Activity DataBase 20 API

© Copyright Integra 2007

The Activity DataBase Application Programming Interface (API)

The ADB API is an object based programming interface accessible from any application that supports the Microsoft Common Object Model (COM). This includes Visual Basic, Visual Basic for Applications (VBA), C++ or .NET Visual Basic or C# via COM Interop.
The objects described below together with their methods and properties are a subset of the full object model but are those which are considered to be most useful to an ADB User. Two methods of using the API are available; the first is to use the objects themselves and the second is to use the ProcessMessage method of the application object. Both approaches are available for MS Excel, Access and Word while the second is the only avaiable for use with AutoCAD

The ADB Object Model

Application	Entities	Lists	Schedules
IADB	IADBProject	IADBProjectBrowseList	IADBRoomSchedule
	IADBDepartment	IADBDepartmentBrowseList	IADBSequencedRoomSchedule
	IADBRoom	IADBRoomBrowseList	IADBOrderedRoomSchedule
	IADBAssembly	IADBAssemblyBrowseList	IADBAssemblySchedule
	IADBComponent	IADBComponentBrowseList	IADBComponentSchedule
	IADBActivity	IADBActivityBrowseList	IADBActivitySchedule
	IADBBrand	IADBModelBrowseList	IADBModelSchedule
	IADBSupplier	IADBBrandBrowseList	
	IADBModel	IADBSupplierBrowseList	

Miscellaneous
IADBConditionList
IADBMsgQueue

© Copyright Integra 2007

1. The ADB application Object

IADB

Methods	Description
Sub Initialise(UserName As String, Password As String)	Initialise the ADB application and logon to ADB
Sub UnInitialise	Close all database connections. Must be called once before application terminates
Function Exists(Project As String, Name As String, Type As Integer) As Boolean	Returns True if the named Entity exists. Type: Component=0,Assembly=1,Room=2,Department=3,Activity=4,Project Class=5,Department Class=6,Room Class=7,Assembly Class=8,Component Class=9,Activity Class=10, Cat1=11, Cat2=12, Cat3=13, User=14, Grp=15, P=16, UsrLayer=17, Brand=18, Model=19, Supplier=20
Function GetLastError() As Object	Returns an IADB_MsgQueue object holding the full error stack for the last error
Function IsCodeInUse(Project As String, Code As String) As Boolean	Is this code already in use by a Department, Room, Assembly or Component within the specified Project
Function ProcessMessage(Msg As Integer, K1 As String, K2 As String, Param1 As String, Param2 As String, Param3 As String) As String	Messaging support – used for Web Services and AutoCAD Msg – Message number see Appendix K1 – Public key (username) K2 – Private key (password) Param1,2,3 - Parameters

© Copyright Integra 2007

Application Example 1

```
‘  Initailsie ADB and login
Dim oADB As New IADB
oADB.Initialise “manager”,”adb”

‘   Check if the Component OUT010 exists in project MYPROJ
If oADB.Exists(“MYPROJ”, “OUT010”, 0) Then
        MsgBox “Component: OUT010 exists”, vbOKOnly + vbInformation
Else
        MsgBox “Component: OUT010 does not exist”, vbOKOnly  + vbInformation
EndIf

‘   Check if the code is in use in project MYPROJ
If oADB.IsCodeInUse(“MYPROJ”, “B0303”) Then
        MsgBox “Code: B0303 in use”, vbOKOnly + vbInformation
Else
        MsgBox “Code: B0303 not used”, vbOKOnly  + vbInformation
EndIf

‘  Uninitialise ADB
oADB.UnInitialise
```

© Copyright Integra 2007

2. Entities

IADBProject

Properties	Data Type	Description
Description	string unlimited	Project title
Issue	string 10 chars	
Name	string 8 chars	Project code
Notes	string unlimited	
Path	String	Path to project database file
ProjectType	String 1 char	S-SQL Server blank or A for MS Access
RevisionDate	Date/Time	

Methods	Description
Function AsXml() As String	Returns the project definition as an XML string see Appendix for definition.
Sub Close()	Reset all Project attributes. The Project must be Open.
Function ComponentSchedule() As Object	Generate a Project Component Schedule.
Sub Delete()	Delete the Project
Sub DeleteUnusedDate(Type As Integer) (note typo)	Delete unused entities. 1 = Dept, 2 = Room, 3 = Assy, 4 = Component, 5 = Activity
Function FromXml(lpszXml As String) As String	Creates a project entity from a project definition as an XML string see Appendix for definition.
Function GetLastError() As Object	Returns an IADB_MsgQueue object holding the full error stack for the last error
Function IsDirty() As Boolean	True if the Project has been modified since it was Opened
Function IsOpen() As Boolean	True if the Project has been Opened
Sub Open()	Read the Project details from the ADB database. The Name property must be set before calling this method
Sub Save(UpdateRevisionDate As Boolean, OverWrite As Boolean)	Write the Project to the database
Sub SaveAs(Name As String, Path As String, UpdateRevisionDate As Boolean, OverWrite As Boolean)	Copy the current Project to a new Project

© Copyright Integra 2007

Project Example 1

```
‘ Delete Unused Data for a Project – it is assumed ADB has been initialised

‘  Open the Project
Dim oProject As IADBProject
Set oProject = New IADBProject
oProject.Name = “MYPROJ”
oProject.Open

‘ Delete unused Rooms
oProject.DeleteUnusedDate 2

‘ Close
oProject.Close
```

Project Example 2

```
‘ Output the Component Schedule for a Project – it is assumed ADB has been initialised

‘ Open the Project
Dim oProject As IADBProject
Set oProject = New IADBProject
oProject.Name = “MYPROJ”
oProject.Open

‘ Get the Project’s Component Schedule
Dim oSchedule As IADBComponentSchedule
Set oSchedule = oProject.ComponentSchedule
oSchedule.Open

‘ Check there is data
If oSchedule.Count > 0 Then

        ‘ FOR EACH item in the Schedule
        oSchedule.MoveFirst
        While Not oSchedule.IsEOF

                ‘  Output
                Debug.Print oSchedule.Name, oSchedule.Description, _
                                oSchedule.NewCount, oSchedule.Group

                ‘  Next Item
                oSchedule.MoveNext
        Wend

Endif

‘ Close and any open schedules
oProject.Close
```

© Copyright Integra 2007

IADBDepartment

Properties	Data Type	Description
Class	String 10 chars	Numeric Class
Description	string unlimited	Department Title
Name	string 10 chars	Department ADB Code. Must be set before calling Open or Save
Notes	string unlimited	Optional notes
Order	String 1 char	Department Ordering scheme (A – alpha, S – Specified, R – Room ordered)
Project	String 8 chars	ADB Project Code. Must be set before calling Open or Save
RevisionDate	Date/Time	Date/Time Department last saved
UserData(n)	string unlimited	Set n=0,1,2,3,4 for UserData1..UserData5

Methods	Description
Function ActivitySchedule() As Object	Returns a Department Activity Schedule IADBActivitySchedule
Sub AddChild(ChildType As Integer, Name As String, NewCount As Long, TransferCount As Long)	Add a child of the specified type (0 = Component, 1 = Assembly, 2 = Room) to the Department. The approprate RoomSchedule must also be open. *Note: the ADB Explorer only supports adding Rooms to Departments.*
Function AssemblySchedule() As Object	Generate an Assembly Schedule for this Department
Function AsXml() As String	Returns the Department definition as an XML string see Appendix for definition.
Sub Close()	Reset all Department attributes and close any open schedules. The Department must be Open.
Function ComponentSchedule() As Object	Returns a Department Component Schedule. IADBComponentSchedule
Sub Delete()	Delete the Department
Sub DeleteChild(ChildType As Integer, Name As String, NewCount As Long, TransferCount As Long)	Delete a child of the specified type (0 = Component, 1 = Assembly, 2 = Room) from the Department. The approprate RoomSchedule must also be open. *Note: the ADB Explorer only supports deleting Rooms from Departments.*
Function FromXml(lpszXml As String) As String	Creates a Departmententity from a Department definition as an XML string see Appendix for definition.
Function GetLastError() As Object	Returns an IADB_MsgQueue object holding the full error stack for the last error
Function IsDirty() As Boolean	True if the Department has been modified since it was Opened
Function IsOpen() As Boolean	True if the Department has been Opened
Sub Open()	Read the Department details from the ADB database. The Name & Project properties must be set before calling this method
Function OrderedRoomSchedule() As Object	Returns an IADBOrderedRoomSchedule for Departments where Order = “R”
Function RoomSchedule() As Object	Returns a IADBRoomSchedule for Departments

© Copyright Integra 2007

	where Order = “A”
Sub Save(UpdateRevisionDate As Boolean, OverWrite As Boolean)	Write the Department to the database
Sub SaveAs(Project As String, Name As String, UpdateRevisionDate As Boolean, OverWrite As Boolean)	Copy this Department, optionally to a different Project
Sub SaveAsWithSuffix(Project As String, Name As String, Suffix As String, UpdateRevisionDate As Boolean, OverWrite As Boolean)	Copy this Department, optionally to a different Project and add a suffix to each Room code
Sub Synchronise()	Re-Open all Schedules where IsDirty() = true. Reset all internal Child Entity Lists
Function SequencedRoomSchedule() As Object	Returns a IADBSequencedRoomSchedule for Departments where Order = “S”

© Copyright Integra 2007

Department Example 1

```
‘ Open an existing Department and display its properties – it is assumed ADB has been
initialised

‘  Open a Department
Dim oDepartment As IADBDepartment
Set oDepartment = New IADBDepartment

‘ Set the Project and Name
oDepartment.Project = “MYPROJ”
oDepartment.Name = “INP01”

‘ Open the Department and display it’s properties
oDepartment.Open

‘ Output the data – note the index of the UserData properties
‘ is 0 for UserData field 1, 1 for UserData field 2 etc.
Debug.Print oDepartment.Name, oDepartment.Description, _
                    oDepartment.Order, oDepartment.Notes, _
                   oDepartment.RevisionDate, oDepartment.UserData(0)

‘ Close
oDepartment.Close
```

Department Example 2

```
‘ Create new Department – it is assumed ADB has been initialised

‘  Open a Department
Dim oDepartment As IADBDepartment
Set oDepartment = New IADBDepartment

‘ Set the Project and Name
oDepartment.Project = “MYPROJ”
oDepartment.Name = “INP01”

‘ Open the Department and set the Description and Order
oDepartment.Open
oDepartment.Description = “IN-PATIENTS”
oDepartment.Order = “A”

‘ Save the Department and set the revision date to the current date
oDepartment.Save True, True

‘ Close
oDepartment.Close
```

© Copyright Integra 2007

Department Example 3
‘ Add/Delete Rooms to/from a Department – it is assumed ADB has been initialised

‘ Open a Department
Dim oDepartment As IADBDepartment
Set oDepartment = New IADBDepartment

‘ Set the Project and Name
oDepartment.Project = “MYPROJ”
oDepartment.Name = “INP01”

‘ Open the Department and set the Description and Order
oDepartment.Open

‘ Note the Room Schedule must also be open
‘ For alphanumeric ordered Departments this must be an IADBRoomSchedule
‘ For specified sequenced Departments this must be an IADBSequencedRoomSchedule
‘ For room ordered Departments this must be an IADBOrderedRoomSchedule
Dim oSchedule As IADBRoomSchedule
Set oSchedule = oDepartment.RoomSchedule
oSchedule.Open

‘ Add 3 Rooms (ChildType = 2) B0303 to the Department
‘ (To delete use the DeleteChildMethod)
oDepartment.AddChild 2,”B0303”,3,0

‘ Add 1 Room C0214 to the Department
oDepartment.AddChild 2,”C0224”,1,0

‘ Save the Department and set the revision date to the current date
oDepartment.Save True, True

‘ Close – This will also close the Room Schedule
oDepartment.Close

Department Example 4
‘ Output the Room Schedule for a Department – it is assumed ADB has been initialised

‘ Open a Department
Dim oDepartment As IADBDepartment
Set oDepartment = New IADBDepartment

‘ Set the Project and Name
oDepartment.Project = “MYPROJ”
oDepartment.Name = “INP01”

‘ Open the Department
oDepartment.Open

‘ Open the Room Schedule:
‘ IADBRoomSchedule for alphanumeric ordered
‘ IADBSequencedRoomSchedule for specified sequenced

© Copyright Integra 2007

```
‘       IADBOrderedRoomSchedule for room ordered
Dim oSchedule As IADBRoomSchedule
Set oSchedule = oDepartment.RoomSchedule
oSchedule.Open

‘ Check there is data
If oSchedule.Count > 0 Then

        ‘ FOR EACH ROOM…
        oSchedule.MoveFirst
        While Not oSchedule.IsEOF

                ‘ Output data
                Debug.Print oSchedule.Name, oSchedule.Description, oSchedule.NewCount

                ‘ Next record
                oSchedule.MoveNext
        Wend
Endif

‘ Close – This will also close the Room Schedule
oDepartment.Close
```

Department Example 5

```
‘ Output the Component Schedule for a Department - it is assumed ADB has been initialised


‘  Open a Department
Dim oDepartment As IADBDepartment
Set oDepartment = New IADBDepartment

‘ Set the Project and Name
oDepartment.Project = “MYPROJ”
oDepartment.Name = “INP01”

‘ Open the Department and set the Description and Order
oDepartment.Open

‘ Open the Component Schedule:
Dim oSchedule As IADBComponentSchedule
Set oSchedule = oDepartment.ComponentSchedule
oSchedule.Open

‘ Check there is data
If oSchedule.Count > 0 Then

        ‘ FOR EACH COMPONENT…
        oSchedule.MoveFirst
        While Not oSchedule.IsEOF

                ‘ Output data
                Debug.Print oSchedule.Name, oSchedule.Description, _
                        oSchedule.Group , oSchedule.NewCount
```

