPIfcEntity



IFC Entity Data Model

Generic Object interfaces

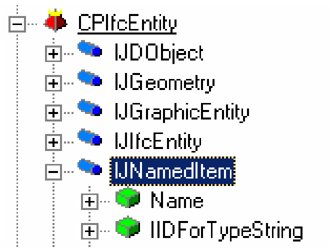
IJDObject interface is a required interface for almost all objects. This interface provides access to the permission group for the object, the status, name of the user who created and last modified the object and the date and time for creation and last modification.



IFC Entity Data Model

Generic Object interfaces

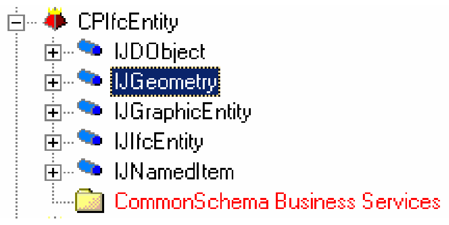
IJNamedItem interface provides the name property and a type string for all named objects. This interface is used by any component that needs the name of an object as well as by components that need to display a simple type string for an object.



IFC Entity Data Model

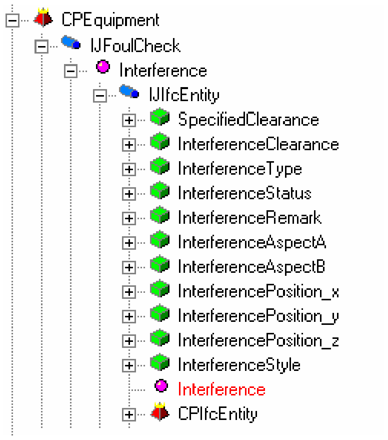
Geometry interfaces

- IJGeometry interface is required if the BO has geometry
- IJGraphicEntity interface is required if the BO is displayable



IJFoulCheck

- IJFoulCheck is the basic interface used by Interference Engine to determine if the object implementing it collides with other objects that implement it.

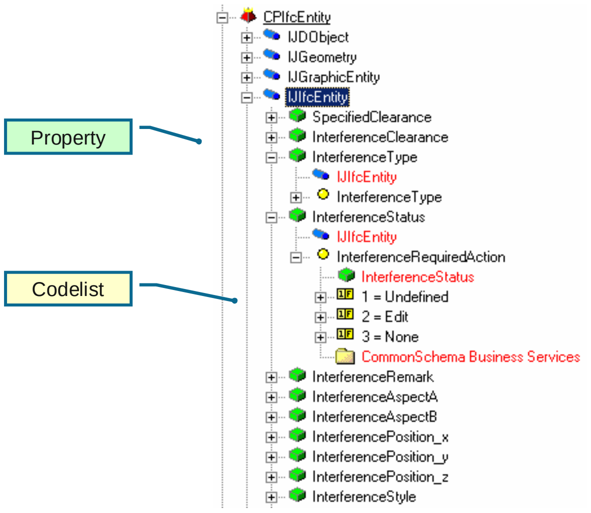


```
CPEquipment
  IJFoulCheck
    Interference
      IJIfcEntity
        SpecifiedClearance
        InterferenceClearance
        InterferenceType
        InterferenceStatus
        InterferenceRemark
        InterferenceAspectA
        InterferenceAspectB
        InterferencePosition_x
        InterferencePosition_y
        InterferencePosition_z
        InterferenceStyle
        Interference
        CPIfcEntity
```

IFC Entity Data Model

Interference Interface

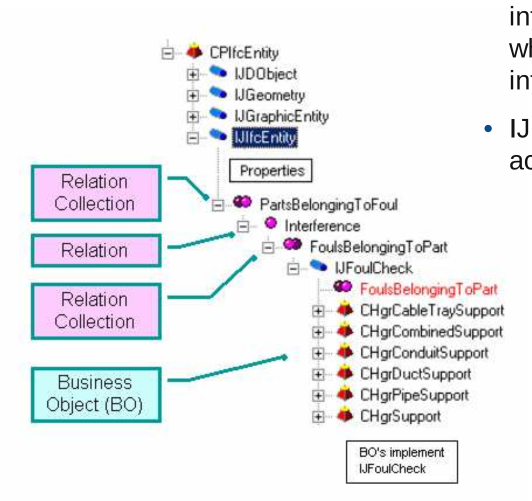
- IJIfcEntity interface keeps track of all the interference properties created by the object.



```
CPIfcEntity
  IJDObject
  IJGeometry
  IJGraphicEntity
  IJIfcEntity
    SpecifiedClearance
    InterferenceClearance
    InterferenceType
      IJIfcEntity
      InterferenceType
    InterferenceStatus
      IJIfcEntity
      InterferenceRequiredAction
        InterferenceStatus
        1 = Undefined
        2 = Edit
        3 = None
        CommonSchema Business Services
    InterferenceRemark
    InterferenceAspectA
    InterferenceAspectB
    InterferencePosition_x
    InterferencePosition_y
    InterferencePosition_z
    InterferenceStyle
```

IFC Entity Data Model

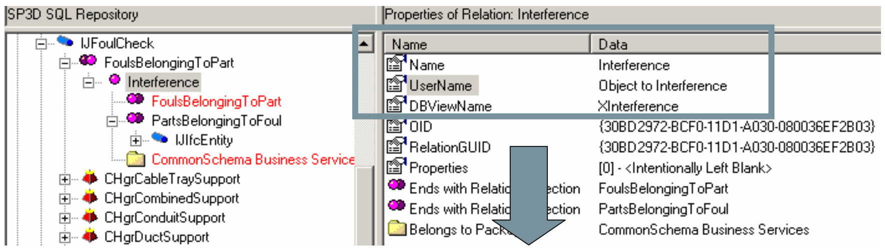
Interference Relationship



- BO's implement the IJFoulCheck interface which is used to identify which objects should be checked for interferences with other object.
- IJFoulCheck has a relation with the actual interference object's interface.

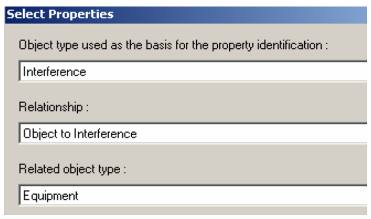
IFC Entity Data Model

Interference Relationship



Properties of Relation: Interference

Name	Data
Name	Interference
UserName	Object to Interference
DBViewName	XInterference
OID	{30BD2972-BCF0-11D1-A030-080036EF2B03}
RelationGUID	{30BD2972-BCF0-11D1-A030-080036EF2B03}
Properties	[0] - <Intentionally Left Blank>
Ends with Relatio ection	FoulsBelongingToPart
Ends with Relatio ection	PartsBelongingToFoul
Belongs to Pack	CommonSchema Business Services



Select Properties

Object type used as the basis for the property identification : Interference

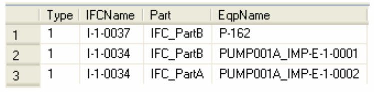
Relationship : Object to Interference

Related object type : Equipment

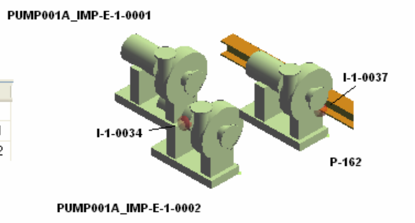
IFC Entity Data Model

Example: List all persistent interferences created by equipment objects

```
Select x1.FoulType as Type, x2.ItemName as IFCName,
x3.RelationName as Part, x5.ItemName as EqpName
 from JIfcEntity x1
JOIN JNamedItem x2 on x2.oid = x1.oid
JOIN XInterference x3 on x3.oidorigin = x1.oid
JOIN JEquipment x4 on x4.oid = x3.oidDestination
JOIN JNamedItem x5 on x5.oid = x4.oid
```

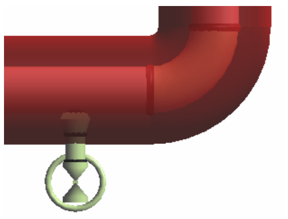


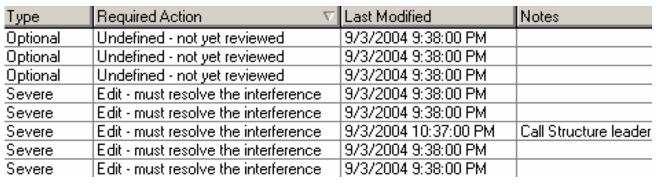
	Type	IFCName	Part	EqpName
1	1	I-1-0037	IFC_PartB	P-162
2	1	I-1-0034	IFC_PartB	PUMP001A_IMP-E-1-0001
3	1	I-1-0034	IFC_PartA	PUMP001A_IMP-E-1-0002



Interference Processing Rule

- Customized component that allows users to define rules that automatically set properties on the Interference objects during creation and modification or to decide whether the Interference objects need to be persisted in the Database.

