WASP3D

Add-In Creation

Add-in

An Add-in is a small component that enhances capability of application. It requires minimal memory resource and is used to perform specific task. Add-in is programmed for a specific larger application, and it acts as a child program after its installation.

For example, Excel, SQL (Store Procedure), and XML are the add-ins for Drone Designer.

Entry in hostedaddin.xml:

```
<addin providerid="B4AA4316-C886-4734-8245-29EB2DF86E78" providername="ClockProvider"
dll="Beesys.Wasp.AddIn.Clock.dll" alias="ClockAddIn" addin="Beesys.Wasp.AddIn.ClockAddInProvider"
class="Beesys.Wasp.AddIn.ClockAddIn" />
```

User can create the Add-in events and actions at container level. When event is raised then user can perform some action and also user can create the Add-in action that will execute on Designer event / Add-in events.

To Create an Add-in event user needs to implement IWEventHelper interface and provide the instance of that event class through IWContainer.

Some Add-in events are like: OnVariableChange, DataReady, PageChange etc.

To Create an Add-in action user needs to implement IWActionHelper interface and provide the instance of that action class through IWContainer.

Some Add-in actions are like: String Format, Blur etc.

Addin Creation

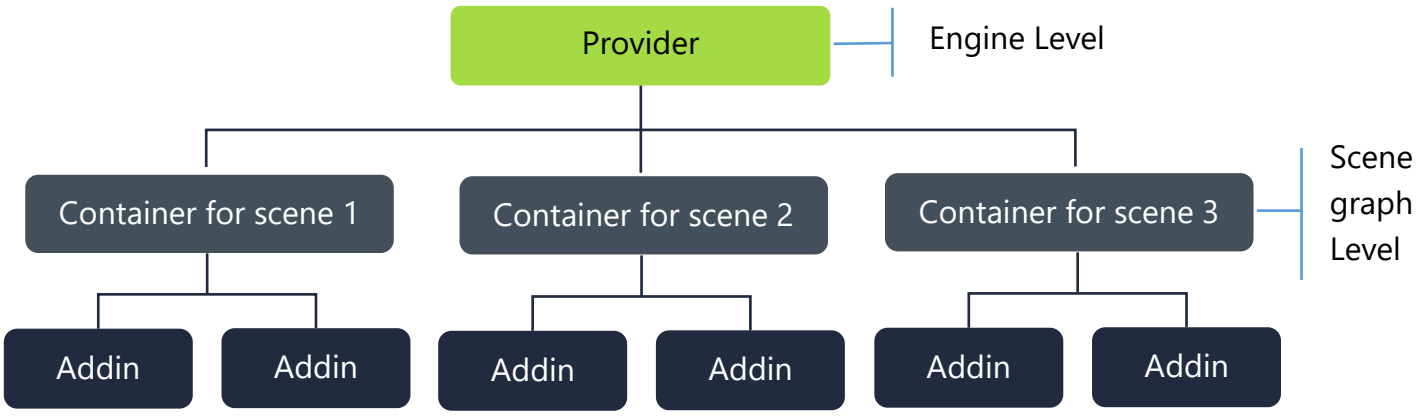
To create an Addin, 3 base class to be create: WProvider, WContainer, WAddIn or WDataSet or WDataTable