© Copyright Integra 2007

```
            ‘ Next record
            oSchedule.MoveNext
        Wend

Endif

‘ Close – This will also close the Component Schedule
oDepartment.Close
```

© Copyright Integra 2007

IADBRoom

Properties	Data Type	Description
AcceptableSoundLevel	Double	(Room Environmental Data). The acceptable sound level in the Room (L10dB(A))
Acoustics	String unlimited	(Room Environmental Data). Brief textual notes relating to the noise levels within the Room
Area	Double	Room Area
Arrestance	Double	(Room Environmental Data). Percentage arrestance
AutomaticDetection	String	(Room Environmental Data). Automatic fire detection
Ceilings	String unlimited	(Room Design Characteristics). Brief textual notes relating to Room ceilings
Class	String 10 chars	Numeric Class
ColourRendering	Boolean	(Room Environmental Data). Colour rendering required (true/false)
ColourRenderingNotes	String unlimited	(Room Environmental Data). Brief textual notes relating to the Room colour rendering requirements
Description	String unlimited	Department Title
DesignNotes	String unlimited	(Room Design Characteristics). Brief textual notes relating to Room Design Characteristics
Doorsets	String unlimited	(Room Design Characteristics). Brief textual notes relating to Room doorsets
DustSpotEfficiency	Double	(Room Environmental Data). Dust Spot Efficiency
Enclosure	String	(Room Environmental Data). Enclosure
FiltrationHumidityNotes	String unlimited	(Room Environmental Data). Brief textual notes relating to the Room humidity
FireProtection	String	(Room Environmental Data). Brief textual notes relating to fire prevention/detection features required by the Room
Flooring	String unlimited	(Room Design Characteristics). Brief textual notes relating to Room flooring
Glazing	String unlimited	(Room Design Characteristics). Brief textual notes relating to Room glazing
Hatch	String unlimited	(Room Design Characteristics). Brief textual notes relating to Room hatches
Height	Double	Room Height
HotSurfaceTemperature	Double	(Room Environmental Data). Maximum temperature in degrees Celsius of hot surfaces within the Room
HotWaterTemperature	Double	(Room Environmental Data). Maximum temperature in degrees Celsius of hot water supplies within the Room
HVAC	String unlimited	(Room Environmental Data). Brief textual notes relating to Heating, Ventilation & Air Conditioning
Illumination	String	(Room Environmental Data). Brief textual

© Copyright Integra 2007

		notes relating to the Room lighting requirements
IntrusiveNoise	Double	(Room Environmental Data). Intrusive noise levels (NR Leq)
LocalIllumination	Double	(Room Environmental Data). Local illumination (Lux)
LocalIlluminationNotes	String unlimited	(Room Environmental Data). Brief textual notes relating to the Room local lighting
MechanicalServices	Double	(Room Environmental Data). Noise related to mechanical services (NR)
MechanicalVentilationExtract	Double	(Room Environmental Data). Mechanical ventilation (extract ac/hr)
MechanicalVentilationNotes	String unlimited	(Room Environmental Data). Brief textual notes relating to the Room mechanical ventilation
MechanicalVentilationSupply	Double	(Room Environmental Data). Mechanical ventialtion (supply ac/hr)
Name	String 10 chars	Department ADB Code. Must be set before calling Open or Save
NoiseNotes	String	
Notes	String unlimited	Optional notes
Personnel	String unlimited	Room personnel requirements
PlanningRelationships	String unlimited	The relationship between this Room and adjoining spaces
Precautions	String	(Room Environmental Data). Brief textual notes relating to special safety features required by the Room
PrivacyFactor	Double	(Room Environmental Data). Privacy factor required (dB)
Project	String 8 chars	ADB Project Code. Must be set before calling Open or Save
QualityNotTolerated	String	(Room Environmental Data). The types of noise which cannt be permitted in the Room (Tonal/Intermittent/Impact)
RelativeHumidity	Double	(Room Environmental Data). Percentage relative humidity
RelativePressure	String	(Room Environmental Data). Relative pressure to adjoining space. NEG/POS
RevisionDate	Date/Time	Date/Time Room last saved
SafetyNotes	String	(Room Environmental Data). Additionaltextual notes relating to special safety features required by the Room
ServiceIllumination	Double	(Room Environmental Data). Service illumination (Lux)
ServiceIlluminationNight	Double	(Room Environmental Data). Night-time service illumination (Lux)
ServiceIlluminationNightNotes	String	(Room Environmental Data). Brief textual notes relating to the Room night-time service lighting
ServiceIlluminationNotes	String	(Room Environmental Data). Brief textual notes relating to the Room service lighting
SpaceNotes	String unlimited	
SpeechPrivacy	Boolean	(Room Environmental Data). Specifies

© Copyright Integra 2007

		whether speech privacy is required in the Room (true/false)
StandbyLightingGrade	String	(Room Environmental Data). Standby lighting grade
StandbyLightingGradeNotes	String	(Room Environmental Data). Standby lighting grade
SummerTemperature	Double	(Room Environmental Data). Maximum summer in degrees Celsius
TemperatureNotes	String unlimited	(Room Environmental Data). Brief textual notes relating to the Room temperature requirements
Type	String	Entity type. Normally T (Template) I (instanced)
UserData(n)	String unlimited	Set n=0,1,2,3,4 for UserData1..UserData5
Walls	String unlimited	Room Design Characteristics). Brief textual notes relating to Room walls
Windows	String unlimited	Room Design Characteristics). Brief textual notes relating to Room windows
WinterTemperature	Double	(Room Environmental Data). Minimum winter in degrees Celsius

Methods	**Description**
Function ActivitySchedule() As Object	Returns a Room Activity Schedule IADBActivitySchedule
Sub AddChild(ChildType As Integer, Name As String, NewCount As Long, TransferCount As Long)	Add a child of the specified type (0 = Component, 1 = Assembly, 2 = Activity) to the Room. The approprate Schedule must also be open.
Function AssemblySchedule() As Object	Returns an Assembly Schedule IADBAssemblySchedule for the Room
Function AsXml() As String	Returns the Room definition as an XML string see Appendix for definition.
Sub ChildDisposition(Type As Integer, Name As String, NewCount As Long, TransferCount As Long)	Adjust the New & Transferred counts for the specified child entity. NewCount + TransferCount must equal the current total count. Components: ChildType = 0
Sub ClearSubEntityMask()	Remove any previously set Sub-Entity filters
Sub Close()	Reset all Room attributes and close any open schedules. The Room must be Open.
Function ComponentSchedule() As Object	Returns a Room Component Schedule. IADBComponentSchedule
Sub Delete()	Delete the Room
Sub DeleteChild(ChildType As Integer, Name As String, NewCount As Long, TransferCount As Long)	Delete a child of the specified type (0 = Component, 1 = Assembly, 2 = Activity) from the Room. The approprate Schedule must also be open.
Function FromXml(lpszXml As String) As String	Creates a Room entity from a Room definition as an XML string see Appendix for definition.
Function GetLastError() As Object	Returns an IADB_MsgQueue object holding the full error stack for the last error
Sub GetSpaceBox(X1 As Double, Y1 As Double, Z1 As Double, X2 As Double, Y2 As Double, Z2 As Double)	Get the coordinates of two opposite corners of a bounding cube which fully encloses the Room
Function IsDirty() As Boolean	True if the Room has been modified since it was Opened

© Copyright Integra 2007

Function IsOpen() As Boolean	True if the Room has been Opened
Sub Open()	Read the Room details from the ADB database. The Name & Project properties must be set before calling this method
Function PanelSchedule() As Object	Returns and IADBPanelSchedule (i.e walls)
Sub Save(UpdateRevisionDate As Boolean, OverWrite As Boolean)	Write the Room to the database
Sub SaveAs(Project As String, Name As String, UpdateRevisionDate As Boolean, OverWrite As Boolean)	Copy this Room, optionally to a different Project
Sub SetSpaceBox(X1 As Double, Y1 As Double, Z1 As Double, X2 As Double, Y2 As Double, Z2 As Double)	Set the coordinates of two opposite corners of a bounding cube which fully encloses the Room
Sub SetSubEntityMask(SubEntity As Integer)	Open/Save will ignore thye sepcified sub-entity data. 0 = Space, 1 = env, 2 = env notes, 3 = character, 4 = personnel, 5 = planning, 6 = class
Sub ShiftActivity(Direction As Integer)	Move the Activity one position Direction = 0 forwards (down list) Direction = 1 backwards (up list)
Sub Synchronise()	Re-Open all Schedules where IsDirty() = true. Reset all internal Child Entity Lists

© Copyright Integra 2007

Room Example 1

```
‘ Open an existing Room and output it’s properties – it is assumed ADB has been initialised

‘  Create a Room entity
Dim oRoom As IADBRoom
Set oRoom = New IADBRoom

‘ Set the Project and Name
oRoom.Project = “MYPROJ”
oRoom.Name = “B0303”

‘ Open the Room
oRoom.Open

‘ Output the Data
Debug.Print oRoom.Name, oRoom.Description, oRoom.RevisionDate

‘ Space data
Debug.Print  oRoom.Area, oRoom.Height

‘ Personnel
Debug.Print  oRoom.Personnel

‘ Planning Relationships
Debug.Print  oRoom.PlanningRelationships

‘ Environmental Data
Debug.Print  oRoom.WinterTemperature

‘ Design Character Data
Debug.Print  oRoom.Flooring

‘ Close
oRoom.Close
```

© Copyright Integra 2007

Room Example 2

```
‘ Create new Room or modify an existing Room and set its properties – it is assumed ADB has been initialised

‘  Create a Room entity
Dim oRoom As IADBRoom
Set oRoom = New IADBRoom

‘ Set the Project and Name
oRoom.Project = “MYPROJ”
oRoom.Name = “B9000”

‘ Open the Room and set the Description
oRoom.Open
oRoom.Description = “Bedroom”

‘ Space data
oRoom.Area = 20.0
oRoom.Height = 2400

‘ Personnel
oRoom.Personnel = “1-Patient, 2-Others”

‘ Planning Relationships
oRoom.PlanningRelationships = “Adjacent to staff base”

‘ Environmental Data
oRoom.WinterTemperature = 21.0

‘ Design Character Data
oRoom.Flooring = “Carpet”

‘ Save the Room and set the revision date to the current date
oRoom.Save True, True

‘ Close
oRoom.Close
```

© Copyright Integra 2007

Room Example 3

```
‘ Add/Delete Activities, Assemblies and Components to/from a Room  – it is assumed ADB has been initialised

‘  Create a Room entity
Dim oRoom As IADBRoom
Set oRoom = New IADBRoom

‘ Set the Project and Name and Open the Room
oRoom.Project = “MYPROJ”
oRoom.Name = “B9000”
oRoom.Open

‘ Note the Component Schedule must also be open
Dim oCompSchedule As IADBComponentSchedule
Set oCompSchedule = oRoom.ComponentSchedule
oCompSchedule.Open

‘ Add new Components  (ChildType = 0) to the Room
‘ The Components must exist in the Project so better to use the Exists
‘ method for the application object to check if not sure
 oRoom.AddChild 0, "OUT010", 2, 0
 oRoom.AddChild 0, "OUT005", 1, 0
 oRoom.AddChild 0, "CHA017", 4, 0
 oRoom.Save True, True

‘ Close – This will also close the schedules
oRoom.Close
```

© Copyright Integra 2007

Room Example 4

```
‘ Output the Actvity, Assembly and Component Schedules for a Room – it is assumed ADB has been initialised

‘  Open a Room
Dim oRoom As IADBRoom
Set oRoom = New IADBRoom

‘ Set the Project and Name
oRoom.Project = “MYPROJ”
oRoom.Name = “B9000”

‘ Open the Room
oRoom.Open

‘ Open the Activity schedule
Dim oActSchedule As IADBActivitySchedule
Set oActSchedule = oRoom.ActivitySchedule
oActSchedule.Open

 ‘ FOR EACH ACTIVITY…
oActSchedule.MoveFirst
While Not oActSchedule.IsEOF

        ‘ Output data
        Debug.Print oActSchedule.Name, oActSchedule.Description

        ‘ Next record
        oActSchedule.MoveNext
Wend

‘ Open the Assembly schedule
Dim oAssySchedule As IADBAssySchedule
Set oAssySchedule = oRoom.AssemblySchedule
oAssySchedule.Open

 ‘ FOR EACH ASSEMBLY…
oAssySchedule.MoveFirst
While Not oAssySchedule.IsEOF

        ‘ Output data
        Debug.Print oAssySchedule.Name, oAssySchedule.Description, _
                oAssySchedule.NewCount

        ‘ Next record
        oAssySchedule.MoveNext
Wend
‘ Open the Component schedule
Dim oCompSchedule As IADBComponentSchedule
Set oCompSchedule = oRoom.ComponentSchedule
oCompSchedule.Open

‘ FOR EACH COMPONENT…
```

© Copyright Integra 2007

```
oCompSchedule.MoveFirst
While Not oCompSchedule.IsEOF

        ‘ Output data
        Debug.Print oCompSchedule.Name, oCompSchedule.Description, _
                oCompSchedule.NewCount, oCompSchedule.TransferCount, _
                oCompSchedule.Group

        ‘ Next record
        oCompSchedule.MoveNext
Wend

‘ Close – This will also close all the open schedules
oRoom.Close
```

© Copyright Integra 2007

IADBAssembly

Properties	Data Type	Description
Class	String 10 chars	Numeric Class
Description	String unlimited	Department Title
Name	String 10 chars	Department ADB Code. Must be set before calling Open or Save
Notes	String unlimited	Optional notes
Project	String 8 chars	ADB Project Code. Must be set before calling Open or Save
RevisionDate	Date/Time	Date/Time Assembly last saved
Type	String	Entity type. Normally T (Template) I (instanced)
UserData(n)	String unlimited	Set n=0,1,2,3,4 for UserData1..UserData5