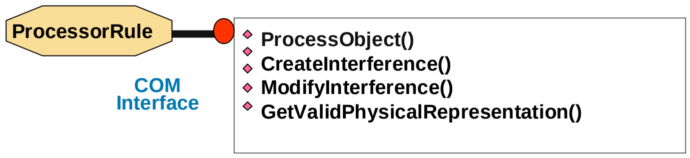




Type	Required Action	Last Modified	Notes
Optional	Undefined - not yet reviewed	9/3/2004 9:38:00 PM	
Optional	Undefined - not yet reviewed	9/3/2004 9:38:00 PM	
Optional	Undefined - not yet reviewed	9/3/2004 9:38:00 PM	
Severe	Edit - must resolve the interference	9/3/2004 9:38:00 PM	
Severe	Edit - must resolve the interference	9/3/2004 9:38:00 PM	
Severe	Edit - must resolve the interference	9/3/2004 10:37:00 PM	Call Structure leader
Severe	Edit - must resolve the interference	9/3/2004 9:38:00 PM	
Severe	Edit - must resolve the interference	9/3/2004 9:38:00 PM	

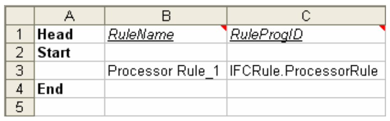
Interference Processing Rule

- The Interference Processing rule is a Visual Basic component
- Implements
  - IJDInterferencePrePrcsrRule
  - IJDInterferenceRule
- Processing:
  - Call at Creation - Create Interference method
  - Call at Modification of existing interference - Modify Interference method
  - Aspect Control - Get valid physical representation method

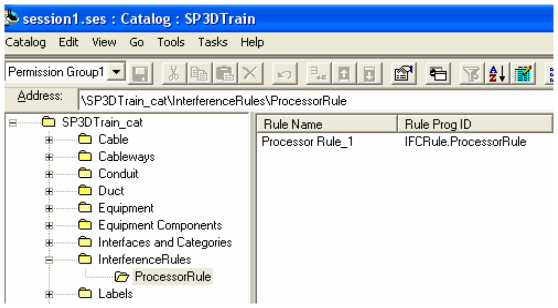


Interference Processing Rule

- The Interference processing rule program is located at <Install Directory>\Programming\ExampleCode\Rules\InterferenceRules



	A	B	C
1	**Head**	*RuleName*	*RuleProgID*
2	**Start**		
3		Processor Rule_1	IFCRule.ProcessorRule
4	**End**		
5			



session1.ses : Catalog : SP3DTrain

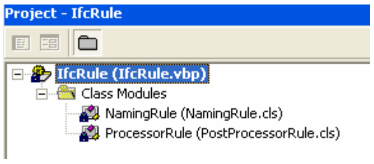
Address: \SP3DTrain_cat\InterferenceRules\ProcessorRule

Rule Name	Rule Prog ID
Processor Rule_1	IFCRule.ProcessorRule

Interference Processing Rule

Processor Rule Module includes:

- Private Sub Class_Initialize()
- Private Sub Class_Terminate()


Project - IfcRule

IfcRule (IfcRule.vbp)
Class Modules
NamingRule (NamingRule.cls)
ProcessorRule (PostProcessorRule.cls)

- Function IJDInterferencePrePrcsrRule_ProcessObject()
- Function IJDInterferenceRule_CreateInterference()
- Sub IJDInterferenceRule_ModifyInterference()
- Function IJDInterferenceRule_GetValidPhysicalRepresentation()

Interference Pre Processing Rule

Pre Processor Rule Module

- This Rule will be called by interference service after it updates the range of the object. The object under processing and the Object type are sent as arguments and this rule returns True or False value. Based on this the object is either ignored or considered further.

```
Private Function IJDInterferencePrePrcsrRule_ProcessObject(ByVal pObject As Object, ByVal
    strObjectType As String) As Boolean
On Error GoTo ErrorHandler
    IJDInterferencePrePrcsrRule_ProcessObject = True

    ' Insert your code

Exit Function
ErrorHandler:
    Err.Clear
    IJDInterferencePrePrcsrRule_ProcessObject = True
End Function
```

Interference Pre Processing Rule

Example:

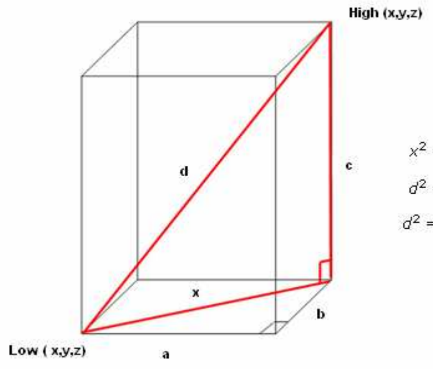
```
Private Function IJDInterferencePrePrcsrRule_ProcessObject(ByVal pObject As Object, ByVal
    strObjectType As String) As Boolean
On Error GoTo ErrorHandler
    IJDInterferencePrePrcsrRule_ProcessObject = True
```

```
'Ignore objects based on the ObjectType.

If strObjectType Like "Plate System" Then
    IJDInterferencePrePrcsrRule_ProcessObject = False
End If

Exit Function
ErrorHandler:
    Err.Clear
    IJDInterferencePrePrcsrRule_ProcessObject = True
End Function
```

Interference Pre Processing Rule

Example:

Ignore objects based on their range extents



$x^2 = a^2 + b^2$

$d^2 = x^2 + c^2$

$d^2 = a^2 + b^2 + c^2$

diagonal length^2 = (high.x - low.x)^2 + (high.y - low.y)^2 - (high.z = low.z)^2

Interference Pre Processing Rule

Example:

Ignore objects based on their range extents

```
Const rangeDiagonalLenLimit = 100# ' meters

If bRangeCheckEnabled = True Then
    Dim rng As GBox
    Dim diagonalLength As Double
    Dim pObjectRange As IJRangeAlias
    Set pObjectRange = pObject
    rng = pObjectRange.GetRange ' results are in meters
    diagonalLength = Sqr(((rng.m_high.x - rng.m_low.x) ^ 2) + ((rng.m_high.y - rng.m_low.y) ^ 2) _
                    + ((rng.m_high.z - rng.m_low.z) ^ 2))
    ' if the diagonal length of the object is larger than limit, skip processing interferences against this object
    If diagonalLength > rangeDiagonalLenLimit Then
        IJDInterferencePrePrcsrRule_ProcessObject = False
        WriteToDebugLog "IJDInterferencePrePrcsrRul::ProcessObject,objectType," & strObjectType & ",objectName," & _
                        Chr(34) & name & Chr(34) & ",rangeDiagonal," & diagonalLength & _
                        "," & Chr(34) & "not processed due to rangeDiagonalLength larger than " & _
                        rangeDiagonalLenLimit & " meters" & Chr(34)
    End If
End If
```

Interference Post Processing Rule

Processor Rule Module includes:

Four example rules:

- Suppress interference between objects imported from CIMSteel
- Assign a proper permission group for the interference object based on the colliding object’s ranking
- Assign proper status for the interference object based on the interference type
- Don’t report Insulation and insulation interferences when 2 pipes are connected by a component

Additional routines to support the example rules

Private Sub AssignIFCPermissionGroup()

Private Function GetPermissionGroupIndex()

Private Function IsConnectedbyIntermediate()

Private Function ConvertPGNameToNumber()

Private Function GetAspectName()

Private Function GetAspectCode()

Private Function Distance()

Private Function GetObjectIdentifyer()

Private Function AreBothImportedObjects()

Private Function GetResourceManager()

Interference Post Processing Rule

Create Interference method

This method gets triggered just after Interference process detects an Interference and just before persisting the same to the Database.

- Decide whether the Interference should be persisted or not in the Database.
- Initialize the interference properties automatically
  - Permission Group
  - Remarks
  - Status (Action Required)

Processing Rule - Creation

Create Interference method

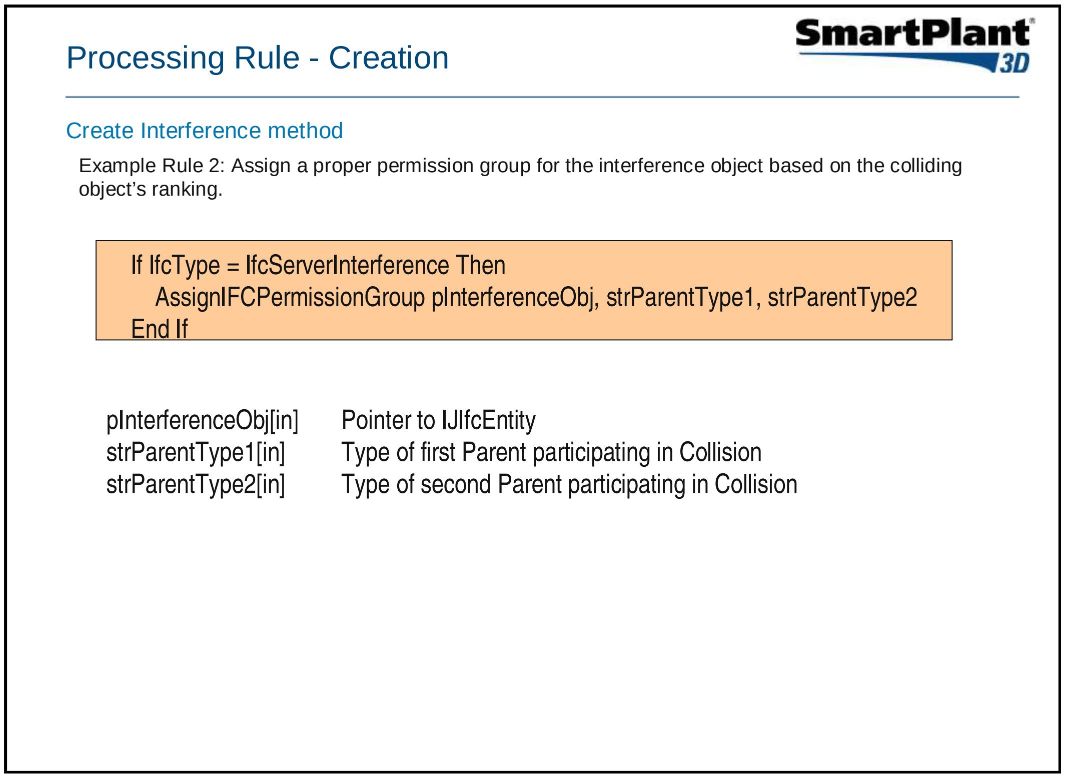
Example Rule 1: Suppress interference between objects imported from CIMSteel or from Tribon

```
If (AreBothImportedObjects(pParent1, pParent2)) Then
      IJDInterferenceRule_CreateInterference = False
      Exit Function
   End If

pParent1[in]       First Parent participating in Collision
pParent2[in]       Second Parent participating in Collision
```

Imported Tribon objects support "IJImportedStructureItem"and "IJDImpPlate" interface