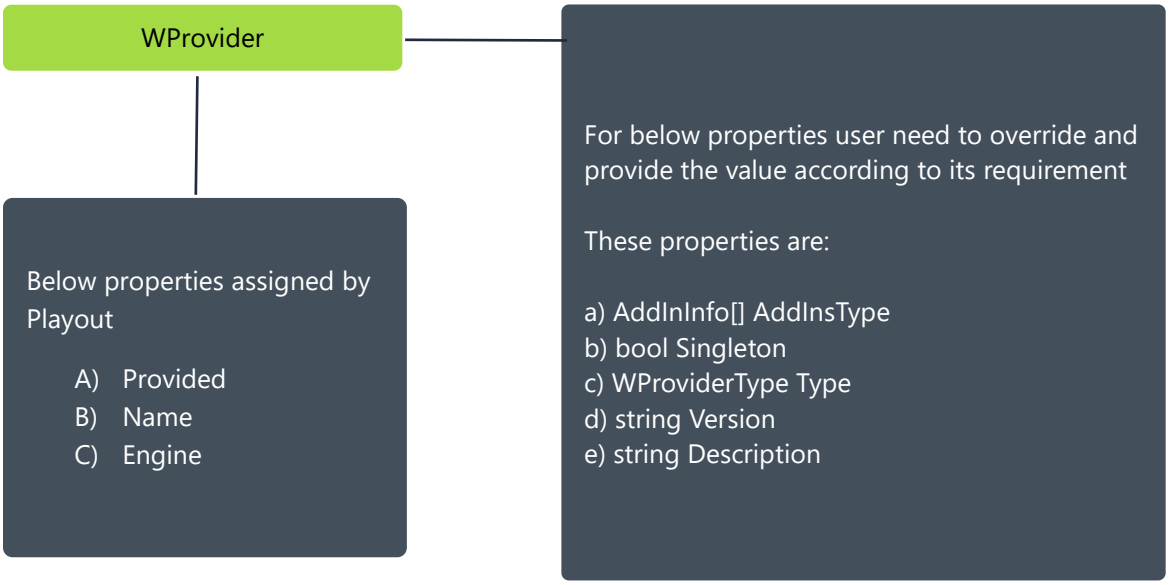
Provider

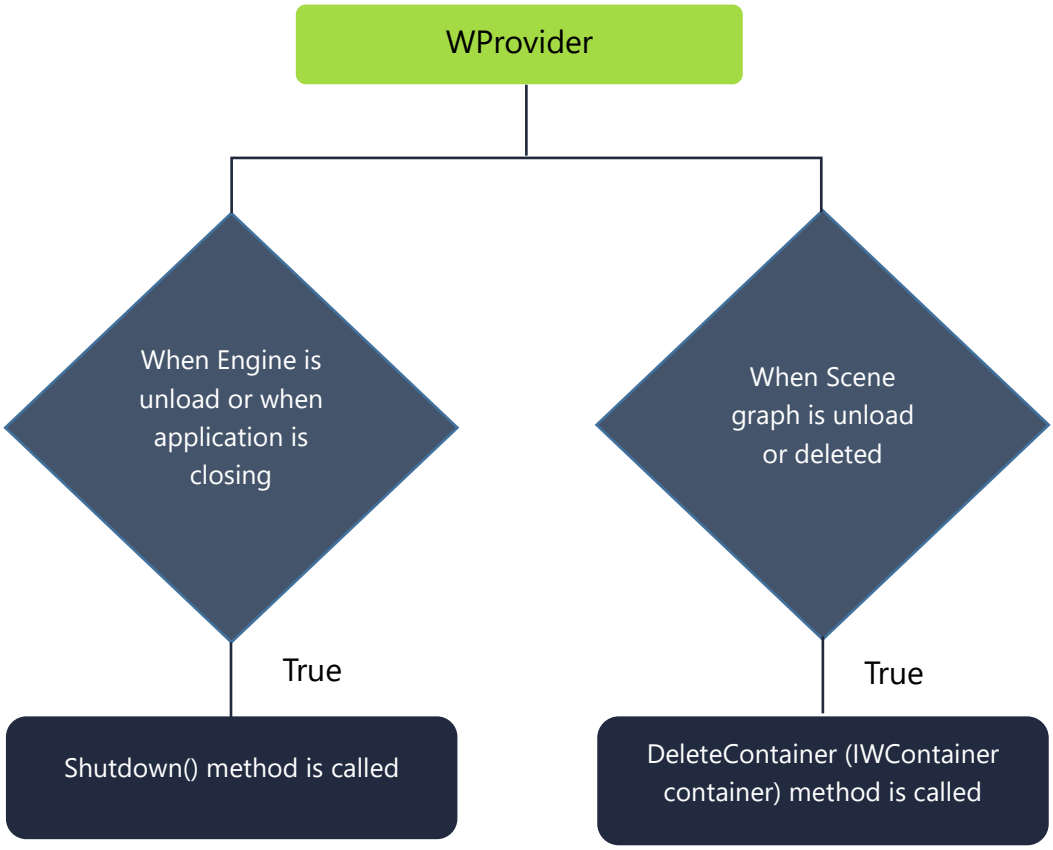
<ProviderClass> implements IWProvider/WProvider

- Instance of this classes created by Engine on Application Start.
- It has the instance of IWContainer, which creates in GetContainer() method provided by IWProvider
- StartUp () and ShutDown() will call on the creation and on when application closes respectively.
- On StartUp, user can create the instances of global resources and shared the same in all Add-in instances.
- On ShutDown, provider release the occupied resources and call the ShutDown() of the container classes.
- DeleteContainer(IWContainer container) is used to delete or shutdown the particular container from Provider list.(this method is called when scene is unload).
- Set the Properties of IWProvider: Name,Description,HelpString,Singleton,Type etc.