Methods	Description
Function ActivitySchedule() As Object	Returns an Assembly Activity Schedule IADBActivitySchedule
Sub AddChild(ChildType As Integer, Name As String, NewCount As Long, TransferCount As Long)	Add a child of the specified type (0 = Component, 1 = Assembly, 2 = Activity) to the Assembly. The approprate Schedule must also be open.
Function AssemblySchedule() As Object	Returns an Assembly Schedule IADBAssemblySchedule for the Room
Function AsXml() As String	Returns the Assembly definition as an XML string see Appendix for definition.
Sub ClearSubEntityMask()	Remove any previously set Sub-Entity filters
Sub Close()	Reset all Assembly attributes and close any open schedules. The Assembly must be Open.
Function ComponentSchedule() As Object	Returns a Assembly Component Schedule. IADBComponentSchedule
Sub Delete()	Delete the Assembly
Sub DeleteChild(ChildType As Integer, Name As String, NewCount As Long, TransferCount As Long)	Delete a child of the specified type (0 = Component, 1 = Assembly, 2 = Activity) from the Assembly. The approprate Schedule must also be open.
Function FromXml(lpszXml As String) As String	Creates an Assembly entity from an Assembly definition as an XML string see Appendix for definition.
Function GetLastError() As Object	Returns an IADB_MsgQueue object holding the full error stack for the last error
Sub GetSpaceBox(X1 As Double, Y1 As Double, Z1 As Double, X2 As Double, Y2 As Double, Z2 As Double)	Get the coordinates of two opposite corners of a bounding cube which fully encloses the Room
Function IsDirty() As Boolean	True if the Assembly has been modified since it was Opened
Function IsOpen() As Boolean	True if the Assembly has been Opened
Sub Open()	Read the Assembly details from the ADB database. The Name & Project properties must be set before calling this method
Sub Save(UpdateRevisionDate As Boolean, OverWrite As Boolean)	Write the Assembly to the database
Sub SaveAs(Project As String, Name As String,	Copy this Assembly, optionally to a different

© Copyright Integra 2007

UpdateRevisionDate As Boolean, OverWrite As Boolean)	Project
Sub SetSpaceBox(X1 As Double, Y1 As Double, Z1 As Double, X2 As Double, Y2 As Double, Z2 As Double)	Set the coordinates of two opposite corners of a bounding cube which fully encloses the Assembly
Sub SetSubEntityMask(SubEntity As Integer)	SubEntity = 0, Open/Save will ignore Assembly Classification data
Sub ShiftActivity(Direction As Integer)	Move the Activity one position Direction = 0 forwards (down list) Direction = 1 backwards (up list)
Sub Synchronise()	Re-Open all Schedules where IsDirty() = true. Reset all internal Child Entity Lists

© Copyright Integra 2007

Assembly Example 1

```
‘ Open an existing Assembly or output it’s properties – it is assumed ADB has been initialised

‘  Create an Assembly entity
Dim oAssembly As IADBAssembly
Set oAssembly = New IADBAssembly

‘ Set the Project and Name
oAssembly.Project = “MYPROJ”
oAssembly.Name = “SA1244”

‘ Open the Assembly
oAssembly.Open

‘ Ouput the Data
Debug.Print oAssembly.Name, oAssembly.Description , _
        oAssembly.Class, oAssembly.RevisionDate

‘ Close
oAssembly.Close
```

© Copyright Integra 2007

Assembly Example 2

```
‘ Create new Assembly or modify an existing Assembly and set its properties – it is assumed ADB has been initialised

‘  Create an Assembly entity
Dim oAssembly As IADBAssembly
Set oAssembly = New IADBAssembly

‘ Set the Project and Name
oAssembly.Project = “MYPROJ”
oAssembly.Name = “SA9000”

‘ Open the Assembly and set the Description
oAssembly.Open
oAssembly.Description = “Sanitary Assembly”
oAssembly.Class = “3311”

‘ Save the Assembly and set the revision date to the current date
oAssembly.Save True, True

‘ Close
oAssembly.Close
```

© Copyright Integra 2007

Assembly Example 3

```
‘ Add/Delete Actvities, Assemblies and Components to/from an Assembly  – it is assumed ADB has been initialised

‘  Create a Assembly entity
Dim oAssembly As IADBAssembly
Set oAssembly = New IADBAssembly

‘ Set the Project and Name and Open the Assembly
oAssembly.Project = “MYPROJ”
oAssembly.Name = “SA9000”
oAssembly.Open

‘ Note the Component Schedule must also be open
Dim oCompSchedule As IADBComponentSchedule
Set oCompSchedule = oAssembly.ComponentSchedule
oCompSchedule.Open

‘ Add new Components  (ChildType = 0) to the Assembly
‘ The Components must exist in the Project so better to use the Exists
‘ method for the application object to check if not sure
 oAssembly.AddChild 0, "OUT010", 2, 0
 oAssembly.AddChild 0, "OUT005", 1, 0
 oAssembly.Save True, True

‘ Close – This will also close the schedule
oAssembly.Close
```

© Copyright Integra 2007

Assembly Example 4

```
‘ Output the Actvity, Assembly and Component Schedules for an Assembly – it is assumed ADB has been initialised

‘  Create a new Assembly object
Dim oAssembly As IADBAssembly
Set oAssembly = New IADBAssembly

‘ Set the Project and Name
oAssembly.Project = “MYPROJ”
oAssembly.Name = “SA9000”

‘ Open the Assembly
oAssembly.Open

‘ Open the Activity schedule
Dim oActSchedule As IADBActivitySchedule
Set oActSchedule = oAssembly.ActivitySchedule
oActSchedule.Open

 ‘ FOR EACH ACTIVITY…
oActSchedule.MoveFirst
While Not oActSchedule.IsEOF

        ‘ Output data
        Debug.Print oActSchedule.Name, oActSchedule.Description

        ‘ Next record
        oActSchedule.MoveNext
Wend

‘ Open the Assembly schedule
Dim oAssySchedule As IADBAssemblySchedule
Set oAssySchedule = oAssembly.AssemblySchedule
oAssySchedule.Open

 ‘ FOR EACH ASSEMBLY…
oAssySchedule.MoveFirst
While Not oAssySchedule.IsEOF

        ‘ Output data
        Debug.Print oAssySchedule.Name, oAssySchedule.Description, _
                oAssySchedule.NewCount

        ‘ Next record
        oAssySchedule.MoveNext
Wend
‘ Open the Component schedule
Dim oCompSchedule As IADBComponentSchedule
Set oCompSchedule = oAssembly.ComponentSchedule
oCompSchedule.Open

‘ FOR EACH COMPONENT…
```

© Copyright Integra 2007

```
oCompSchedule.MoveFirst
While Not oCompSchedule.IsEOF

        ‘ Output data
        Debug.Print oCompSchedule.Name, oCompSchedule.Description, _
                oCompSchedule.NewCount, oCompSchedule.TransferCount, _
                oCompSchedule.Group

        ‘ Next record
        oCompSchedule.MoveNext
Wend

‘ Close – This will also close all the open schedules
oAssembly.Close
```

© Copyright Integra 2007

IADBComponent

Properties	Data Type	Description
Class	String 10 chars	Numeric Class
Cost	Double	New Component Cost
DefaultModel	String 10 chars	ADB Code of the default Model
Description	String unlimited	Department Title
GenericSpec	String unlimited	
Group	String 1 char	Component Group Code
InstallerType	String 1 char	The ADB code for the type of organisation which installs the Component
Layer	String 31 char	AutoCAD layer on which Component will be drawn
Level	String 1 char	The Component level. Used by the AutoCAD library 1-Minor 2-Mid range 3-Major Component
Name	String 10 chars	Component ADB Code. Must be set before calling Open or Save
Notes	String unlimited	Descriptive textual notes
NSVCode	String 10 chars	Alternative e.g. supplies or code
PartNumber	String 30 chars	Component Manufacturers Identification code.
PhysicalSize	String unlimited	Textual description of the spatial requirements of the Component. No longer used by ADB
Project	String 8 chars	ADB Project Code. Must be set before calling Open or Save
RevisionDate	Date/Time	Date/Time Assembly last saved
ScheduleFlag	Boolean	If TRUE, this Component will be included in Component Schedules
ServiceBool(nItem As Integer)	Boolean	10 TRUE/FALSE fileds used for services Set nItem=0,1,2,3,4,5,6,7,8,9
ServiceFloat(nItem As Integer)	Single	10 Single precision floating point numbers fileds used for services Set nItem=0,1,2,3,4,5,6,7,8,9
ServiceInt(nItem As Integer)	Integer	10 Integer fileds used for services Set nItem=0,1,2,3,4,5,6,7,8,9
ServiceString(nItem As Integer)	String unlimited	10 String fileds used for services Set nItem=0,1,2,3,4,5,6,7,8,9
SupplierType	String 1 char	The ADB code for the type of organisation which supplies the Component
InstallerType	String 1 char	The ADB code for the type of organisation which installs the Component
TransferCost	Double	Transferred Component Cost
Type	String	Entity type. Normally T (Template) I (instanced)
User1Type	String 1 char	First user-definable type attribute
User2Type	String 1 char	Second user-definable type attribute
UserData(n)	String unlimited	Set n=0,1,2,3,4 for UserData1..UserData5

© Copyright Integra 2007

Methods	Description
Function AsXml() As String	Returns the Component definition as an XML string see Appendix for definition.
Sub ClearSubEntityMask()	Remove any previously set Sub-Entity filters
Sub Close()	Reset all Component attributes and close any open schedules. The Component must be Open.
Sub Delete()	Delete the Component
Function FromXml(lpszXml As String) As String	Creates an Component entity from a Component definition as an XML string see Appendix for definition.
Function GetLastError() As Object	Returns an IADB_MsgQueue object holding the full error stack for the last error
Sub GetSpaceBox(X1 As Double, Y1 As Double, Z1 As Double, X2 As Double, Y2 As Double, Z2 As Double)	Get the coordinates of two opposite corners of a bounding cube which fully encloses the Component
Function HasGraphic(nView As Integer) As Boolean	Retruns TRUE if the Component has a graphic view. Set nView 0-3D, 1-Front Elevation, 2-Plan, 3-Rear, 4-Left, 5-Right
Function IsDirty() As Boolean	True if the Component has been modified since it was Opened
Function IsOpen() As Boolean	True if the Component has been Opened
Function ModelSchedule() As Object	Returns an IADBModelSchedule for the Component
Sub Open()	Read the Component details from the ADB database. The Name & Project properties must be set before calling this method
Sub Save(UpdateRevisionDate As Boolean, OverWrite As Boolean)	Write the Component to the database
Sub SaveAs(Project As String, Name As String, UpdateRevisionDate As Boolean, OverWrite As Boolean)	Copy this Component, optionally to a different Project. The Component must be Open, and must not have been modified (i.e. IsDirty = false)
Sub SetSpaceBox(X1 As Double, Y1 As Double, Z1 As Double, X2 As Double, Y2 As Double, Z2 As Double)	Set the coordinates of two opposite corners of a bounding cube which fully encloses the Component
Sub SetSubEntityMask(SubEntity As Integer)	Ignore the specified sub-Entity, where 0=Class,1=NSV,2=Graphic(-3),3=Graphic(-E),4=Graphic(-P),5=Graphic(RE),6=Graphic(SE),7=Graphic(TE)

© Copyright Integra 2007

Component Example 1

```
‘ Open an existing Component or output it’s properties – it is assumed ADB has been
initialised

‘  Create a Component entity
Dim oComponent As IADBComponent
Set oComponent = New IADBComponent

‘ Set the Project and Name
oComponent.Project = “MYPROJ”
oComponent.Name = “CHA017”

‘ Open the Component
oComponent.Open

‘ Ouput the Data
Debug.Print oComponent.Name, oComponent.Description , _
          oComponent.Class, oComponent.RevisionDate, _
          oComponent.Group, oComponent.ServiceBool(0), _
          oComponent.ServiceString(0), oComponent.ServiceFloat(0)


‘ Close
oComponent.Close
```

© Copyright Integra 2007

Component Example 2

```
‘ Create new Component or modify an existing Component and set its properties – it is assumed ADB has been initialised

‘  Create an Component entity
Dim oComponent As IADBComponent
Set oComponent = New IADBComponent

‘ Set the Project and Name
oComponent.Project = “MYPROJ”
oComponent.Name = “DEF900”

‘ Open the Component and set the Description and other properties
oComponent.Open
oComponent.Description = “DEFIBRILLATOR”
oComponent.Class = “2301”
oComponent.Group = “3”
oComponent.Cost = 2000.0


‘ Save the Component and set the revision date to the current date
oComponent.Save True, True

‘ Close
oComponent.Close
```

© Copyright Integra 2007

Component Example 3

```
‘ Add/Delete Models to the Component
‘ Note that unlike Departments, Rooms and Assemblies the Component Entity
‘  does not support children and therefore has no AddChild method.
‘  As support for Models etc. was added at a later date the an the Add method
‘  has been assigned to the ModelSchedule
‘ – again it is assumed ADB has been initialised

‘  Create a Component entity
Dim oComponent As IADBComponent
Set oComponent = New IADBComponent

‘ Set the Project and Name and Open the Component
oComponent.Project = “MYPROJ”
oComponent.Name = “DEF900”
oComponent.Open

‘ Note the Model Schedule must also be open
Dim oModelSchedule As IADBModelSchedule
Set oModelSchedule = oComponent.ModelSchedule
oModelSchedule.Open

‘ Add a Model – The Model, Brand and Supplier must exist
‘ Note place holders “” are used for fields which are not required
oModelSchedule.AddModel  “HP4537A”,””, “”, “”, “HP”, “”, “AGILENT”, 0.0, Now

‘  Save the ModelSchedule
oModelSchedule.Save

‘  Save the Component
oComponent.Save True, True

‘ Close – This will also close the schedules
oComponent.Close
```