Imported CIMSteel (CIS/2) objects support “IJStructEDIData” interface

Tribon is a shipbuilding CAD/CAM program



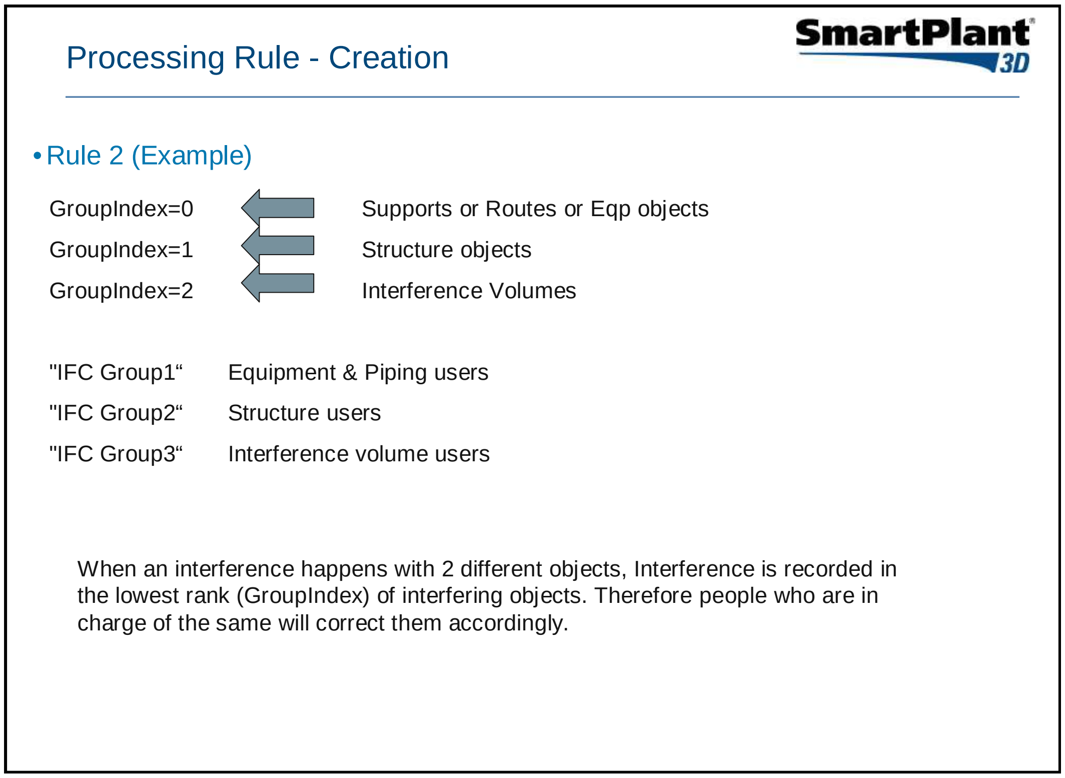
Processing Rule - Creation

Create Interference method

Example Rule 2: Assign a proper permission group for the interference object based on the colliding object's ranking.

```
If IfcType = IfcServerInterference Then
   AssignIFCPermissionGroup pInterferenceObj, strParentType1, strParentType2
End If
```

pInterferenceObj[in]	Pointer to IJIfcEntity
strParentType1[in]	Type of first Parent participating in Collision
strParentType2[in]	Type of second Parent participating in Collision



Processing Rule - Creation

- Rule 2 (Example)

GroupIndex=0	←	Supports or Routes or Eqp objects
GroupIndex=1	←	Structure objects
GroupIndex=2	←	Interference Volumes

"IFC Group1"	Equipment & Piping users
"IFC Group2"	Structure users
"IFC Group3"	Interference volume users

When an interference happens with 2 different objects, Interference is recorded in the lowest rank (GroupIndex) of interfering objects. Therefore people who are in charge of the same will correct them accordingly.

Processing Rule - Creation

- Rule 2 (Example)

Class_Initialize()

```
ReDim m_strPermissionGroups(3) As String
   m_strPermissionGroups(0) = "IFC Group1“     (Supports + Routes + Eqp objects)
   m_strPermissionGroups(1) = "IFC Group2“     (Structure objects)
   m_strPermissionGroups(2) = "IFC Group3“     (Interference Volumes)
```

Processing Rule - Creation

Function GetPermissionGroupIndex()

```
Select Case (strParentType)
  Case "Pipe Supports", "Cable Tray Supports", "Duct Supports"
      GetPermissionGroupIndex = 0
  Case "Conduits“, "Cable Tray Components", "Cableway Straight", "Cable Trays",
           "Pipes", "Piping Welds", "Piping Components", "Piping Instruments", _
           "Piping Specialty Items“, "Equipment“, "Cableway Turn“, "Cableway Along Leg“, _
           "HVAC Components", "Ducts"
      GetPermissionGroupIndex = 0
  Case "Member Part Linear", "Slab", "Footing", "Stairs", "Ladders", "Handrails“ _
           "Member Part Curve", "Equipment Foundation"
      GetPermissionGroupIndex = 1
  Case "Interference Volumes"
      GetPermissionGroupIndex = 2
   Case Default   GetPermissionGroupIndex = -1
   End Select
```

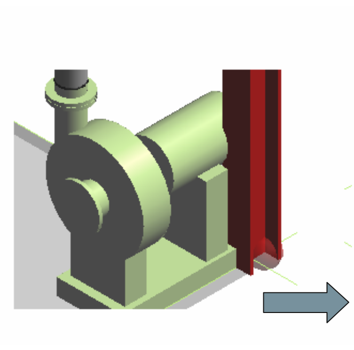
Processing Rule - Creation

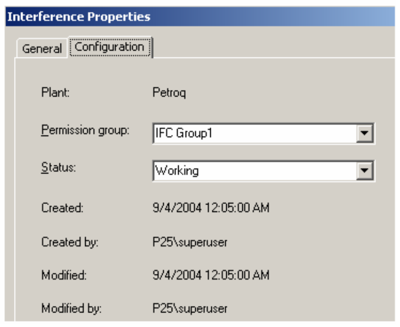
```
Private Sub AssignIFCPermissionGroup(ByVal pNewInterference As IJIfcEntity,
        ByVal strParentType1 As String, ByVal , strParentType2 As String)
PGIndexObj1 = GetPermissionGroupIndex(strParentType1)
PGIndexObj2 = GetPermissionGroupIndex(strParentType2)
If PGIndexObj1 < PGIndexObj2 Then
  strInterferencePG = m_strPermissionGroups(PGIndexObj1)
Else
  strInterferencePG = m_strPermissionGroups(PGIndexObj2)
End If
Set pObject = pNewInterference
nPGNumber = ConvertPGNameToNumber(strInterferencePG)
pObject.PermissionGroup = nPGNumber
Set pObject = Nothing
```

Processing Rule - Creation

- Rule 2(Example)

Equipment collides with Member Part Prismatic





Plant:	Petroq
Permission group:	IFC Group1
Status:	Working
Created:	9/4/2004 12:05:00 AM
Created by:	P25\superuser
Modified:	9/4/2004 12:05:00 AM
Modified by:	P25\superuser

Processing Rule - Creation

Create Interference method

- Example Rule 3: Assign proper status for the interference object based on the interference type.

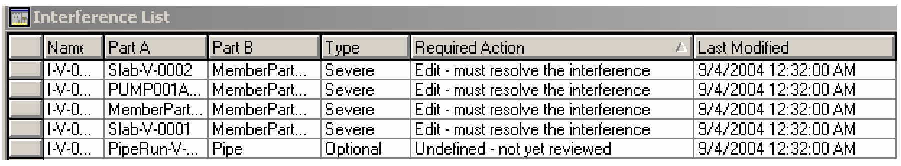
Type	Action Required	Status
Severe	Edit – Must resolved the interference	fsUnacceptableFoul
Optional	Undefined – Not yet reviewed	fsUndefined
Clearance	None – Ignore the interference	fsAcceptableFoul

```
If IfcType = IfcServerInterference Then
    If pInterferenceObj.InterferenceType = ftSevereFoul Then
        pInterferenceObj.InterferenceStatus = fsUnacceptableFoul
    ElseIf pInterferenceObj.InterferenceType = ftOptionalFoul Then
        pInterferenceObj.InterferenceStatus = fsUndefinedFoul
    ElseIf pInterferenceObj.InterferenceType = ftClearanceFoul Then
        pInterferenceObj.InterferenceStatus = fsAcceptableFoul
    Else 'It is a Bad part foul
        pInterferenceObj.InterferenceStatus = fsUnacceptableFoul
    End If
End If
```

Processing Rule - Creation

- Rule 3 (Example)

Type	Action Required	Status
Severe	Edit – Must resolved the interference	fsUnacceptableFoul
Optional	Undefined – Not yet reviewed	fsUndefined
Clearance	None – Ignore the interference	fsAcceptableFoul


Interference List

Name	Part A	Part B	Type	Required Action	Last Modified
I-V-0...	Slab-V-0002	MemberPart...	Severe	Edit - must resolve the interference	9/4/2004 12:32:00 AM
I-V-0...	PUMP001A...	MemberPart...	Severe	Edit - must resolve the interference	9/4/2004 12:32:00 AM
I-V-0...	MemberPart...	MemberPart...	Severe	Edit - must resolve the interference	9/4/2004 12:32:00 AM
I-V-0...	Slab-V-0001	MemberPart...	Severe	Edit - must resolve the interference	9/4/2004 12:32:00 AM
I-V-0...	PipeRun-V-...	Pipe	Optional	Undefined - not yet reviewed	9/4/2004 12:32:00 AM

Processing Rule - Modification

Modification Interference method

Rule 1: Default implementation just resetting the status and add a note.