© Copyright Integra 2007

Component Example 4
‘ Output the Model Schedule for a Component – it is assumed ADB has been initialised

‘ Open a Component
```
Dim oComponent As IADBComponent
Set oComponent = New IADBComponent

‘ Set the Project and Name
oComponent.Project = “MYPROJ”
oComponent.Name = “DEF900”

‘ Open the Component
oComponent.Open

‘ Open the Model schedule
Dim oModSchedule As IADBModelSchedule
Set oModSchedule = oComponent.ModelSchedule
oModSchedule.Open

 ‘ FOR EACH MODEL…
oModSchedule.MoveFirst
While Not oModSchedule.IsEOF

        ‘ Output data
      Debug.Print oModSchedule.Name, oModSchedule.Description, _
            oModSchedule.Brand, oModSchedule.Supplier, _
            oModSchedule.Cost, oModSchedule.IsDefault

        ‘ Next record
        oModSchedule.MoveNext
Wend

‘ Close – This will also any open schedules
oComponent.Close
```

© Copyright Integra 2007

IADBActivity

Properties	Data Type	Description
Class	String 10 chars	Numeric Class
Description	String unlimited	Activity Title
Name	String 10 chars	Activity ADB Code. Must be set before calling Open or Save
Notes	String unlimited	Descriptive textual notes
Project	String 8 chars	ADB Project Code. Must be set before calling Open or Save
RevisionDate	Date/Time	Date/Time Assembly last saved
UserData(n)	String unlimited	Set n=0,1,2,3,4 for UserData1..UserData5

Methods	Description
Function AsXml() As String	Returns the Activity definition as an XML string see Appendix for definition.
Sub Close()	Reset all Activity attributes. The Activity must be Open.
Sub Delete()	Delete the Activity
Function FromXml(lpszXml As String) As String	Creates an Activity entity from an Activity definition as an XML string see Appendix for definition.
Function GetLastError() As Object	Returns an IADB_MsgQueue object holding the full error stack for the last error
Function IsDirty() As Boolean	True if the Activity has been modified since it was Opened
Function IsOpen() As Boolean	True if the Activity has been Opened
Sub Open()	Read the Activity details from the ADB database. The Name & Project properties must be set before calling this method
Sub Save(UpdateRevisionDate As Boolean, OverWrite As Boolean)	Write the Activity to the database
Sub SaveAs(Project As String, Name As String, UpdateRevisionDate As Boolean, OverWrite As Boolean)	Copy this Activity, optionally to a different Project. The Activity must be Open, and must not have been modified (i.e. IsDirty = false)

© Copyright Integra 2007

Activity Example 1

```
‘ Open an existing Activity and output it’s properties – it is assumed ADB has been initialised

‘  Create an Activity entity
Dim oActivity As IADBActivity
Set oActivity = New IADBActivity

‘ Set the Project and Name
oActivity.Project = “MYPROJ”
oActivity.Name = “WAS009”

‘ Open the Activity
oActivity.Open

‘ Ouput the Data
Debug.Print oActivity.Name, oActivity.Description , _
        oActivity.Class, oActivity.RevisionDate


‘ Close
oActivity.Close
```

© Copyright Integra 2007

Activity Example 2

```
‘ Create new Activity or modify an existing Activity and set its properties – it is assumed ADB has been initialised

‘  Create an Activity entity
Dim oActivity As IADBActivity
Set oActivity = New IADBActivity

‘ Set the Project and Name
oActivity.Project = “MYPROJ”
oActivity.Name = “WAS900”

‘ Open the Activity and set the Description and other properties
oActivity.Open
oActivity.Description = “Washing Activity”
oActivity.Class = “3310”

‘ Save the Activity and set the revision date to the current date
oActivity.Save True, True

‘ Close
oActivity.Close
```

© Copyright Integra 2007

IADBBrand

Properties	Data Type	Description
Description	String unlimited	Brand Title
Name	String 10 chars	Brand ADB Code. Must be set before calling Open or Save
Notes	String unlimited	Descriptive textual notes
Project	String 8 chars	ADB Project Code. Must be set before calling Open or Save
RevisionDate	Date/Time	Date/Time Assembly last saved
UserData(n)	String unlimited	Set n=0,1,2,3,4 for UserData1..UserData5

Methods	Description
Function AsXml() As String	Returns the Brand definition as an XML string see Appendix for definition.
Sub Close()	Reset all Brand attributes. The Brand must be Open.
Sub Delete()	Delete the Brand
Function FromXml(lpszXml As String) As String	Creates an Brand entity from an Brand definition as an XML string see Appendix for definition.
Function GetLastError() As Object	Returns an IADB_MsgQueue object holding the full error stack for the last error
Function IsDirty() As Boolean	True if the Brand has been modified since it was Opened
Function IsOpen() As Boolean	True if the Brand has been Opened
Sub Open()	Read the Brand details from the ADB database. The Name & Project properties must be set before calling this method
Sub Save(UpdateRevisionDate As Boolean, OverWrite As Boolean)	Write the Brand to the database
Sub SaveAs(Project As String, Name As String, UpdateRevisionDate As Boolean, OverWrite As Boolean)	Copy this Brand, optionally to a different Project. The Brand must be Open, and must not have been modified (i.e. IsDirty = false)

© Copyright Integra 2007

Brand Example 1
Open an existing Brand and output it's properties – it is assumed ADB has been initialised

```
‘  Create an Brand entity
Dim oBrand As IADBBrand
Set oBrand = New IADBBrand

‘ Set the Project and Name
oBrand.Project = “MYPROJ”
oBrand.Name = “HP”

‘ Open the Brand
oBrand.Open

‘ Ouput the Data
Debug.Print oBrand.Name, oBrand.Description , oBrand.RevisionDate

‘ Close
oBrand.Close
```

© Copyright Integra 2007

Brand Example 2
Create new Brand or modify an existing Brand and set its properties – it is assumed ADB has been initialised

```
‘  Create an Brand entity
Dim oBrand As IADBBrand
Set oBrand = New IADBBrand

‘ Set the Project and Name
oBrand.Project = “MYPROJ”
oBrand.Name = “HP”

‘ Open the Brand and set the Description and other properties
oBrand.Open
oBrand.Description = “Hewlett Packard”

‘ Save the Brand and set the revision date to the current date
oBrand.Save True, True

‘ Close
oBrand.Close
```

© Copyright Integra 2007

IADBSupplier

Properties	Data Type	Description
Description	String unlimited	Supplier Title
Name	String 10 chars	Supplier ADB Code. Must be set before calling Open or Save
Notes	String unlimited	Descriptive textual notes
Project	String 8 chars	ADB Project Code. Must be set before calling Open or Save
RevisionDate	Date/Time	Date/Time Assembly last saved
UserData(n)	String unlimited	Set n=0,1,2,3,4 for UserData1..UserData5

Methods	Description
Function AsXml() As String	Returns the Supplier definition as an XML string see Appendix for definition.
Sub Close()	Reset all Supplier attributes. The Supplier must be Open.
Sub Delete()	Delete the Supplier
Function FromXml(lpszXml As String) As String	Creates an Supplier entity from an Supplier definition as an XML string see Appendix for definition.
Function GetLastError() As Object	Returns an IADB_MsgQueue object holding the full error stack for the last error
Function IsDirty() As Boolean	True if the Supplier has been modified since it was Opened
Function IsOpen() As Boolean	True if the Supplier has been Opened
Sub Open()	Read the Supplier details from the ADB database. The Name & Project properties must be set before calling this method
Sub Save(UpdateRevisionDate As Boolean, OverWrite As Boolean)	Write the Supplier to the database
Sub SaveAs(Project As String, Name As String, UpdateRevisionDate As Boolean, OverWrite As Boolean)	Copy this Supplier, optionally to a different Project. The Supplier must be Open, and must not have been modified (i.e. IsDirty = false)

© Copyright Integra 2007

Supplier Example 1
Open an existing Supplier and output it's properties – it is assumed ADB has been initialised

```
‘  Create an Supplier entity
Dim oSupplier As IADBSupplier
Set oSupplier = New IADBSupplier

‘ Set the Project and Name
oSupplier.Project = “MYPROJ”
oSupplier.Name = “AGILENT”

‘ Open the Supplier
oSupplier.Open

‘ Ouput the Data
Debug.Print oSupplier.Name, oSupplier.Description , oSupplier.RevisionDate

‘ Close
oSupplier.Close
```

© Copyright Integra 2007

Supplier Example 2
Create new Supplier or modify an existing Supplier and set its properties – it is assumed ADB has been initialised

```
‘  Create an Supplier entity
Dim oSupplier As IADBSupplier
Set oSupplier = New IADBSupplier

‘ Set the Project and Name
oSupplier.Project = “MYPROJ”
oSupplier.Name = “AGILENT”

‘ Open the Supplier and set the Description and other properties
oSupplier.Open
oSupplier.Description = “Agilent”

‘ Save the Supplier and set the revision date to the current date
oSupplier.Save True, True

‘ Close
oSupplier.Close
```

© Copyright Integra 2007

IADBModel

Properties	Data Type	Description
Brand	String 10 chars	ADB Brand Code
Description	String unlimited	Description of model
ModelCode	String 255 chars	Manufacturers Model code/identifier typically as appears on the equipement
ModelType	String 255 chars	Manufacturers Model type where appropriate
Name	String 10 chars	Model ADB Code. Must be set before calling Open or Save
Price	Double	Price ex VAT
Project	String 8 chars	ADB Project Code. Must be set before calling Open or Save
RevisionDate	Date/Time	Date/Time Model last saved
Supplier	String 8 chars	ADB Supplier Code
UserData(n)	String unlimited	Set n=0,1,2,3,4 for UserData1..UserData5

Methods	Description
Function AsXml() As String	Returns the Model definition as an XML string see Appendix for definition.
Sub Close()	Reset all Model attributes. The Activity must be Open.
Sub Delete()	Delete the Model
Function FromXml(lpszXml As String) As String	Creates an Model entity from an Model definition as an XML string see Appendix for definition.
Function GetLastError() As Object	Returns an IADB_MsgQueue object holding the full error stack for the last error
Function IsDirty() As Boolean	True if the Model has been modified since it was Opened
Function IsOpen() As Boolean	True if the Model has been Opened
Sub Open()	Read the Model details from the ADB database. The Name & Project properties must be set before calling this method
Sub Save(UpdateRevisionDate As Boolean, OverWrite As Boolean)	Write the Model to the database
Sub SaveAs(Project As String, Name As String, UpdateRevisionDate As Boolean, OverWrite As Boolean)	Copy this Model, optionally to a different Project. The Model must be Open, and must not have been modified (i.e. IsDirty = false)

© Copyright Integra 2007

Model Example 1

```
‘ Open an existing Model and output it’s properties – it is assumed ADB has been initialised

‘  Create a Model entity
Dim oModel As IADBModel
Set oModel = New IADBModel

‘ Set the Project and Name
oModel.Project = “MYPROJ”
oModel.Name = “HP4537A”

‘ Open the Model
oModel.Open

‘ Ouput the Data
Debug.Print oModel.Name, oModel.ModelCode, oModel.ModelType, _
                oModel.Description , oModel.RevisionDate, _
                oModel.Brand , oModel.Supplier, oModel.Price

‘ Close
oModel.Close
```

© Copyright Integra 2007

Model Example 2

```
‘ Create new Model or modify an existing Model and set its properties – it is assumed ADB has been initialised

‘  Create a Model entity
Dim oModel As IADBModel
Set oModel = New IADBModel

‘ Set the Project and Name
oModel.Project = “MYPROJ”
oModel.Name = “HP4537A”

‘ Open the Model
oModel.Open

‘ Set the properties
oModel.ModelCode = “4537”
oModel.ModelType = “A”
oModel.Description = “Defribrillator”
oModel.Brand = “HP”
oModel.Supplier = “AGILENT”
oModel.Price =2000.0

‘ Save a set the revision date
oModel.Save, True, True

‘ Close
oModel.Close
```

© Copyright Integra 2007

Entity Browse Lists

Entity Lists termed browse lists are available for all the following ADB Entities:

IADBProjectBrowseList
IADBDepartmentBrowseList
IADBRoomBrowseList
IADBAssemblyBrowseList
IADBComponentBrowseList
IADBActivityBrowseList
IADBModelBrowseList
IADBBrandBrowseList
IADBSupplierBrowseList

The following properties and methods are common to all entities:

Commom Properties

Properties	Data Type	Description
Count	Long	The number of entries in the list
Description	String unlimited	Entity Title
Filter	Object	An IADBConditionList
Name	String 10 chars	ADB Entity Code
OrderBy	Object	Set a list of fields (actually an IADB_FieldList) to order the entries in the list. Must be set before calling Open. The field list should not contain references to memo fields
RevisionDate	Date/Time	The last modification date/time of the entity identified by the current list element

Entity Specific Properties

IADBDepartmentBrowseList

Properties	Data Type	Description
Project	String 8 chars	ADB Project Code. Must be set before calling Open
Ordering	String 1 char	Department Ordering scheme (A – alpha, S – Specified, R – Room ordered)

IADBRoomBrowseList

Properties	Data Type	Description
Area	Double	The Area of the Room identified by the current list element
Project	String 8 chars	ADB Project Code. Must be set before calling Open

© Copyright Integra 2007

IADBAssemblyBrowseList
IADBActivityBrowseList

Properties	Data Type	Description
Class	String 10 chars	Numeric Class
Project	String 8 chars	ADB Project Code. Must be set before calling Open

IADBComponentBrowseList

Properties	Data Type	Description
Class	String 10 chars	Numeric Class
Project	String 8 chars	ADB Project Code. Must be set before calling Open
Cost	Double	The cost (new) of the Component identified by the current list element
TransferCost	Double	The transfer cost of the Component identified by the current list element

IADBBrandBrowseList
IADBSupplierBrowseList

Properties	Data Type	Description
Brand	String 10 chars	ADB Brand Code
Project	String 8 chars	ADB Project Code. Must be set before calling Open

IADBModelBrowseList

Properties	Data Type	Description
Class	String 10 chars	Numeric Class
Cost	Double	The Cost/Price ex VAT identified by the current list element
ModelCode	String 255 chars	Manufacturers Model code/identifier typically as appears on the equipement
ModelType	String 255 chars	Manufacturers Model type where appropriate
Project	String 8 chars	ADB Project Code. Must be set before calling Open
Supplier	String 8 chars	

Common Methods

Methods	Description
Sub Close()	Reset all BrowseList attributes. The BrowseList must be Open.
Function GetLastError() As Object	Returns an IADB_MsgQueue object holding the full error stack for the last error
Function IsDirty() As Boolean	True if the BrowseList l has been modified since it was Opened
Function IsOpen() As Boolean	True if the BrowseList l has been Opened

© Copyright Integra 2007

Sub Open()	Read the BrowseList l details from the ADB database. The Name & Project properties must be set before calling this method
Sub Move(iPos As Long)	Move to the specified position in the list
Sub MoveFirst()	Move to first item in the list
Sub MoveNext()	Move to next item in the list

Browse List Example 1

```
‘ Display a Browse List of all Rooms in a Project– it is assumed ADB has been initialised

‘  Create a Room BrowseList entity
Dim oRoomBL As IADBRoomBrowseList
Set oRoomBL = New IADBRoomBrowseList

‘ Set the Project
oRoomBL.Project = “MYPROJ”

‘ Open the Browse Listl
oRoomBL.Open

‘ FOR EACH ROOM…
oRoomBL.MoveFirst
While Not oRoomBL.IsEOF
        Debug.Print oRoomBL.Name, oRoomBL.Description, _
                oRoomBL.Area, oRoomBL.RevisionDate
        oRoomBL.MoveNext
Wend

‘ Close
oRoomBL.Close
```

© Copyright Integra 2007

Browse List Example 2

```
   ' Display a Filtered Browse List of all Rooms in a Project– it is assumed ADB has been
initialised

   '  Create a Room BrowseList entity
   Dim oRoomBL As IADBRoomBrowseList
   Set oRoomBL = New IADBRoomBrowseList

   ' Set the Project
   oRoomBL.Project = "MYPROJ"

   ' Create a filter using an IADBConditionList
   ' to display all Rooms with a code beginning with C.
   ' The first argument of the Add method is the Public name of the filter field
   ' as given in Appendix 2
   Dim oCondList As IADBConditionList
   Set oCondList = New IADBConditionList
   oCondList.Add "Code", "like", "C%"
   Set oRoomBL.Filter = oCondList


   ' Open the Browse List
   oRoomBL.Open

   ' FOR EACH ROOM…
   oRoomBL.MoveFirst
   While Not oRoomBL.IsEOF
      Debug.Print oRoomBL.Name, oRoomBL.Description, _
      oRoomBL.Area, oRoomBL.RevisionDate
      oRoomBL.MoveNext
   Wend

   ' Close
   oRoomBL.Close
```

© Copyright Integra 2007

Browse List Example 3

```
' Display a Filtered Browse List of all Group 1 Components in a Project
' - it is assumed ADB has been initialised

'  Create a Component BrowseList entity
Dim oCompBL As IADBComponentBrowseList
Set oCompBL = New IADBComponentBrowseList

' Set the Project
oCompBL.Project = "ADB204"

' Create a filter using an IADBConditionList to display all Components in Group 1
' The first argument of the Add method is the Public name of the filter field
' as given in Appendix 2
Dim oCondList As IADBConditionList
Set oCondList = New IADBConditionList
oCondList.Add "Group", "=", "1"
Set oCompBL.Filter = oCondList


' Open the Browse List
oCompBL.Open

' FOR EACH COMPONENT…
oCompBL.MoveFirst
While Not oCompBL.IsEOF
   Debug.Print oCompBL.Name, oCompBL.Description, _
   oCompBL.Group, oCompBL.Class, , oCompBL.Cost
   oCompBL.MoveNext
Wend

' Close
oCompBL.Close
```

© Copyright Integra 2007

Browse List Example 4

```
' Display a Filtered Browse List of Group 2 and 3 Components in a Project
' - it is assumed ADB has been initialised

'  Create a Component BrowseList entity
Dim oCompBL As IADBComponentBrowseList
Set oCompBL = New IADBComponentBrowseList

' Set the Project
oCompBL.Project = "MYPROJ"

' Create a filter using an IADBConditionList to display all Group 2 or 3 Components
' of the filter value string (‘2’,’3’) used by the "in" clause
' The first argument of the Add method is the Public name of the filter field
' as given in Appendix 2
Dim oCondList As IADBConditionList
Set oCondList = New IADBConditionList
Dim sValue As String
sValue = "(‘2’,’3’)"
oCondList.Add "Group", "in", sValue
Set oCompBL.Filter = oCondList

' Open the Browse List
oCompBL.Open

' FOR EACH COMPONENT…
oCompBL.MoveFirst
While Not oCompBL.IsEOF
   Debug.Print oCompBL.Name, oCompBL.Description, _
   oCompBL.Group, oCompBL.Class, oCompBL.Cost
   oCompBL.MoveNext
Wend

' Close
oCompBL.Close
```

© Copyright Integra 2007

Entity Schedules

IADBRoomSchedule
IADBSequencedRoomSchedule
IADBOrderedRoomSchedule
IADBAssemblySchedule
IADBComponentSchedule
IADBActivitySchedule
IADBModelSchedule

The following properties and methods are common to all entities:

Commom Properties

Properties	Data Type	Description
Count	Long	The number of entries in the list
Description	String unlimited	Entity Title
Filter	Object	An IADBConditionList
Name	String 10 chars	ADB Entity Code
OrderBy	Object	Set a list of fields (actually an IADB_FieldList) to order the entries in the list. Must be set before calling Open. The field list should not contain references to memo fields
RevisionDate	Date/Time	The last modification date/time of the entity identified by the current list element

Entity Specific Properties

IADBRoomSchedule

Properties	Data Type	Description
Area	Double	The area of the Room identified by the current list element
NewCount	Long	The quantity of instances of the Room identified by the current Schedule element

IADBSequencedRoomSchedule

Properties	Data Type	Description
Quantity	Long	The quantity of instances of the Room identified by the current Schedule element
Sequence	String	Room Sequence Number

© Copyright Integra 2007

IADBOrderedRoomSchedule

Properties	Data Type	Description
RoomNumber	String	Room number

IADBAssemblySchedule

Properties	Data Type	Description
NewCount	Long	The quantity of instances of the Assembly identified by the current Schedule element

IADBComponentSchedule

Properties	Data Type	Description
Class	String	The ADB Classification of the Component identified by the current Schedule element
Cost	Double	The cost (new) of the Component identified by the current Schedule element
Group	String	The ADB Group code of the Component identified by the current Schedule element
NewCount	Long	The quantity of new instances of the Component identified by the current Schedule element
NSVCode	String	The National Supplies or Alternative code of the Component identified by the current Schedule element
Total	Long	The total quantity of new and transferred instances of the Component identified by the current Schedule element
TransferCost	Double	The cost (transferred) of the Component identified by the current Schedule element
TransferCount	Long	The quantity of transferred instances of the Component identified by the current Schedule element

IADBModelSchedule

Properties	Data Type	Description
Brand	String unlimited	Brand Title
BrandCode	String 10 chars	ADB Brand Code
Price	Double	Cost/Price ex VAT
ModelCode	String 255 chars	Manufacturers Model code/identifier typically as appears on the equipement
ModelType	String 255 chars	Manufacturers Model type where appropriate
Supplier	String unlimited	Supplier Title
SupplierCode	String 8 chars	ADB Supplier Code

Common Methods

Methods	Description
Sub Close()	Reset all Schedule attributes. The BrowseList must be Open.

© Copyright Integra 2007

Function GetLastError() As Object	Returns an IADB_MsgQueue object holding the full error stack for the last error
Function IsDirty() As Boolean	True if the Schedule has been modified since it was Opened
Function IsOpen() As Boolean	True if the Schedule has been Opened
Sub Open()	Read the Schedule details from the ADB database. The Name & Project properties must be set before calling this method
Sub Move(iPos As Long)	Move to the specified position in the list
Sub MoveFirst()	Move to first item in the list
Sub MoveNext()	Move to next item in the list

Entity Specific Properties

IADBModelSchedule

Methods	**Description**
Sub AddModel(sName As String, sDescription As String, sModelCode As String, sModelType As String, sBrand As String, sBrandCode As String, sSupplier As String, sSupplierCode As String, fPrice As Double, dDate As Date)	As support for Models etc. was added at a later date the an the Add method has been assigned to the ModelSchedule tarther than the Component Entity. sDescription, sModelType, sBrand, sSupplier, fPrice and date are not used and should be assigned place holder values e.g. “” for strings 0.0 for reals and Now for dates.
Sub Delete (sCode as String)	Delete the Model specified by its code from the schedule.
Sub Save	Save the Model schedule to the database

© Copyright Integra 2007

ScheduleExample1

```
' Output the Component Schedule for a Room
' - it is assumed ADB has been initialised

' Open the Room
Dim oRoom As IADBRoom
Set oRoom = New IADBRoom
oRoom.Project = "DEMO1"
oRoom.Name = "B0303"
oRoom.Open

' Get the Rooms's Component Schedule
Dim oSchedule As IADBComponentSchedule
Set oSchedule = oRoom.ComponentSchedule
oSchedule.Open

' FOR EACH item in the Schedule
oSchedule.MoveFirst
While Not oSchedule.IsEOF

   '  Output
   Debug.Print oSchedule.Name, oSchedule.Description, _
            oSchedule.NewCount, oSchedule.Group

   '  Next Item
   oSchedule.MoveNext
Wend

' Close the Room and any open schedules
oRoom.Close
```

ScheduleExample2

```
' Output the filtered Component Schedule for a Department
' - it is assumed ADB has been initialised

' Open the Department
Dim oDepartment As IADBDepartment
Set oDepartment = New IADBDepartment
oDepartment.Project = "DEMO1"
oDepartment.Name = "INP01"
oDepartment.Open

' Get the Department's Component Schedule
Dim oSchedule As IADBComponentSchedule
Set oSchedule = oDepartment.ComponentSchedule

' Create a filter using an IADBConditionList
' to display all Group 2 or 3 Components
' using an "in clause" ('2','3')
Dim oCondList As IADBConditionList
Set oCondList = New IADBConditionList
Dim sValue As String
```

© Copyright Integra 2007

```
    sValue = "('2','3')"
    oCondList.Add "Group", "in", sValue
    Set oSchedule.Filter = oCondList
    oSchedule.Open

    ' FOR EACH item in the Schedule
    oSchedule.MoveFirst
    While Not oSchedule.IsEOF

        '  Output
        Debug.Print oSchedule.Name, oSchedule.Description, _
                oSchedule.NewCount, oSchedule.Group

        '  Next Item
        oSchedule.MoveNext
    Wend

    ' Close the Department and any open schedules
    oDepartment.Close

ScheduleExample3

    ' Output the Room Schedule for an aplhanumeric ordered Department
    ' - it is assumed ADB has been initialised

    '  Initailsie ADB and login
    Dim oADB As New IADB
    oADB.Initialise "manager", "adb"

    ' Open the Department
    Dim oDepartment As IADBDepartment
    Set oDepartment = New IADBDepartment
    oDepartment.Project = "DEMO1"
    oDepartment.Name = "INP01"
    oDepartment.Open

    ' Get the Department's Room Schedule
    Dim oSchedule As IADBRoomSchedule
    Set oSchedule = oDepartment.RoomSchedule
    oSchedule.Open

    ' FOR EACH item in the Schedule
    oSchedule.MoveFirst
    While Not oSchedule.IsEOF

        '  Output
        Debug.Print oSchedule.Name, oSchedule.Description, _
                oSchedule.NewCount

        '  Next Item
        oSchedule.MoveNext
    Wend

    ' Close the Department and any open schedules
```

© Copyright Integra 2007

```
    oDepartment.Close
```

ScheduleExample4

```
    ' Output the Room Schedule for an aplhanumeric ordered, room ordered
    ' or sequenced Department
    ' - it is assumed ADB has been initialised

    ' Open the Department
    Dim oDepartment As IADBDepartment
    Set oDepartment = New IADBDepartment
    oDepartment.Project = "DEMO1"
    oDepartment.Name = "INP01"
    oDepartment.Open

    ' Get the Department's Room Schedule
    ' Note use of late binding as the type of schedule is not known
    ' until the Department is opened
    Dim oSchedule As Object
    If oDepartment.Order = "S" Then
       Set oSchedule = oDepartment.SequencedRoomSchedule
    ElseIf oDepartment.Order = "R" Then
       Set oSchedule = oDepartment.OrderedRoomSchedule
    Else
       Set oSchedule = oDepartment.RoomSchedule
    End If
    oSchedule.Open

    ' FOR EACH item in the Schedule
    oSchedule.MoveFirst
    While Not oSchedule.IsEOF

       '  Output
       If oDepartment.Order = "S" Then
          Debug.Print oSchedule.Name, oSchedule.Description, oSchedule.Quantity
       ElseIf oDepartment.Order = "R" Then
          Debug.Print oSchedule.Name, oSchedule.Description, oSchedule.RoomNumber
       Else
          Debug.Print oSchedule.Name, oSchedule.Description, oSchedule.NewCount
       End If

       '  Next Item
       oSchedule.MoveNext
    Wend

    ' Close the Department and any open schedules
    oDepartment.Close
```