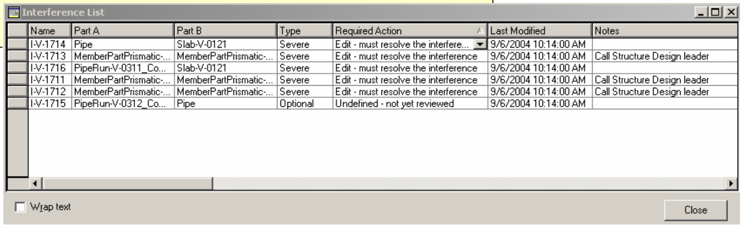
```
Dim strAppendNotes As String
Dim strexistingNotes As String
  strAppendNotes = " Changes did not remove the Interference"
  If IfcType = IfcServerInterference Then
    Select Case (pOldInterference.InterferenceType)
      Case ftSevereFoul
        pOldInterference.InterferenceStatus = fsUnacceptableFoul
      Case ftOptionalFoul
        pOldInterference.InterferenceStatus = fsUndefinedFoul
      Case ftClearanceFoul
        pOldInterference.InterferenceStatus = fsAcceptableFoul
      Case ftBadPartFoul
        pOldInterference.InterferenceStatus = fsUndefinedFoul
    End Select
    If Not (strAppendNotes Like "") Then
      strexistingNotes = pOldInterference.InterferenceRemark
      strexistingNotes = strexistingNotes + strAppendNotes
      pOldInterference.InterferenceRemark = strexistingNotes
    End If
  End If
```

Processing Rule - Creation

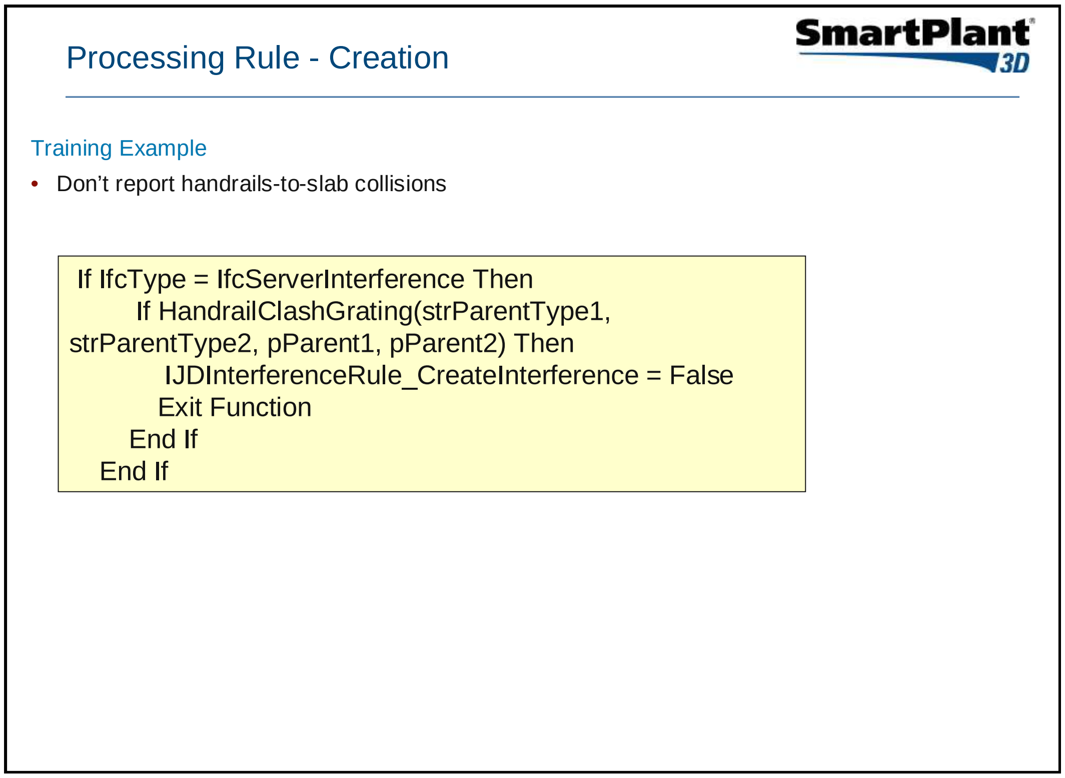
Training Example

- Add notes for the interference object based on the object type

```
 Dim strNotes As String
   strNotes = ""
   If IfcType = IfcServerInterference Then
       If strParentType1 Like "Member Part Linear"
And _
           strParentType2 Like "Member Part Linear"
Then
           strNotes = "Call Structure Design leader"
       pInterferenceObj.InterferenceRemark = strNotes
       End If
   End If
```



Name	Part A	Part B	Type	Required Action	Last Modified	Notes
I-V-1714	Pipe	Slab-V-0121	Severe	Edit - must resolve the interfere...	9/6/2004 10:14:00 AM	
I-V-1713	MemberPartPrismatic-...	MemberPartPrismatic-...	Severe	Edit - must resolve the interference	9/6/2004 10:14:00 AM	Call Structure Design leader
I-V-1716	PipeRun-V-0311_Co...	Slab-V-0121	Severe	Edit - must resolve the interference	9/6/2004 10:14:00 AM	
I-V-1711	MemberPartPrismatic-...	MemberPartPrismatic-...	Severe	Edit - must resolve the interference	9/6/2004 10:14:00 AM	Call Structure Design leader
I-V-1712	MemberPartPrismatic-...	MemberPartPrismatic-...	Severe	Edit - must resolve the interference	9/6/2004 10:14:00 AM	Call Structure Design leader
I-V-1715	PipeRun-V-0312_Co...	Pipe	Optional	Undefined - not yet reviewed	9/6/2004 10:14:00 AM	

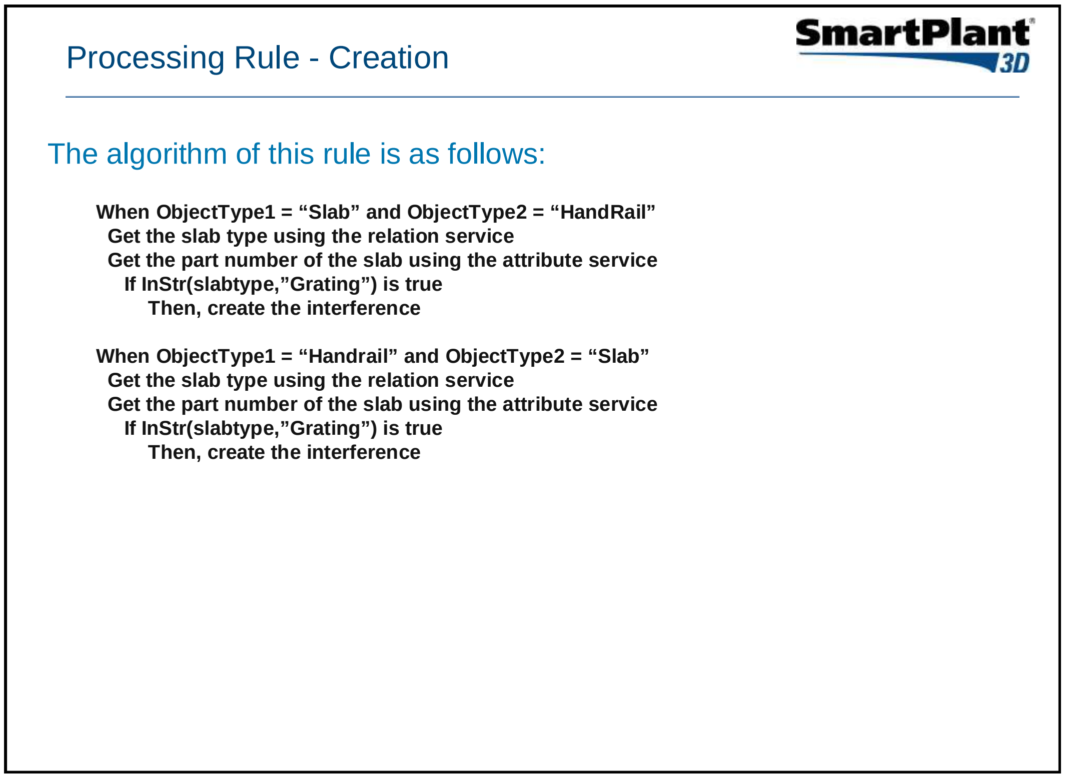


Processing Rule - Creation

Training Example

- Don’t report handrails-to-slab collisions

```
 If IfcType = IfcServerInterference Then
        If HandrailClashGrating(strParentType1,
strParentType2, pParent1, pParent2) Then
            IJDInterferenceRule_CreateInterference = False
           Exit Function
       End If
   End If
```



Processing Rule - Creation

The algorithm of this rule is as follows:

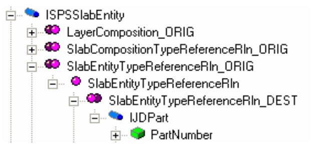
```
When ObjectType1 = “Slab” and ObjectType2 = “HandRail”
  Get the slab type using the relation service
  Get the part number of the slab using the attribute service
    If InStr(slabtype,”Grating”) is true
       Then, create the interference

When ObjectType1 = “Handrail” and ObjectType2 = “Slab”
  Get the slab type using the relation service
  Get the part number of the slab using the attribute service
    If InStr(slabtype,”Grating”) is true
       Then, create the interference
```

Processing Rule - Creation

Relationship

The following picture shows an example. This defines a relationship of type SlabEntityReferenceRln between the interface ISPSSlabEntity, origin of the relationship, and the interface IJDPart for a destination.