Error handling

ErrorHandlingExample1

© Copyright Integra 2007

Basic Error Handling

```
    On Error GoTo ErrorHandlingExample1_err:

    '  Initailsie ADB and login
    Dim oADB As New IADB
    oADB.Initialise "manager", "xxx"

ErrorHandlingExample1_err:

    MsgBox "Application Error:" & Err.Description, vbCritical + vbOKOnly
```

© Copyright Integra 2007

ErrorHandlingExample2 – Using the ADB Message Queue

```
' Initailsie ADB and login
Dim oADB As New IADB

' Ignore the error and trap afterwards
On Error Resume Next
oADB.Initialise "manager", "xxx"

' Err.Number will be non zero
If Err.Number <> 0 Then
   Dim oMsgQueue As IADBMsgQueue
   Set oMsgQueue = oADB.GetLastError
   Dim i As Integer
   oMsgQueue.MoveFirst
   For i = 0 To oMsgQueue.Count - 1
      Debug.Print oMsgQueue.Summary
      oMsgQueue.MoveNext
   Next
End If
```

ErrorHandlingExample3 – Using the ADB Message Queue version 3

```
' Initailsie ADB and login
Dim oADB As New IADB

' Ignore the error and trap afterwards
On Error Resume Next
oADB.Initialise "manager", "xxx"

' Report any errors
If Err.Number <> 0 Then
   ReportError oADB
End If

Public Sub ReportError(oADBObject As Object)

   Dim oMsgQueue As IADBMsgQueue
   Set oMsgQueue = oADBObject.GetLastError
   Dim i As Integer
   oMsgQueue.MoveFirst
   For i = 0 To oMsgQueue.Count - 1
      Debug.Print oMsgQueue.Summary
      oMsgQueue.MoveNext
   Next

End Sub
```

© Copyright Integra 2007

ErrorHandlingExample4 – Using the ADB Message Queue version 4

```
' Initailsie ADB and login
Dim oADB As New IADB

' Ignore the error and trap afterwards
On Error Resume Next
oADB.Initialise "manager", "adb"

' Report any errors
If Err.Number <> 0 Then
   ReportError oADB
End If

Dim oRoom As IADBRoom
Set oRoom = New IADBRoom
oRoom.Project = "XXX"
oRoom.Open

' Report any errors
If Err.Number <> 0 Then
   ReportError oRoom
End If
```

© Copyright Integra 2007

Message Based Processing

Message based processing provides an alternative method of accessing and updating ADB data competely through XML. It is the only method of using the ADB API with web services and AutoCAD.

Three API interfaces are avaiable:

ADBOLE32.DLL	General use with MS Excel, Access, Word etc
OLEACAD17.DLL	for AutoCAD 2004/5 and 6 and ADT equivalents
OLEACAD16.DLL	for AutoCAD 207/8 and ADT equivalents

Only three methods on the ADB Application object are used

IADB

Methods	**Description**
Sub Initialise(UserName As String, Password As String)	Initialise the ADB application and logon to ADB
Sub UnInitialise	Close all database connections. Must be called once before application terminates
Function ProcessMessage(Msg As Integer, K1 As String, K2 As String, Param1 As String, Param2 As String, Param3 As String) As String	Messaging support – used for Web Services and AutoCAD Msg – Message number see Appendix K1 – Public key (username) K2 – Private key (password) Param1,2,3 - Parameters

Messaging Example 1

‘ Project Department list

‘ Initialise and login to ADB
Dim oADB As New IADB
oADB.Initialise “manager”,”adb”
‘ Process the message
Dim sXML As String
sXml = oADB.ProcessMessage(8,”manager”,”adb”,”MYPROJ”,””,””)
Debug.Print sXML
‘ Uninitialise
oADB.UnInitialise

Messaging Example 2

```
' Extracting data from a Project Department Browse List returned as XML
' – it is assumed ADB has been initialised
```

© Copyright Integra 2007

```
   ' Process the message - note the password must be prepended with [PLAINTEXT]
   Dim sXML As String
   sXML = oADB.ProcessMessage(8, "manager", "[PLAINTEXT]adb", "DEMO1", "", "")

   ' Use Document Object Model (DOM) parser to extract the data
   ' This requires a reference to DOM parser - Microsoft XML, v3.0 will do
   Dim oDOMDoc As DOMDocument
   Dim oXMLNodeList As IXMLDOMNodeList
   Dim oXMLNode As IXMLDOMNode
   Dim oXMLChildNode As IXMLDOMNode

   ' Load the XML returned into the DOM document
   Set oDOMDoc = New DOMDocument
   oDOMDoc.loadXML sXML
   Set oXMLNodeList = oDOMDoc.selectNodes("ExplorerGridList/ExplorerGridListRow")
   For Each oXMLNode In oXMLNodeList
      Dim sCode As String, sDescription As String
      Set oXMLChildNode = oXMLNode.selectSingleNode("code")
      sCode = oXMLChildNode.Text
      Set oXMLChildNode = oXMLNode.selectSingleNode("description")
      sDescription = oXMLChildNode.Text
      Debug.Print sCode, sDescription
   Next
```

Messaging Example 3

```
   ' Room details – it is assumed ADB has been initialised

   ' Process the message - note the password must be prepended with [PLAINTEXT]
   Dim sXML As String
   sXML = oADB.ProcessMessage(218, "manager", "[PLAINTEXT]adb", "DEMO1",
"B0303", "")
```

© Copyright Integra 2007

Messaging Example 4

```
   ' Extracting data from a Room returned as XML – it is assumed ADB has been initialised

   ' Process the message - note the password must be prepended with [PLAINTEXT]
   Dim sXML As String
   sXML = oADB.ProcessMessage(218, "manager", "[PLAINTEXT]adb", "DEMO1",
"B0303", "")

   ' Use Document Object Model (DOM) parser to extract the data
   ' This requires a reference to DOM parser - Microsoft XML, v3.0 will do
   Dim oDOMDoc As DOMDocument
   Dim oXMLNode As IXMLDOMNode
   Dim oXMLAtts As IXMLDOMNamedNodeMap

   ' Load the XML returned into the DOM document
   Set oDOMDoc = New DOMDocument
   oDOMDoc.loadXML sXML

   ' Get the main Entity element this will enable retrieval of the attributes
   Set oXMLNode = oDOMDoc.selectSingleNode("ADBEntity")
   Dim sCode As String

   ' Get the attributes and extract the named item
   Set oXMLAtts = oXMLNode.Attributes
   sCode = oXMLAtts.getNamedItem("EntityName").Text

   ' Get the Description element this will enable retrieval of the attributes
   Set oXMLNode = oDOMDoc.selectSingleNode("ADBEntity/Description")
   Dim sDescription As String
   sDescription = oXMLNode.Text
   Debug.Print sCode, sDescription
```

Messaging Example 5

```
   ' Filtered Project Room List – it is assumed ADB has been initialised

   ' Process the message - note the password must be prepended with [PLAINTEXT]
   '   P1 is the Project code
   '   P2 is the Filter string e.g. [Code like 'C%']
   Dim sFilter As String
   sFilter = "[Code like 'C%']"
   Dim sXML As String
   sXML = oADB.ProcessMessage(14, "manager", "[PLAINTEXT]adb",
                                    "ADB204", sFilter, "")
```

© Copyright Integra 2007

Messaging Example 6

```
    ' Room, Assembly or Component graphic definition as XML – it is assumed ADB has been
initialised

    ' Process the message - note the password must be prepended with [PLAINTEXT]
    '   P1 Project code
    '   P2 Room or Assembly code
    '   P3 View Code -3,-P,-E,SE,TE,RE
    Dim sXML As String
    sXML = oADB.ProcessMessage(500, "manager", "[PLAINTEXT]adb", "ADB204",
"EA1631", "-P")
```

© Copyright Integra 2007

Messages List

Key:
PC – Project Code
EC – Entity Code
REC – Replacement Entity Code
CS – Connect string
e.g. PROVIDER=MICROSOFT.JET.OLEDB.4.0;DATA SOURCE = C:\PROGRAM FILES\DHEFD\ACTIVITY DATABASE\PROJECTS\ADB204.MDB

MV- Project Major Version future use
mV-Project Minor Version future use
FS-Filter string
XMLA-Audit Definition as XML see Appendix 1
XMLED-Entity Definition as XML see Appendix 1
XMLI-Interface Definition as XML see Appendix 1
XMLGC-Graphics Container Definition as XML see Appendix 1
XMLGE-Graphics Enclosure Definition as XML see Appendix 1
XMLGP-Graphics Primitive Definition as XML see Appendix 1
UR-Yes to Update Revision Date

Message Type	Param1	Param2	Param3	Msg	Description
Standard					
	Username	Password		-1	Remote login
	EC	PC		-3	Audit history for an entity
	ID	PC		-4	Audit record
	PC	XMLA		-5	Add new audit record
	PC			-6	Audit consequence browse list
	PC	EC		50	Find the entity type of the specified object
	PC	EC		51	Fetch the entity properties
	PC			52	Unregister a project
	CS	MV	mV	53	Register a project
Tree view					
				100	List of servers
				101	List of projects
	PC	FS		102	List of departments
	PC	FS		103	Department room schedule
	PC	FS		104	Room assembly schedule
	PC	FS		105	Assembly component schedule
Project-level list & grid views					
				1	List-view list of servers
				2	List-view list of projects
	PC	FS		3	List-view list of departments
	PC	FS		6	List-view list of rooms

© Copyright Integra 2007

PC	FS		7	List-view list of assemblies
PC	FS		16	List-view list of assemblies
PC	FS		18	List-view list of activities
PC	FS		20	List-view list of models
PC	FS		22	List-view list of suppliers
PC	FS		24	List-view list of brands
PC	FS		54	List-view list of audit records
			12	Grid-view list of servers
			13	Grid-view list of projects
PC	FS		8	Grid-view list of departments
PC	FS		14	Grid-view list of rooms
PC	FS		15	Grid-view list of assemblies
PC	FS		17	Grid-view list of components
PC	FS		19	Grid-view list of activities
PC	FS		21	Grid-view list of models
PC	FS		23	Grid-view list of suppliers
PC	FS		25	Grid-view list of brands
PC	FS		55	Grid-view list of audit records
PC	FS		257	Grid-view list of department classes
PC	FS		258	Grid-view list of room classes
PC	FS		259	Grid-view list of assembly classes
PC	FS		260	Grid-view list of component classes
PC	FS		261	Grid-view list of activity classes
PC	FS		262	Grid-view list of component groups
PC	FS		263	Grid-view list of floor design character
PC	FS		264	Grid-view list of wall design character
PC	FS		265	Grid-view list of ceiling design character
PC	FS		266	Grid-view list of component installer
PC	FS		267	Grid-view list of component supplier
PC	FS		268	Grid-view list of component user1
PC	FS		269	Grid-view list of component user2
PC	FS		270	Grid-view list of CISfB layers
PC	FS		271	Grid-view list of layer discilines
PC	FS		272	Grid-view list of user layers
PC	FS		273	Grid-view list of lighting grades
PC	FS		274	Grid-view list of layer services
PC	FS		275	Grid-view list of component views

Schedules: list & grid views

PC	EC	FS	4	Departmental room schedule (order does not matter)
PC	EC	FS	5	Departmental assembly schedule
PC	EC	FS	11	Departmental component schedule
PC	EC	FS	27	Departmental room schedule
PC	EC	FS	9	Departmental assembly schedule

© Copyright Integra 2007

	PC	EC	FS	26	Departmental component schedule
	PC	EC	FS	28	Room assembly schedule
	PC	EC	FS	29	Room component schedule
	PC	EC	FS	38	Room activity schedule
	PC	EC	FS	30	Room assembly schedule
	PC	EC	FS	31	Room component schedule
	PC	EC	FS	39	Room activity schedule
	PC	EC	FS	32	Assembly assembly schedule
	PC	EC	FS	33	Assembly component schedule
	PC	EC	FS	40	Assembly activity schedule
	PC	EC	FS	34	Assembly assembly schedule
	PC	EC	FS	35	Assembly component schedule
	PC	EC	FS	41	Assembly activity schedule
	PC	EC	FS	36	Component model schedule
	PC	EC	FS	37	Component model schedule
	PC	EC	FS	42	Room nested schedule as XML
Entity					
	PC			200	User has write access to project
	UR		XMLED	201	Update Project
	UR		XMLED	202	Update Department
	UR		XMLED	203	Update Room
	UR		XMLED	204	Update Assembly
	UR		XMLED	205	Update Component
	UR		XMLED	206	Update Activity
	UR		XMLED	207	Update Model
	UR		XMLED	208	Update Supplier
	UR		XMLED	209	Update Brand
			XMLI	210	Delete Department Rooms
			XMLI	211	Delete Room Assemblies, Components and Activities
			XMLI	212	Delete Assembly Sub-Assemblies, Components and Activiti
			XMLI	213	Delete Component Models
			XMLI	214	Add Department Rooms
			XMLI	215	Add Room Assemblies, Components and Activities
			XMLI	216	Add Assembly Sub-Assemblies, Components and Activities
			XMLI	217	Add Component Models
	PC	EC		218	Get the full entity properties – USE FOR ALL ENTITIES EXCEPT ACTIVITES
	PC	EC		219	Check if any entity with this code already exists in the Projec
				220	Update and entities properties and schedules
	PC	EC		221	Check if Project with this code exists
	PC	EC		222	Check if a Department with this code exists in the Project
	PC	EC		223	Check if a Room with this code exists in the Project
	PC	EC		224	Check if a Assembly with this code exists in the Project