- The interface origin of the relationship
- The interface destination of the relationship
- A relationship type


ISPSSlabEntity
LayerComposition_ORIG
SlabCompositionTypeReferenceRln_ORIG
SlabEntityTypeReferenceRln_ORIG
SlabEntityTypeReferenceRln
SlabEntityTypeReferenceRln_DEST
IJDPart
PartNumber

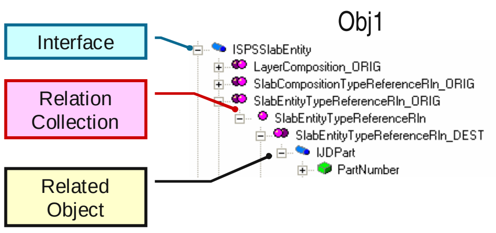
- Use the relation helper to access the part of the colliding object
- Use the Attribute Helper to get the part number of the slab

Processing Rule - Creation

Relation Helper

CollectionRelations(interfaceID, collectionName As String) As Object

```
Set oRelationHelper = Obj1
Dim oAttrbs As IJDAttributes
Set oCollection =
oRelationHelper.CollectionRelations("ISPSSlabEntity",“SlabEntityTypeReferenceRln_ORIG")
Set oAttrbs = oCollection.Item(1)
slabtype = oAttrbs.CollectionOfAttributes("IJDPart").Item("PartNumber").Value
```


Obj1
Interface
Relation Collection
Related Object
ISPSSlabEntity
LayerComposition_ORIG
SlabCompositionTypeReferenceRln_ORIG
SlabEntityTypeReferenceRln_ORIG
SlabEntityTypeReferenceRln
SlabEntityTypeReferenceRln_DEST
IJDPart
PartNumber

Processing Rule - Creation

Training Example

- Don’t report interferences when colliding objects belong to a Test Permission Group

```
If IfcType = IfcServerInterference Then
        If objectBelongToPG("TESTPG", pParent1, pParent2) Then
            IJDInterferenceRule_CreateInterference = False
           Exit Function
       End If
   End If
```

Processing Rule - Creation

objectBelongToPG()

```
Private Function objectBelongToPG(UPPERCASE_pgName_substring As String _
                                  , ByVal pParent1 As Object _
                                  , ByVal pParent2 As Object _
                                  ) As Boolean
On Error GoTo ErrHndlr
objectBelongToPG = False

   Dim pNum1 As Long, pNum2 As Long
   Dim strPgName1 As String, strPgName2 As String
   Dim pObject1 As IJDObject
   Dim pObject2 As IJDObject
   Set pObject1 = pParent1
   Set pObject2 = pParent2
   pNum1 = pObject1.PermissionGroup
   pNum2 = pObject2.PermissionGroup
   strPgName1 = ConvertPGNumberToName(pNum1)
   strPgName2 = ConvertPGNumberToName(pNum2)
   If InStr(UCase(strPgName1), UPPERCASE_pgName_substring) > 0 Or  _
  InStr(UCase(strPgName2), UPPERCASE_pgName_substring) > 0 Then
      objectBelongToPG = True
      Exit Function
   End If
```

Processing Rule - Creation

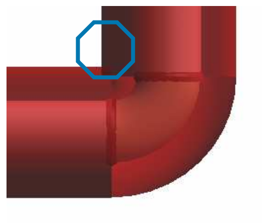
ConvertPGNumberToName()

```
Private Function ConvertPGNumberToName(ByVal PGnum As Long) As String
On Error GoTo ErrHndlr
    Dim PGID, count As Long
    Dim pgName As String
    Dim i As Long
    Dim oMidCtx As IJMiddleContext
    Dim oDBTypeConfig As IJDBTypeConfiguration
    Dim oDataBaseConfig As IJDataBaseConfiguration
    Dim oACConfig As IJAccessControlConfiguration
    Dim oAccessControl As IJAccessControl
    ConvertPGNumberToName = ""
    Set oDBTypeConfig = New DBTypeConfiguration
    Set oDataBaseConfig = New DataBaseConfiguration
    Set oACConfig = New AccessControlConfiguration
    Set oMidCtx = New GSCADMiddleContext
    oMidCtx.GetConfigurationTablesFromMiddle oDBTypeConfig, oDataBaseConfig, oACConfig
    Set oAccessControl = oACConfig.AccessControl
    count = oACConfig.NumberConditionIDs
    For i = 1 To count
        oACConfig.GetConditionIDByIndex i, pgName, PGID
        If PGID = PGnum Then
            ConvertPGNumberToName = pgName
            Exit Function
        End If
    Next i
```

Processing Rule - Creation

Create Interference method

Example Rule 4: Don't report Insulation and insulation interferences when 2 pipes are connected by a component.



```
When ObjectType1 = "Pipe part"
If Aspect1 = Aspect2 = Insulation. Then do the following logic
    otherwise break and create an Interference.
    Get connected Objects for Object1. --- List1.
    For each object in List1
        Get Connected objects, -- List2
        For each Object in List2
            If  Object is Object 2 then
                Both objects are connected through a middle object.
                Break the loop
            Endif
        Loop for next object in List 2
    Loop for next object in List 1
If the Objects are not connected through an intermediate object
then
    create the interference otherwise not.
```

Processing Rule - Creation

- Rule 4 (Example)

```
If (strParentType1 <> strParentType2) Then
    Exit Function
  Else
    If Not (strParentType1 Like "Pipes") Then
      Exit Function
    End If
    If pInterferenceObj.InterferenceAspectA = pInterferenceObj.InterferenceAspectB Then
      If GetAspectName(pInterferenceObj.InterferenceAspectA) Like "Insulation" Then
        If IsConnectedbyIntermediate(pParent1, pParent2) = True Then
          IJDInterferenceRule_CreateInterference = False
          Exit Function
        Else IJDInterferenceRule_CreateInterference = True
          Exit Function
        End If
      End If
    End If
End If
```

Processing Rule - Modification

Modification Interference method

- This rule gets triggered when Interference detection process is trying to modify an interference, because of the modification of the parts participating in the collision.

- Decide what to do when the parent(s) of an existing interference have been modified. Example:
  - Change status
  - Add Notes

Interference Post processing Rule

Get Valid Physical Representation Rule method

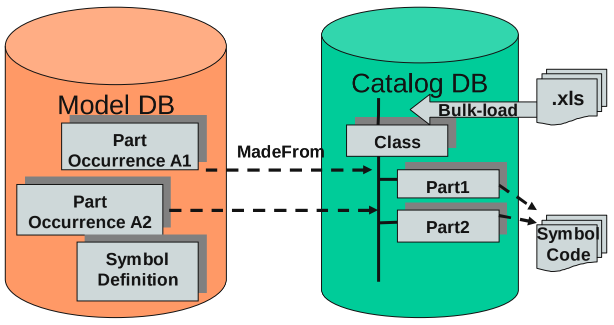
- If Interference process finds an object which has 2 physical representations, this rule will choose the proper representation to be used for that type of object.

```
Private Function IJDInterferenceRule_GetValidPhysicalRepresentation(ByVal strObjectType
As String, ByVal SupportedPhysicalReps As Long) As Long
On Error GoTo ErrorHandler
    'Kept a check for Pipe Supports as Detailed physical aspect Geometry is not good.
    If ((strObjectType Like "Pipe Supports") Or (strObjectType Like "Duct Supports") Or
(strObjectType Like "Cable Tray Supports")) Then
        IJDInterferenceRule_GetValidPhysicalRepresentation = GetAspectCode("Simple
physical")
    Else
        IJDInterferenceRule_GetValidPhysicalRepresentation = GetAspectCode("Detailed
physical")
    End If
Exit Function
```

Overview

Part Occurrence Model (Part Class/Part Occurrence)

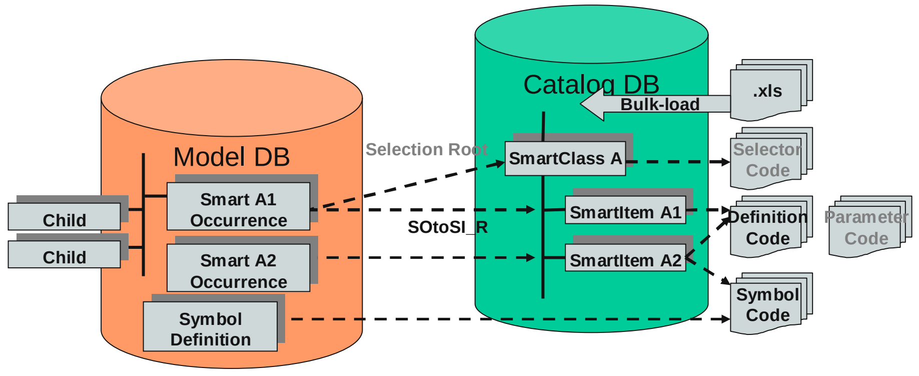
- Nozzles are nested symbols and are defined as output of the piping symbols
- Nozzle inputs come from the catalog part, which are used to create the nested nozzles
- There is no direct way for the user to control the input values for the nozzles on the piping components except by changing the data in the excel sheets and re-bulk loading them in the catalog



Overview

Smart Class/Smart Occurrence (Equipment Assembly)

- Equipment adopts the Smart Occurrence behavior
- Equipment is a custom assembly that contains members (nozzles, shapes, equipment components, structure objects, zone objects, etc..)