© Copyright Integra 2007

	PC	EC		225	Check if a Component with this code exists in the Project
	PC	EC		226	Check if an Activity with this code exists in the Project
	PC	EC		227	Check if a Model with this code exists in the Project
	PC	EC		228	Check if a Supplier with this code exists in the Project
	PC	EC		229	Check if a Brand with this code exists in the Project
			XMLED	230	Create a new Project
			XMLED	231	Create a new Department
			XMLED	232	Create a new Room
			XMLED	233	Create a new Assembly
			XMLED	234	Create a new Component
			XMLED	235	Create a new Activity
			XMLED	236	Create a new Model
			XMLED	237	Create a new Supplier
			XMLED	238	Create a new Brand
			XMLED	239	Delete a Project
			XMLED	240	Delete a Department
			XMLED	241	Delete a Room
			XMLED	242	Delete an Assembly
			XMLED	243	Delete a Component
			XMLED	244	Delete an Activity
			XMLED	245	Delete a Model
			XMLED	246	Delete a Supplier
			XMLED	247	Delete a Brand
			XMLI	248	Save a Project to a new database
			XMLI	249	Save a Department to a new code and/or Project
			XMLI	250	Save a Room to a new code and/or Project
			XMLI	251	Save an Assembly to a new code and/or Project
			XMLI	252	Save a Component to a new code and/or Project
			XMLI	253	Save an Activity to a new code and/or Project
			XMLI	254	Save a Model to a new code and/or Project
			XMLI	255	Save a Supplier to a new code and/or Project
			XMLI	256	Save a Brand to a new code and/or Project
	PC	EC		276	Fetch the activities properties
Graphics specific					
	PC	EC	VC	500	Graphics Container as XML
	PC	EC	VC	501	Graphics Block as XML
	PC	EC	VC	502	Graphics Primitive as XML
	PC	EC	XMLGC	503	Save the Graphics Container
	PC	EC	XMLGP	504	Save the Graphics Primitive
	PC	EC		505	Enclosure as XML
	PC	EC	XMLGE	506	Save the Enclosure
Add-On specific					
	PC	EC		600	Delete child rooms from deplk
	PC	EC		601	Delete child assemblies from deplk
	PC	EC		602	Delete child components from deplk
	PC	EC	REC	603	Replace child rooms in deplk

© Copyright Integra 2007

PC	EC	REC	604	Replace child assemblies in deplk
PC	EC	REC	605	Replace child components in deplk
PC	DATE		606	Set the revision date of all departments
PC	DATE		607	Set the revision date of all rooms
PC	DATE		608	Set the revision date of all assemblies
PC	DATE		609	Set the revision date of all components
PC	DATE		610	Set the revision date of all activities

© Copyright Integra 2007

Appendix 1

Entity XML Definitions

Project

```
<ADBEntity EntityName='ADB204' Project='ICLDBX ADB Meta-Database' EntityType='P'
RevisionDate='14/08/2007' Issue='' Path=' C:\PROGRAM FILES\DHEFD\ACTIVITY
DATABASE\PROJECTS\ADB204.MDB' Class=''
ConnectString='PROVIDER=MICROSOFT.JET.OLEDB.4.0;Data Source = C:\Program
Files\DHEFD\Activity Database\Projects\ADB204.mdb' ProjectType='A'>
        <Description>Activity Database Version 20.4 © Crown Copyright</Description>
        <Notes></Notes>
</ADBEntity>
```

Department

```
<ADBEntity EntityName='INP01' Project='DEMO1' EntityType='D'
RevisionDate='02/07/2007' Class='0400'>
        <Description>IN-PATIENT ACCOMMODATION</Description>
        <User1>User Data 1</User1>
        <User2></User2>
        <User3></User3>
        <User4></User4>
        <User5></User5>
        <Notes>Departmental Notes</Notes>
        <RoomOrder>A</RoomOrder>
</ADBEntity>
```

© Copyright Integra 2007

Room

<ADBEntity EntityName='B0303' Project='DEMO1' EntityType='R' RevisionDate='06/06/2005' Class='0102'>
<User1>AS, 06-06-05; Basin assembly SA1112 replaced by SA1244 - Sensor tap. Wardrobe positioned</User1>
<User2></User2>
<User3></User3>
<User4></User4>
<User5></User5>
<Notes>Space may required to accommodate use of hoist. Ceiling mounted hoist subject to local decision.
Storage of patient drug - see hospital policy.
Outlet, compressed air, medical is a project option.
Curtain rail for door vision panel is required.
</Notes>
<Description>Single bedroom: Adult acute
With clinical support. Relative overnight stay. Access to en-suite
</Description>
<RoomSpaceData Area='34.3' Height='2700'>
<SpaceNotes></SpaceNotes>
</RoomSpaceData>
<RoomPersonnel>1 x Patient
4 x Others
</RoomPersonnel>
<PlanningRelationships>Close to staff base.
Close to ancillary rooms.
Ward activity to be visible from room.
En-suite sanitary facilities.
</PlanningRelationships>
<RoomEnvironmentalDataAir WinterTemperature='21.0' SummerTemperature='0.0' MechanicalVentilationSupply='0.0' MechanicalVentilationExtract='0.0' MechanicalVentilationUnits='0' DustSpotEfficiency='0' RelativeHumidity='0' Arrestance='0'>
<HVAC></HVAC>
<TemperatureNotes></TemperatureNotes>
<MechanicalVentilationNotes></MechanicalVentilationNotes>
<FiltrationHumidityNotes></FiltrationHumidityNotes>
<RelativePressure></RelativePressure>
</RoomEnvironmentalDataAir>
<RoomEnvironmentalDataLighting ServiceIllumination='100.0' ServiceIlluminationNight='5.0' LocalIllumination='150.0' ColourRendering='Y'>
<Illumination></Illumination>
<StandbyLightingGradeNotes>B: Lighting of the level and quality one third to one half that provided by normal lighting.
Day Bed centre: A: Lighting of the level and quality equal or nearly equal to that provided by normal lighting. For local examination & inspection.
</StandbyLightingGradeNotes>
<ServiceIlluminationNotes>Floor. 200-400 Bed centre. 30-50 Bedhead. Areas for VDT's: See CIBSE Lighting Guide LG3 "The Visual Environment for Display Screen Use" Addendum 2001
</ServiceIlluminationNotes>
<ServiceIlluminationNightNotes>Floor. 1-5 Bed centre. 0.1 Bedhead. Evening (lux): 50 Bed centre.

© Copyright Integra 2007

```
        </ServiceIllluminationNightNotes>
        <LocalIllluminationNotes>Bedhead</LocalIllluminationNotes>
        <StandbyLightingGrade></StandbyLightingGrade>
        <ColourRenderingNotes>Not night &amp;amp; local</ColourRenderingNotes>
        </RoomEnvironmentalDataLighting>
        <RoomEnvironmentalDataNoise AcceptableSoundLevel='0.0' SpeechPrivacy='N'
        QualityNotTolerated='' PrivacyFactor='80.0' MechanicalServices='30.0'
        IntrusiveNoise='35.0'>
                <Acoustics></Acoustics>
                <NoiseNotes>Ref: HTM2045</NoiseNotes>
        </RoomEnvironmentalDataNoise>
        <RoomEnvironmentalDataSafety HotSurfaceTemperature
                ='43.0' HotWaterTemperature='0.0'>
                <Precautions></Precautions>
                <SafetyNotes></SafetyNotes>
        </RoomEnvironmentalDataSafety>
        <RoomEnvironmentalDataFire>
                <FireProtection></FireProtection>
                <Enclosure></Enclosure>
                <AutomaticDetection>Smoke</AutomaticDetection>
        </RoomEnvironmentalDataFire>
        <RoomDesignCharacter>
        <Flooring>Surface Finish (HTM 61): 3 i.e. Hard, impervious, jointless, smooth
        Cleaning Routine (HTM 61): To manufacturers recommendations
        </Flooring>
        <Walls>Surface Finish (HTM 56): 5
        Moisture Resistance (HTM 56): N i.e. Normal humidity.
        Cleaning Routine (HTM 56): To manufacturers recommendations
        </Walls>
        <Ceilings>Surface Finish (HTM 60): 5 i.e. Imperforate
        Moisture Resistance (HTM 60): N i.e. Normal Humidity
        Cleaning Routine (HTM 60): To manufacturers recommendations
        </Ceilings>
        <Doorsets>(HTM 58) Two sets of doors: 1x 1500mm, one &amp;amp; a half leaf, half
        glazed, obscurable; bed access. 1x 1000mm,s ingle leaf, plain flush; wheelchair access.
        </Doorsets>
        <Glazing>(HTM 57) Clear with privacy control
        </Glazing>
        <Hatch></Hatch>
        <Windows>(HTM 55) Clear, solar control, privacy control.
        </Windows>
        <DesignNotes></DesignNotes>
        </RoomDesignCharacter>
</ADBEntity>
```

Assembly

```
<ADBEntity EntityName='SA1244' Project='DEMO1' EntityType='A'
RevisionDate='06/06/2005' Class='3309'>
        <Description>SANITARY: Clinical handwash; medium hospital basin with automatic
        action mixer tap, soap\scrub solution dispenser . Fixing Ht. basin 860.
        </Description>
        <User1></User1>
        <User2></User2>
```

© Copyright Integra 2007

```
        <User3></User3>
        <User4></User4>
        <User5></User5>
        <Notes></Notes>
</ADBEntity>
```

Component

```
<ADBEntity EntityName='CHA017' Project='DEMO1' EntityType='C'
RevisionDate='06/06/2005' Class='3401' Group='3' Layer='A725_3' NSV='' PartNumber=''
Size='' TransferCost='0.00' Cost='84.00' Schedule='Yes' SupplyType='' InstallType=''
ComponentType='' ComponentType2='' Level='' DefaultModel=''>
        <Description>CHAIR, upright, upholstered, stacking</Description>
        <User1></User1>
        <User2></User2>
        <User3></User3>
        <User4></User4>
        <User5></User5>
        <Notes>Option: Low hazard fabrics (£78) vinyl (£88.20) to specify</Notes>
        <GenericSpec></GenericSpec>
        <ComponentServicesInt Value1='0' Value2='0' Value3='0' Value4='0' Value5='0'
        Value6='0' Value7='0' Value8='0' Value9='0' Value10='0'>
        </ComponentServicesInt>
        <ComponentServicesFloat Value1='0.000000' Value2='0.000000' Value3='0.000000'
        Value4='0.000000' Value5='0.000000' Value6='0.000000' Value7='0.000000'
        Value8='0.000000' Value9='0.000000' Value10='0.000000'>
        </ComponentServicesFloat>
        <ComponentServicesString>
        <ServicesString1></ServicesString1>
        <ServicesString2></ServicesString2>
        <ServicesString3></ServicesString3>
        <ServicesString4></ServicesString4>
        <ServicesString5></ServicesString5>
        <ServicesString6></ServicesString6>
        <ServicesString7></ServicesString7>
        <ServicesString8></ServicesString8>
        <ServicesString9></ServicesString9>
        <ServicesString10></ServicesString10>
        </ComponentServicesString>
        <ComponentServicesBool Value1='No' Value2='No' Value3='No' Value4='No'
        Value5='No' Value6='No' Value7='No' Value8='No' Value9='No' Value10='No'>
        </ComponentServicesBool>
</ADBEntity>
```

Activity

```
<ADBEntity EntityName='WAS009' Project='DEMO1' EntityType='V'
RevisionDate='30/12/1899' Class='3310'>
        <Description>Clinical hand washing.</Description>
        <User1></User1>
        <User2></User2>
        <User3></User3>
        <User4></User4>
        <User5></User5>
```

© Copyright Integra 2007

```
        <Notes></Notes>
</ADBEntity>
```

Interface Entity

```
<ADBEntity  Project=”DEMO1”
        TargetProject=””
        EntityType="R”
        EntityName=”B0303”
        TargetEntityName=””
        Suffix=””
        UpdateRevisionDate=”Yes”
        RevisionDate=”24-Aug-2007”
        Notes=””
        User1=””
        User2=””
        User3=””
        User4=””
        User5=””
        Class=””
        Description=”Single Bedroom”>
  <ChildEntityChangeList Project=”DEMO1" EntityType=”C”>
        <ChangeListItem EntityName=”OUT010”
                Quantity=”4” ID=”” InstanceAttribute=””>
        <ChangeListItem EntityName=”CHA017”
                Quantity=”2” ID=”” InstanceAttribute=””>
        ….
    </ChildEntityChangeList>
</ADBEntity >
```

Note:

1. Quantity is the actual quantity and not the change
2. ID is the instance ID and only used for Room Ordered Departments
3. InstanceAttribute is the Room Number or Sequence Number

Audit

```
<ICLEntityHistory>
        <ICLEntityHistoryEntry>
                <pk>    </pk>
                <entityid>       </entityid>
                <entitycode>   </entitycode>
                <entitydescription> </entitydescription>
                <version> </version>
                <who> </who>
                <what> </what>
                <why> </why>
                <when> </when>
                <requestedby> </requestedby>
                <authorisedby> </authorisedby>
```

© Copyright Integra 2007

```
                <consequence> </consequence>
                <consequenceid> </consequenceid>
                <consequencenotes> </consequencenotes>
        </ICLEntityHistory>
</ICLEntityHistoryEntry>
```

© Copyright Integra 2007

Brand

```
<ADBEntity EntityName='HP' Project='DEMO1' EntityType='B'
RevisionDate='27/08/2007'>
        <Description>Hewlett Packard</Description>
        <User1></User1>
        <User2></User2>
        <User3></User3>
        <User4></User4>
        <User5></User5>
        <Notes></Notes>
</ADBEntity>
```

Supplier

```
<ADBEntity EntityName='AGILENT' Project='DEMO1' EntityType='S'
RevisionDate='27/08/2007'>
        <Description>Agilent Technologies</Description>
        <User1></User1>
        <User2></User2>
        <User3></User3>
        <User4></User4>
        <User5></User5>
        <Notes></Notes>
</ADBEntity>
```

Model

```
<ADBEntity EntityName='HP4537A' Project='DEMO1' EntityType='M'
RevisionDate='27/08/2007' ModelCode='4537' ModelType='A' Brand='HP'
Manufacturer='AGILENT' Price='2000.00'>
        <Description>Defribrillator wirh text and ECG display</Description>
        <User1></User1>
        <User2></User2>
        <User3></User3>
        <User4></User4>
        <User5></User5>
        <Notes></Notes>
</ADBEntity>
```

© Copyright Integra 2007

Graphics – Sample Assembly Definition

<ICL_GFXContainer Name="EA1631" View="-P" Type="Unknown" X1="-1.00"
Y1="0.00" Z1="0.00" X2="0.00" Y2="0.00" Z2="0.00" Phi="0.00">
<ICL_GFXBlockList Count="3">
<ICL_GFXBlock Name="EA1631" Layer="" Type="Structure"
Description="ENGINEERING: Double 13 amp socket outlet & telephone outlet, Fixing
Hts. 400.
" View="-P" SheetSize="0" SheetScale="0.02" X1="-1.00" Y1="0.00" Z1="0.00" X2="0.00"
Y2="0.00" Z2="0.00" Phi="0.00">
<ICL_GFXInsertionList Count="2">
<ICL_GFXInsert Name="OUT010" Layer="A623_1" Type="Primitive" X="300.00"
Y="0.00" Z="400.00" Phi="0.00"/>
<ICL_GFXInsert Name="OUT215" Layer="A642_1" Type="Primitive" X="0.00"
Y="0.00" Z="400.00" Phi="0.00"/>
</ICL_GFXInsertionList>
</ICL_GFXBlock>
<ICL_GFXBlock Name="OUT010" Layer="A623_1" Type="Primitive"
Description="SOCKET outlet switched 13amp twin, wall mounted" View="-P" SheetSize="0"
SheetScale="0.02" Group="1" Level="" X1="-1.00" Y1="0.00" Z1="0.00" X2="0.00"
Y2="0.00" Z2="0.00" Phi="0.00">

<ICL_GFXVectorList Count="6">
<ICL_GFXLine LineStyle="CONTINUOUS" Layer="0" X1="0.00" Y1="-172.50"
Z1="0.00" X2="0.00" Y2="-230.00" Z2="0.00"/>
<ICL_GFXArc LineStyle="CONTINUOUS" X="0.00" Y="-100.00" Z="0.00"
Radius="42.50" Phi1="3.14" Phi2="0.00"/>
<ICL_GFXLine LineStyle="CONTINUOUS" Layer="0" X1="0.00" Y1="0.00" Z1="0.00"
X2="0.00" Y2="-100.00" Z2="0.00"/>
<ICL_GFXLine LineStyle="CONTINUOUS" Layer="0" X1="72.50" Y1="-100.00"
Z1="0.00" X2="-72.50" Y2="-100.00" Z2="0.00"/>
<ICL_GFXArc LineStyle="CONTINUOUS" X="0.00" Y="-100.00" Z="0.00"
Radius="72.50" Phi1="3.14" Phi2="0.00"/>
<ICL_GFXPolyline Flags="1" LineStyle="CONTINUOUS">
<ICL_GFXPolylineVertex X="-7.59" Y="-187.53" Z="0.00" T1="14.75" T2="14.75"
Bulge="1.00"/>
<ICL_GFXPolylineVertex X="7.66" Y="-187.53" Z="0.00" T1="14.75" T2="14.75"
Bulge="1.00"/>
</ICL_GFXPolyline>
</ICL_GFXVectorList>
</ICL_GFXBlock>
<ICL_GFXBlock Name="OUT215" Layer="A642_1" Type="Primitive"
Description="SOCKET outlet telephone, wall mounted" View="-P" SheetSize="0"
SheetScale="0.02" Group="1" Level="" X1="-1.00" Y1="0.00" Z1="0.00" X2="0.00"
Y2="0.00" Z2="0.00" Phi="0.00">
<ICL_GFXVectorList Count="9">
<ICL_GFXPolyline Flags="1" LineStyle="CONTINUOUS">
<ICL_GFXPolylineVertex X="-6.25" Y="-150.00" Z="0.00" T1="12.50" T2="12.50"
Bulge="1.00"/>
<ICL_GFXPolylineVertex X="6.25" Y="-150.00" Z="0.00" T1="12.50" T2="12.50"
Bulge="1.00"/>
</ICL_GFXPolyline>
<ICL_GFXLine LineStyle="CONTINUOUS" Layer="0" X1="-21.88" Y1="-131.48"
Z1="0.00" X2="-54.09" Y2="-131.48" Z2="0.00"/>

© Copyright Integra 2007

<ICL_GFXLine LineStyle="CONTINUOUS" Layer="0" X1="21.88" Y1="-99.45"
Z1="0.00" X2="21.88" Y2="-131.48" Z2="0.00"/>
<ICL_GFXLine LineStyle="CONTINUOUS" Layer="0" X1="-21.88" Y1="-99.45"
Z1="0.00" X2="21.88" Y2="-99.45" Z2="0.00"/>
<ICL_GFXLine LineStyle="CONTINUOUS" Layer="0" X1="-21.88" Y1="-131.48"
Z1="0.00" X2="-21.88" Y2="-99.45" Z2="0.00"/>
<ICL_GFXLine LineStyle="CONTINUOUS" Layer="0" X1="-0.97" Y1="-225.23"
Z1="0.00" X2="52.16" Y2="-131.48" Z2="0.00"/>
<ICL_GFXLine LineStyle="CONTINUOUS" Layer="0" X1="-54.09" Y1="-131.48"
Z1="0.00" X2="-0.97" Y2="-225.23" Z2="0.00"/>
<ICL_GFXLine LineStyle="CONTINUOUS" Layer="0" X1="52.16" Y1="-131.48"
Z1="0.00" X2="21.88" Y2="-131.48" Z2="0.00"/>
<ICL_GFXLine LineStyle="CONTINUOUS" Layer="0" X1="0.00" Y1="0.00" Z1="0.00"
X2="0.00" Y2="-100.00" Z2="0.00"/>
</ICL_GFXVectorList>
</ICL_GFXBlock>
</ICL_GFXBlockList>
</ICL_GFXContainer>

© Copyright Integra 2007

Appendix 2 – Entity Public Field Names used for Filters

Entity Public Field Names		
Entity	**PublicName**	**Data Type**
Activity	Class	VarChar
Activity	Code	Char
Activity	Description	String
Activity	RevisionDate	DateTime
Activity	UserData1	String
Activity	UserData2	String
Activity	UserData3	String
Activity	UserData4	String
Activity	UserData5	String
Assembly	Class	VarChar
Assembly	Code	Char
Assembly	Description	String
Assembly	Notes	String
Assembly	RevisionDate	DateTime
Assembly	Schedule	Boolean
Assembly	SheetScale	Double
Assembly	SheetSize	Integer
Assembly	SpaceBox_x1	Double
Assembly	SpaceBox_x2	Double
Assembly	SpaceBox_y1	Double
Assembly	SpaceBox_y2	Double
Assembly	SpaceBox_z1	Double
Assembly	SpaceBox_z2	Double
Assembly	UserData1	String
Assembly	UserData2	String
Assembly	UserData3	String
Assembly	UserData4	String
Assembly	UserData5	String
Audit	Approved By	String
Audit	Changed By	String
Audit	Code	VarChar
Audit	Consequence	String
Audit	Consequence Notes	String
Audit	Date	DateTime
Audit	Details of Change	String
Audit	Id	VarChar
Audit	Reason for Change	String
Audit	Requested By	String
Audit	Short Name	VarChar
Audit	Version	String

© Copyright Integra 2007

Entity Public Field Names		
Entity	**PublicName**	**Data Type**
Brand	Code	Char
Brand	Description	String
Brand	RevisionDate	DateTime
Brand	UserData1	String
Brand	UserData2	String
Brand	UserData3	String
Brand	UserData4	String
Brand	UserData5	String
Component	AC	Boolean
Component	AlternativeCode	Char
Component	Breadth	Double
Component	Cabling	String
Component	Class	Char
Component	ClassDescription	String
Component	ClassSchedule	Boolean
Component	Code	Char
Component	Cost	Double
Component	DC	Boolean
Component	DefaultModelID	Char
Component	DefaultModelName	Char
Component	Description	String
Component	Earth	String
Component	Gases	String
Component	GenericSpec	String
Component	Group	Char
Component	Height	Double
Component	InstallerType	Char
Component	Length	Double
Component	Level	Char
Component	MiscellaneousServices	String
Component	ModelClass	String
Component	Notes	String
Component	PartNumber	Char
Component	Phase	String
Component	Power	Double
Component	ProtectionType	String
Component	RevisionDate	DateTime
Component	Schedule	Boolean
Component	SheetScale	Double
Component	SheetSize	Integer
Component	Size	Char
Component	SpaceBox_x1	Double

© Copyright Integra 2007

Entity Public Field Names		
Entity	**PublicName**	**Data Type**
Component	SpaceBox_x2	Double
Component	SpaceBox_y1	Double
Component	SpaceBox_y2	Double
Component	SpaceBox_z1	Double
Component	SpaceBox_z2	Double
Component	SupplierType	Char
Component	TransferCost	Double
Component	UserData1	String
Component	UserData2	String
Component	UserData3	String
Component	UserData4	String
Component	UserData5	String
Component	UserType1	Char
Component	UserType2	Char
Component	VentilationServices	String
Component	Voltage	Double
Component	Volume	Double
Component	WaterServices	String
Component	Weight	Double
Department	Class	Char
Department	Code	Char
Department	Description	Char
Department	Notes	String
Department	Order	Char
Department	RevisionDate	DateTime
Department	UserData1	String
Department	UserData2	String
Department	UserData3	String
Department	UserData4	String
Department	UserData5	String
Level 1 Category	CategoryType	String
Level 2 Category	CategoryType	String
Level 3 Category	CategoryType	String
Model	Brand	String
Model	BrandCode	String
Model	Code	Char
Model	Cost	Double
Model	Description	String
Model	Name	String
Model	RevisionDate	DateTime
Model	Supplier	String
Model	SupplierCode	String

© Copyright Integra 2007

Entity Public Field Names		
Entity	**PublicName**	**Data Type**
Model	Type	String
Model	UserData1	String
Model	UserData2	String
Model	UserData3	String
Model	UserData4	String
Model	UserData5	String
Room	Area	Double
Room	Class	VarChar
Room	Code	VarChar
Room	Description	String
Room	Design_Ceilings	String
Room	Design_DoorSets	String
Room	Design_Flooring	String
Room	Design_Glazing	String
Room	Design_Hatches	String
Room	Design_Notes	String
Room	Design_Walls	String
Room	Design_Windows	String
Room	Fire_AutoDetectionRequired	String
Room	Fire_Enclosure	String
Room	Fire_GeneralNotes	String
Room	Height	Double
Room	HVAC_Arrestance	Double
Room	HVAC_DustSpotEfficiency	Double
Room	HVAC_FiltrationAndHumidityNotes	String
Room	HVAC_GeneralNotes	String
Room	HVAC_MechanicalVentilationExtract	Double
Room	HVAC_MechanicalVentilationNotes	String
Room	HVAC_MechanicalVentilationSupply	Double
Room	HVAC_MechanicalVentilationUnits	Integer
Room	HVAC_RelativeHumidity	Double
Room	HVAC_RelativePressure	String
Room	HVAC_SummerTemperature	Double
Room	HVAC_TemperatureNotes	String
Room	HVAC_WinterTemperature	Double
Room	Lighting_ColourRenderingNotes	String
Room	Lighting_ColourRenderingRequired	Boolean
Room	Lighting_GeneralNotes	String
Room	Lighting_LocalIllumination	Double
Room	Lighting_LocalIlluminationNotes	String
Room	Lighting_ServiceIllumination	Double
Room	Lighting_ServiceIllumination_Night	Double

© Copyright Integra 2007

Entity Public Field Names		
Entity	**PublicName**	**Data Type**
Room	Lighting_ServiceIlluminationNotes	String
Room	Lighting_ServiceIlluminationNotes_Night	String
Room	Lighting_StandbyLightingGrade	Char
Room	Lighting_StandbyLightingNotes	String
Room	Noise_AcceptableNoiseLevel	Double
Room	Noise_GeneraNotes	String
Room	Noise_IntolerableNoiseQuality	String
Room	Noise_IntrusiveNoiseLevel	Double
Room	Noise_MechanicalServices	Double
Room	Noise_NoiseLevelNotes	String
Room	Noise_PrivacyFactor	Double
Room	Noise_SpeechPrivacyRequired	Boolean
Room	Notes	String
Room	Personnel	String
Room	PlanningRelationships	String
Room	RevisionDate	DateTime
Room	Safety_GeneralNotes	String
Room	Safety_HotSurfaceTemperature	Double
Room	Safety_HotWaterTemperature	Double
Room	Safety_TemperatureNotes	Double
Room	SheetScale	Double
Room	SheetSize	Integer
Room	SpaceBox_x1	Double
Room	SpaceBox_x2	Double
Room	SpaceBox_y1	Double
Room	SpaceBox_y2	Double
Room	SpaceBox_z1	Double
Room	SpaceBox_z2	Double
Room	UserData1	String
Room	UserData2	String
Room	UserData3	String
Room	UserData4	String
Room	UserData5	String
Supplier	Code	Char
Supplier	Description	String
Supplier	RevisionDate	DateTime
Supplier	UserData1	String
Supplier	UserData2	String
Supplier	UserData3	String
Supplier	UserData4	String
Supplier	UserData5	String

© Copyright Integra 2007