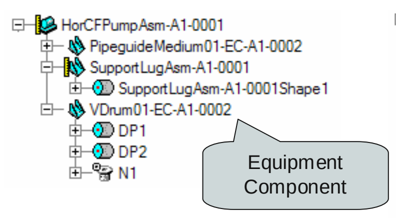
Overview

- All nozzles become First Class Business Object with their own identity. They are not nested outputs of the symbol
- Fusion of the two concepts: catalog and design equipment

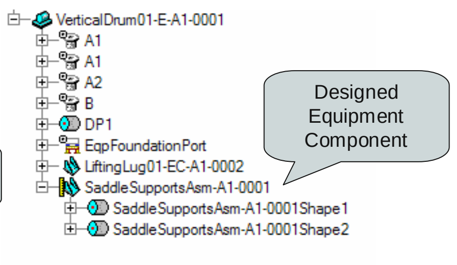
  - A catalog equipment simply becomes a programmatically defined design equipment

  - A catalog equipment can be designed once is placed

Designed Equipment



Catalog Equipment

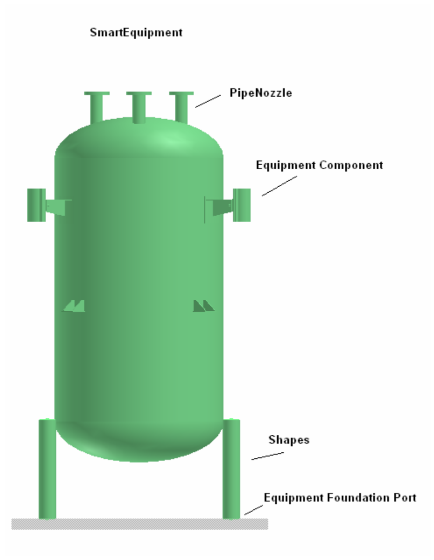

VerticalDrum01-E-A1-0001
A1
A1
A2
B
DP1
EqpFoundationPort
LiftingLug01-EC-A1-0002
SaddleSupportsAsm-A1-0001
SaddleSupportsAsm-A1-0001Shape1
SaddleSupportsAsm-A1-0001Shape2
Designed Equipment Component

SmartPlant 3D

Equipment Data Model

- Business Objects Defined in Equipment Application

  First Class Business Objects
  - CPSmartEquipment
  - CPEquipmentComponent
  - CPShape
  - CPPrismaticShape
  - CPUAImportedShapeOcc
  - CPPipeNozzle
  - CPCableTrayNozzle
  - CPConduitNozzle
  - CPCableNozzle
  - CPHvacNozzle
  - CPEqpFoundationPort



SmartPlant 3D

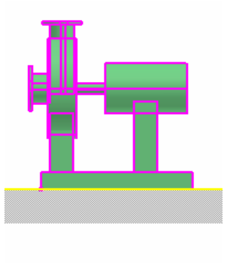
Equipment Data Model

- Business Objects Defined in Equipment Application

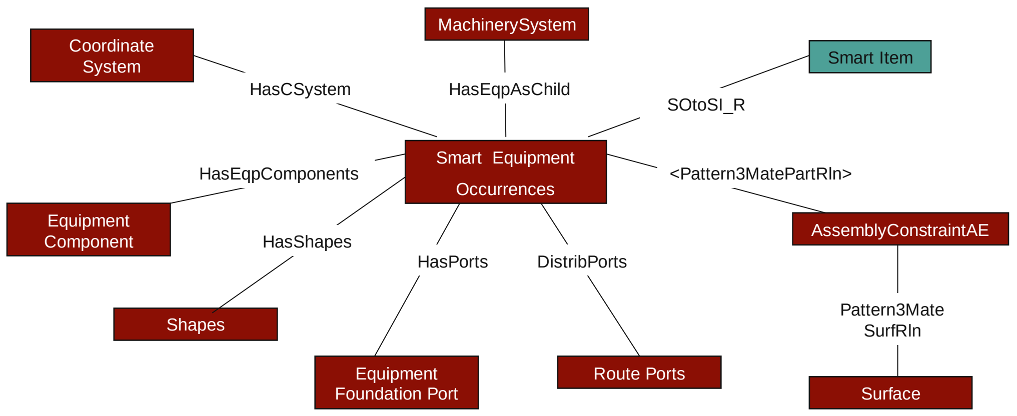
  Non-First Class Business Objects
  - CPAssemblyConstraintAE
  - CPNozzleOrientation
  - CPPipeNozzlePH
  - CPCableTrayNozzlePH
  - CPConduitNozzlePH
  - CPCableNozzlePH
  - CPHvacNozzlePH
  - CPEqpFoundationPortPH

    Port Placeholder is a persistent object that holds the information about the actual port.

Smart Equipment
|
AssemblyConstraitAE
|
Slab



Equipment Data Model



Equipment Assembly

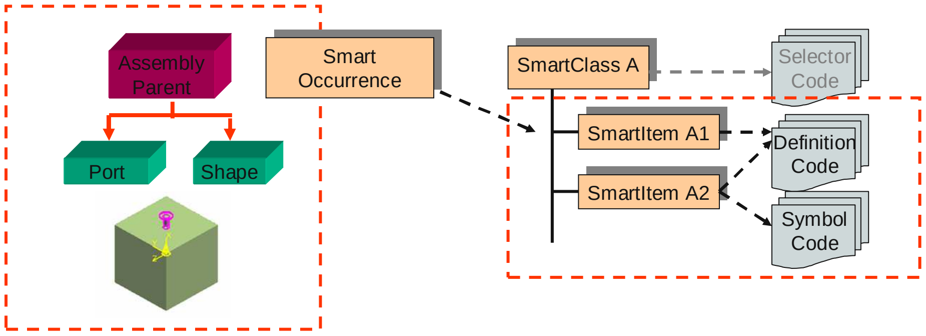
The following items have driven the design of the smart occurrence behavior:

- **similitude with symbol** : the pair custom assembly occurrence/definition is similar to the pair symbol occurrence/definition. The second one manages outputs, while the first manages members
- **no requirement on member :** a member can be any object. There is no required interface. The member is created by the owner
- **member lifetime :** the owner manages the life time of a member : when to create it and when to delete it
- **custom semantic on aggregator** : the properties of the aggregator can be controlled at compute time
- **custom semantic on members** : the properties of the members, in the context of the custom assembly, can also be controlled at compute time

Equipment Assembly

Custom Assembly

- Supports 2 primary programmable customizations (*Rules*)
  - Definition – Required, specifies the members of the equipment assembly hierarchy and the customization rules
  - Symbol – defines the graphic symbol of the aggregator (Equipment)



Equipment Assembly

Custom Assembly Definition (CAD)

- Object to be instantiated in the catalog database
- A COM (VB/VC++) class
- Implements
  - Interfaces
    - IJDUserSymbolServices: initialize and instantiate the definition
    - IJEquipUserAttrMgmt: provide methods to control the member's attributes
  - Custom methods
    - Controls some aspects of the smart equipment (Aggregator) itself
    - Creates/Controls member objects

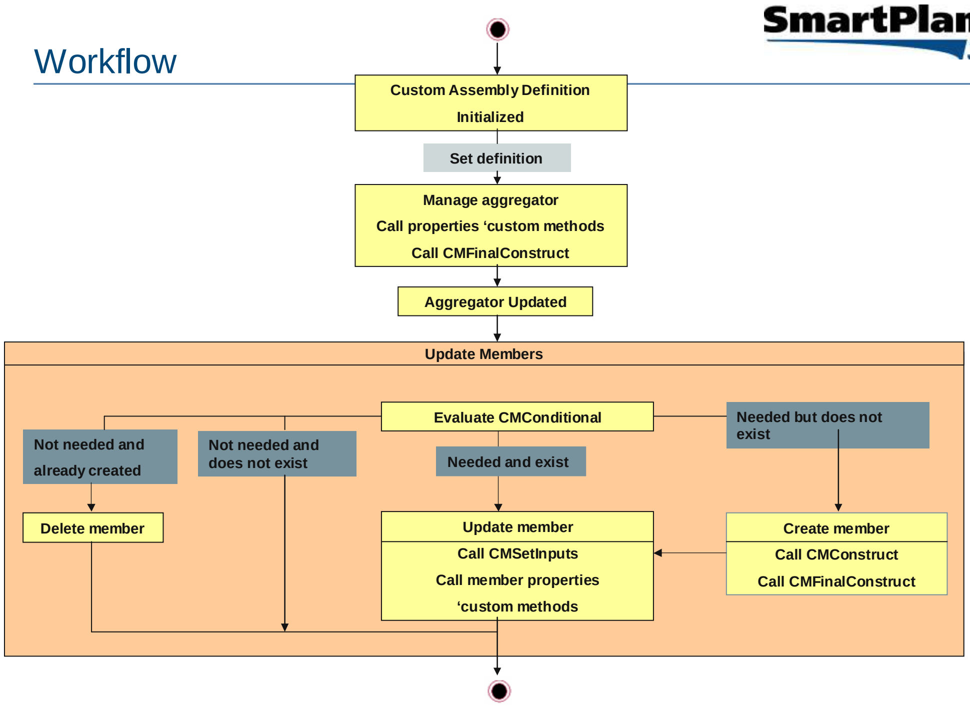
Equipment Assembly

Custom method

- Call-back function specified by users to provide customizations
  - Create objects
  - Modify existing objects
  - Create relationships between objects

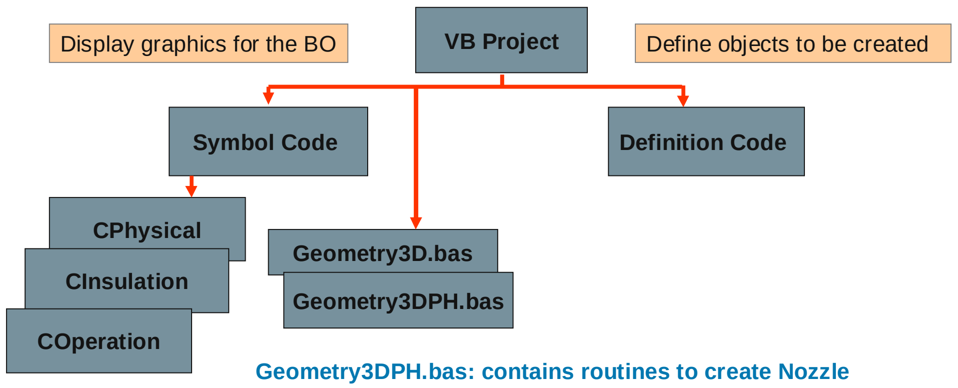
- Must be unique by name in implementing class
  - CMConditional() – optional
  - CMConstruct()
  - CMFinalConstruct() - optional
  - CMSetInputs() - optional
  - CMEvaluate() – optional
  - CMEvaluateGeometry()
  - CMRelease() - optional
  - CMCount() - optional

Workflow



Equipment Assembly

Visual Basic Project (.vbp) and Class Modules (.cls) files:



Geometry3DPH.bas: contains routines to create Nozzle placeholder. Example:

CreateNozzlePHWithLength(index, oPartFclt, m_OutputColl, oDir, CenterPos, parLength)

Equipment Assembly

Symbol Definition

- Set the following information:
  - Inputs: are entries that drive the graphic representation of the symbol
  - Outputs: Graphics Output Description
  - Representation(s): Description

IJSymbolHelper

This interface provides methods to help in creating the symbol definition. It provides the implementation of the IJDUserSymbolServices interface as well as provide support for declaring the inputs and outputs of the symbol.

Equipment Assembly

Custom Assembly Definition

- Implements the interface IJDMemberDescriptions and IJDMemberDescription.
- IJDMemberDescriptions: provides methods to manipulate the member collection
- IJDMemberDescription: specify how the member of the smart occurrence is generated

```
Dim oMemberDescriptions As IJDMemberDescriptions
Dim oMemberDescription As IJDMemberDescription
Set oMemberDescription = Nothing
Set oMemberDescription = oMemberDescriptions.AddMember("NozzleN1", 1, "CMConstructNozzleN1",
imsCOOKIE_ID_USS_LIB)
…..
```

Equipment Assembly

Custom Assembly Definition

```
….
oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructNozzleN1"
oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsNozzleN1"
oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB,  "CMConditionalNozzleN1"
oMemberDescription.SetCMCount imsCOOKIE_ID_USS_LIB, "CMCountNozzleN1"
oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseNozzleN1"
```

Equipment Assembly

Custom Assembly Definition

- Implements the interface IJDPropertyDescriptions
- IJDPropertyDescriptions: manipulate a property on a member

```
..
Set oPropertyDescriptions = Nothing
Set oPropertyDescriptions = oMemberDescription
oPropertyDescriptions.AddProperty "NozzleN1Properties", 1, IID_IJDATTRIBUTES,
"CMEvaluateNozzleN1", imsCOOKIE_ID_USS_LIB
oPropertyDescriptions.AddProperty "NozzleN1GeometryProperties", 2, IID_IJDGEOMETRY,
"CMEvaluateGeometryNozzleN1", imsCOOKIE_ID_USS_LIB
…
```

Custom method

CMConstruct_*Xxxx*()

- Code to construct the child object
- Sets input to the child object

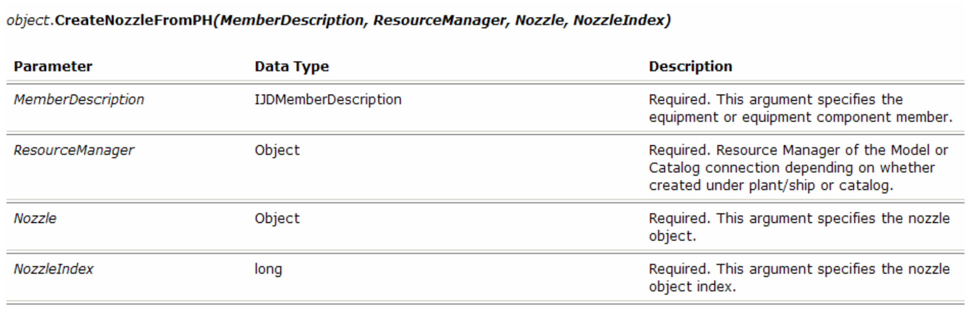
```
Public Sub CMConstructNozzleN1(ByVal pMemberDescription As IJDMemberDescription, _
                    ByVal pResourceManager As IUnknown, ByRef pObject As Object)
   Const METHOD = "CMConstructNozzleN1"
   On Error GoTo ErrorHandler
'Create Nozzle
   m_oEquipCADHelper.CreateNozzleFromPH pMemberDescription, pResourceManager, pObject, 1
   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

EquipCADHelper service

The Equipment Symbol CAD Helper service provides methods and properties to manage the custom assembly definition.

The CreateNozzleFromPH method creates a pipe nozzle from a nozzle placeholder defined in the equipment symbol.

```
m_oEquipCADHelper.CreateNozzleFromPH pMemberDescription, pResourceManager, pObject, 1
```



object.**CreateNozzleFromPH(*MemberDescription, ResourceManager, Nozzle, NozzleIndex*)**

Parameter	Data Type	Description
MemberDescription	IJDMemberDescription	Required. This argument specifies the equipment or equipment component member.
ResourceManager	Object	Required. Resource Manager of the Model or Catalog connection depending on whether created under plant/ship or catalog.
Nozzle	Object	Required. This argument specifies the nozzle object.
NozzleIndex	long	Required. This argument specifies the nozzle object index.

Custom method

CMConditional_*Xxxx()*

- Evaluates a condition
- Returns True Or False
- If True, child object is constructed

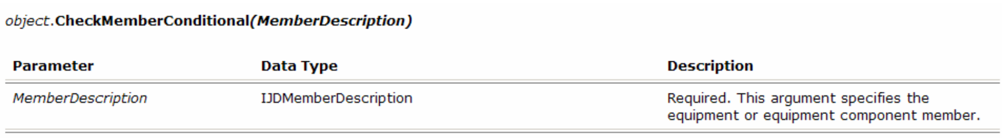
```
Public Sub CMConditionalNozzleN1(ByVal pMemberDesc As IJDMemberDescription,
ByRef IsNeeded As Boolean)
   Const METHOD = "CMConditionalNozzleN1"
   On Error GoTo ErrorHandler
   IsNeeded = m_oEquipCADHelper.CheckMemberConditional(pMemberDesc)
   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

EquipCADHelper service

The Equipment Symbol CAD Helper service provides methods and properties to manage the custom assembly definition

The CheckMemberConditional method checks whether the member is conditional based on the CanBeDeleted flag in the MakeMemberDeletable method.

```
m_oEquipCADHelper.CheckMemberConditional(pMemberDesc)
```



object.**CheckMemberConditional*(MemberDescription)***

Parameter	Data Type	Description
MemberDescription	IJDMemberDescription	Required. This argument specifies the equipment or equipment component member.

Custom method

CMEvaluateGeometry_*Xxxx*()

- Used for computing the child object.

```
Public Sub CMEvaluateGeometryNozzleN1(ByVal oPropertyDescription As
IJDPropertyDescription, pObject As Object)
    Const METHOD = "CMEvaluateGeometryNozzleN1"
    On Error GoTo ErrorHandler
    'Transform the nozzle so that it behaves like a rigid body inside the equipment
    m_oEquipCADHelper.TransformNozzleWrtPH oPropertyDescription, pObject, 1
    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

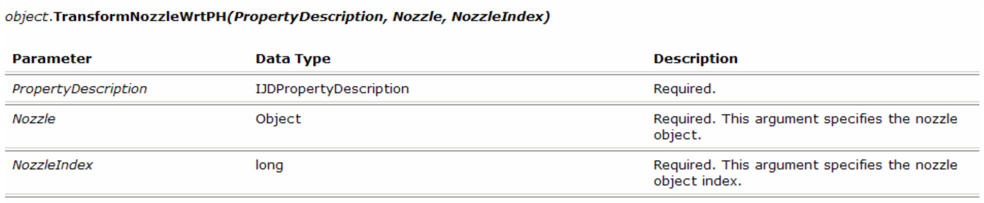
EquipCADHelper service

The Equipment Symbol CAD Helper service provides methods and properties to manage the custom assembly definition.

The TransformNozzleWrtPH method keeps the Nozzle, as a member, always positioned as the matching nozzle placeholder in the symbol.

This method prevents free transformation of the Nozzle when it is called in its CMEvaluate method

```
m_oEquipCADHelper.TransformNozzleWrtPH oPropertyDescription, pObject, 1
```



object.**TransformNozzleWrtPH(*PropertyDescription, Nozzle, NozzleIndex*)**

Parameter	Data Type	Description
PropertyDescription	IJDPropertyDescription	Required.
Nozzle	Object	Required. This argument specifies the nozzle object.
NozzleIndex	long	Required. This argument specifies the nozzle object index.

Equipment Assembly

Attribute Management

IJUserAttributeMgmt: nanage the state of a *non-SystemReadOnly attributes*

- Methods to
  - Enable/disable attributes based on state
  - Validate user edits
- 3 methods
  - OnPreLoad() – called just before displaying the property page to initialize property field states (Enabled/Disabled)
  - OnAttributeChange() – called when an attribute is edited to validate input value and potentially update other field states
  - OnPreCommit() – called when OK or Apply is clicked to persist states as needed. *Note: Most states can be determined at runtime without need to persist*

Equipment Assembly

Attribute Management

- Example:
  - OnPreLoad, you set read-only on the Member properties
  - OnAttributeChange, transfer IJDeletableMember::CanBeDeleted flag of member to IJMemberControls::DeleteFlags on Parent.
    Validate the property value.
  - OnePreCommit, Properties can be validated

Custom Assembly Definition

IJEquipUserAttrMgmt Methods

- Optional
- Methods to validate user inputs
- Override default setting

```
Private Function IJEquipUserAttrMgmt_OnAttributeChange()
Private Function IJEquipUserAttrMgmt_OnPreCommit()
Private Function IJEquipUserAttrMgmt_OnPreLoad()
```

Custom Assembly Definition

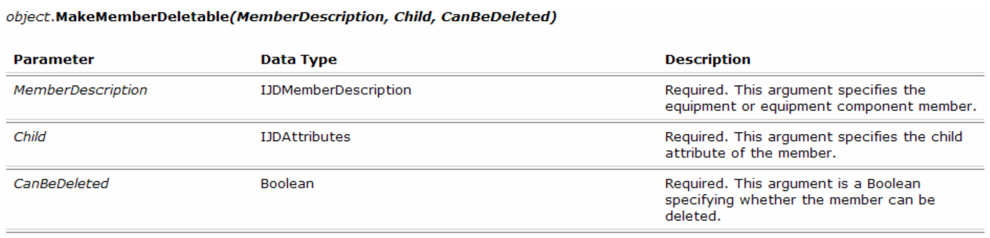
```
Private Function IJEquipUserAttrMgmt_OnAttributeChange(…)
Dim oMemberDescription As IJDMemberDescription
    Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(pIJDAttrs)
    Select Case oMemberDescription.Name
        Case "NozzleN1"
            Select Case UCase(pAttrToChange.InterfaceName)
                Case "IJDELETABLEMEMBER"
                    If UCase(pAttrToChange.AttrName) = "CANBEDELETED" Then
                        m_oEquipCADHelper.MakeMemberDeletable oMemberDescription, pIJDAttrs, _
                                                                    CBool(varNewAttrValue)
                    End If
                Case Else
            End Select
        Case Else
    End Select
    IJEquipUserAttrMgmt_OnAttributeChange = ""
```

EquipCADHelper service

The Equipment Symbol CAD Helper service provides methods and properties to manage the custom assembly definition.

The MakeMemberDeletable method sets the Custom Assembly member deletable or non-deletable depending on the setting of the CanBeDeleted flag.

```
m_oEquipCADHelper.MakeMemberDeletable oMemberDescription, pIJDAttrs, CBool(varNewAttrValue)
```

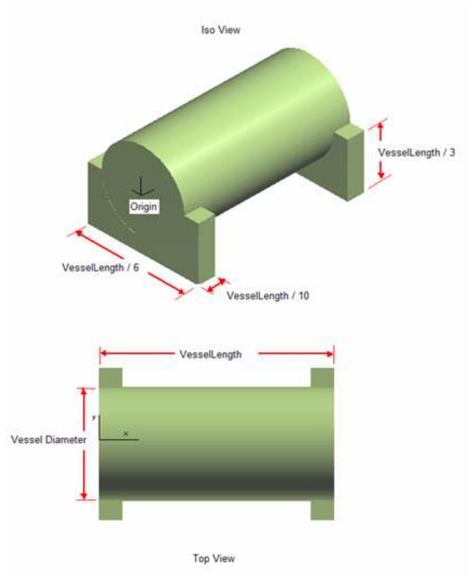


object.**MakeMemberDeletable*(MemberDescription, Child, CanBeDeleted)***

Parameter	Data Type	Description
MemberDescription	IJDMemberDescription	Required. This argument specifies the equipment or equipment component member.
Child	IJDAttributes	Required. This argument specifies the child attribute of the member.
CanBeDeleted	Boolean	Required. This argument is a Boolean specifying whether the member can be deleted.

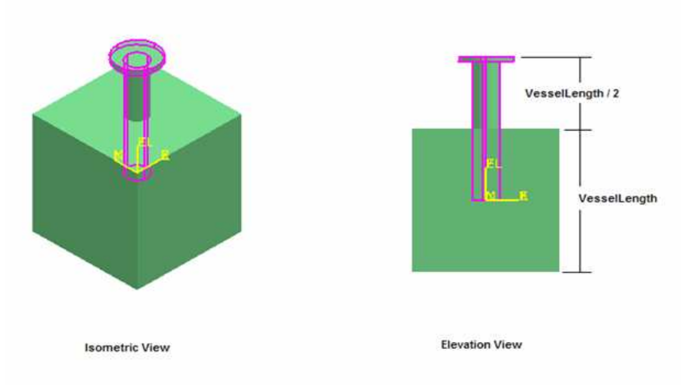
Equipment Component Assembly

Example: Equipment component without members



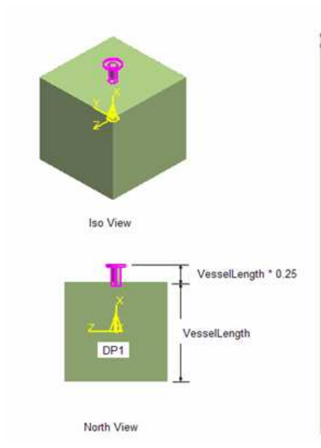
Equipment Assembly

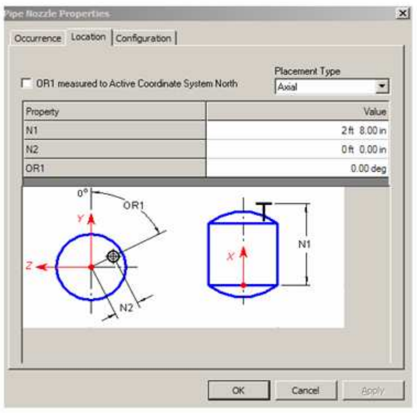
Example: Equipment with nozzle placeholder



Equipment Assembly

Example: Equipment with pipe port created relative to a shape





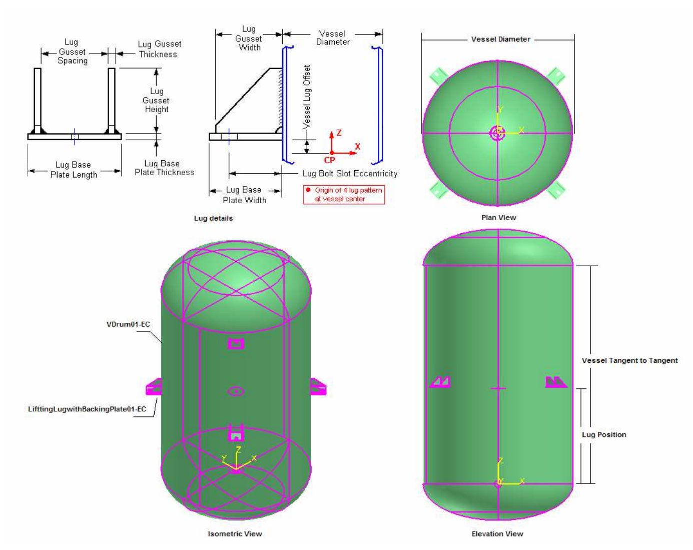
☐ OR1 measured to Active Coordinate System North

Placement Type: Axial

Property	Value
N1	2 ft 8.00 in
N2	0 ft 0.00 in
OR1	0.00 deg

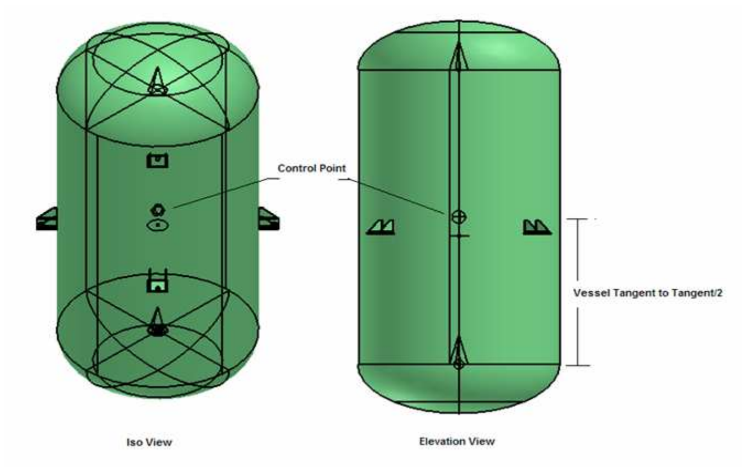
Equipment Assembly

Example: Equipment components as a member of the equipment assembly



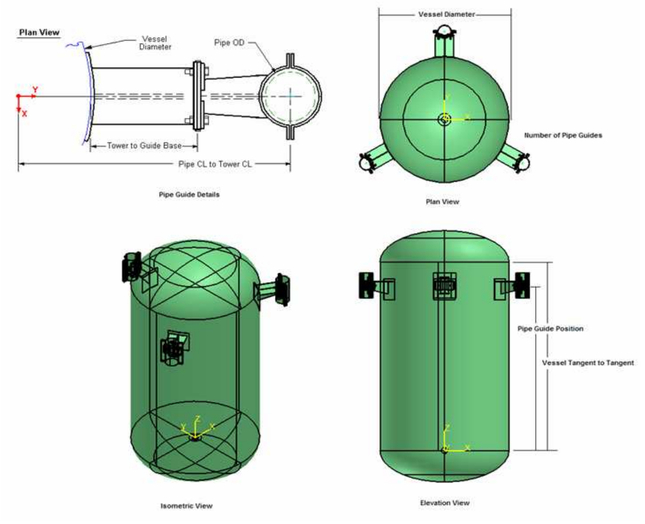
Equipment Assembly

Example: Control Point as member of the equipment assembly



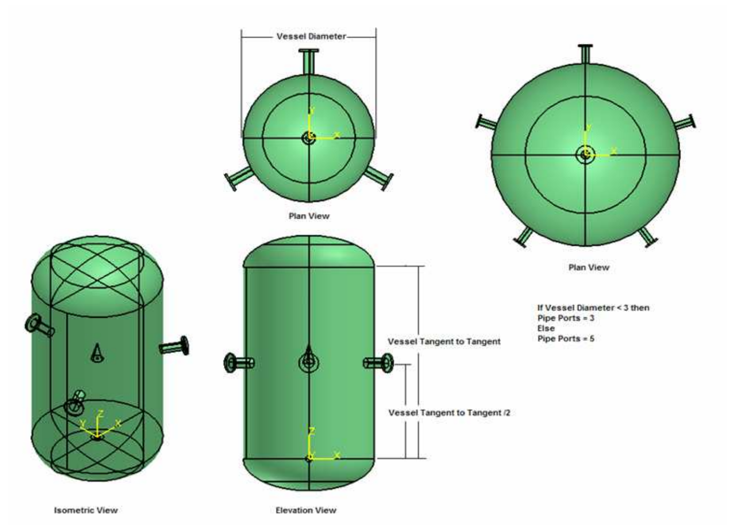
Equipment Assembly

Example: Variable members



Equipment Assembly

Example: Variable members



```
If Vessel Diameter < 3 then
Pipe Ports = 3
Else
Pipe Ports = 5
```