NOTE: *If user will bind m_evtEWEnginRenderInfo event of Engine then it is preferable to write the code asynchronously inside that method (Binded Method of m_evtEWEnginRenderInfo event) to ignore any delay in graphics.*



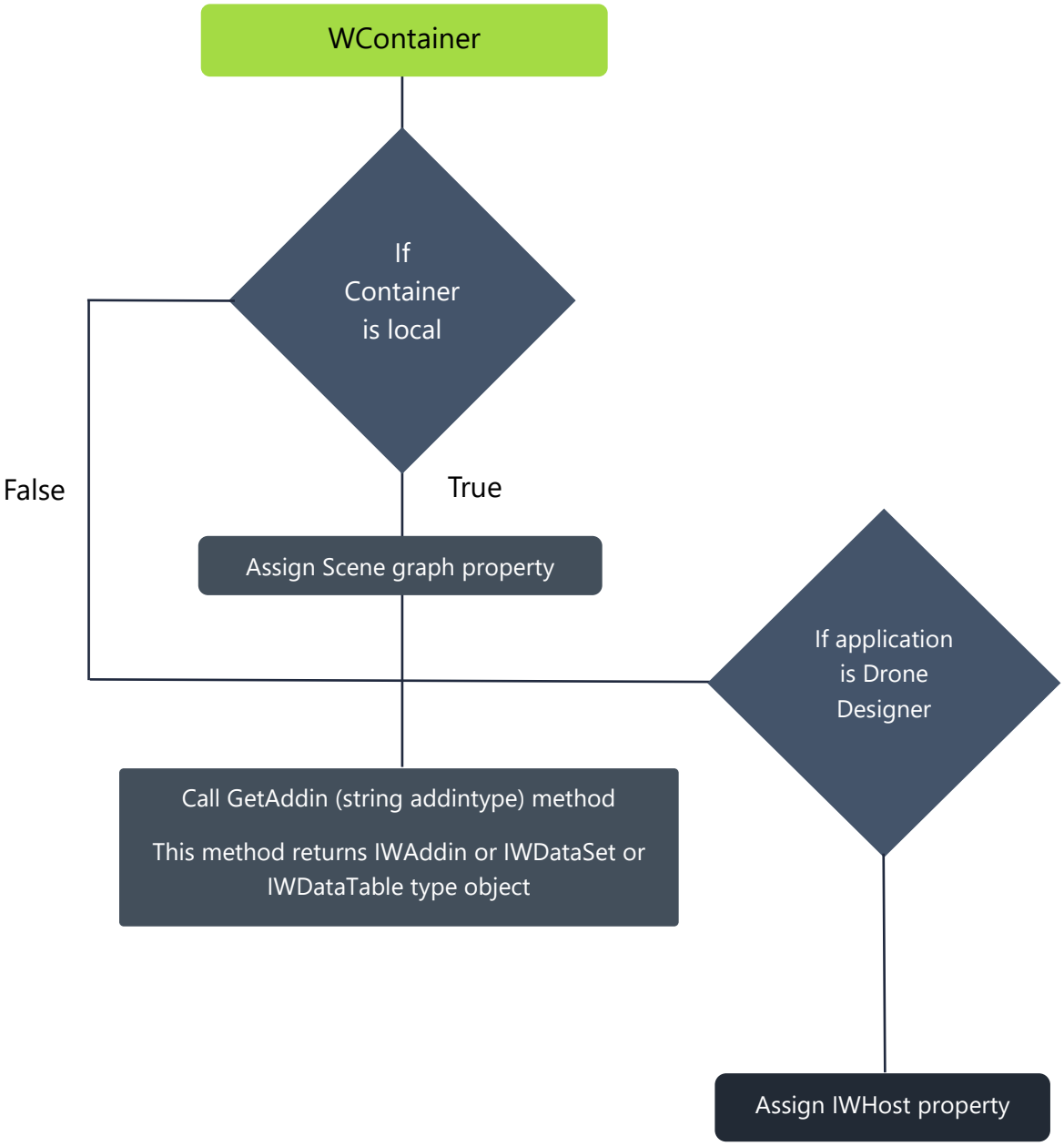

WProvider
Below properties assigned by Playout
A) Provided
B) Name
C) Engine
For below properties user need to override and provide the value according to its requirement
These properties are:
a) AddInInfo[] AddInsType
b) bool Singleton
c) WProviderType Type
d) string Version
e) string Description

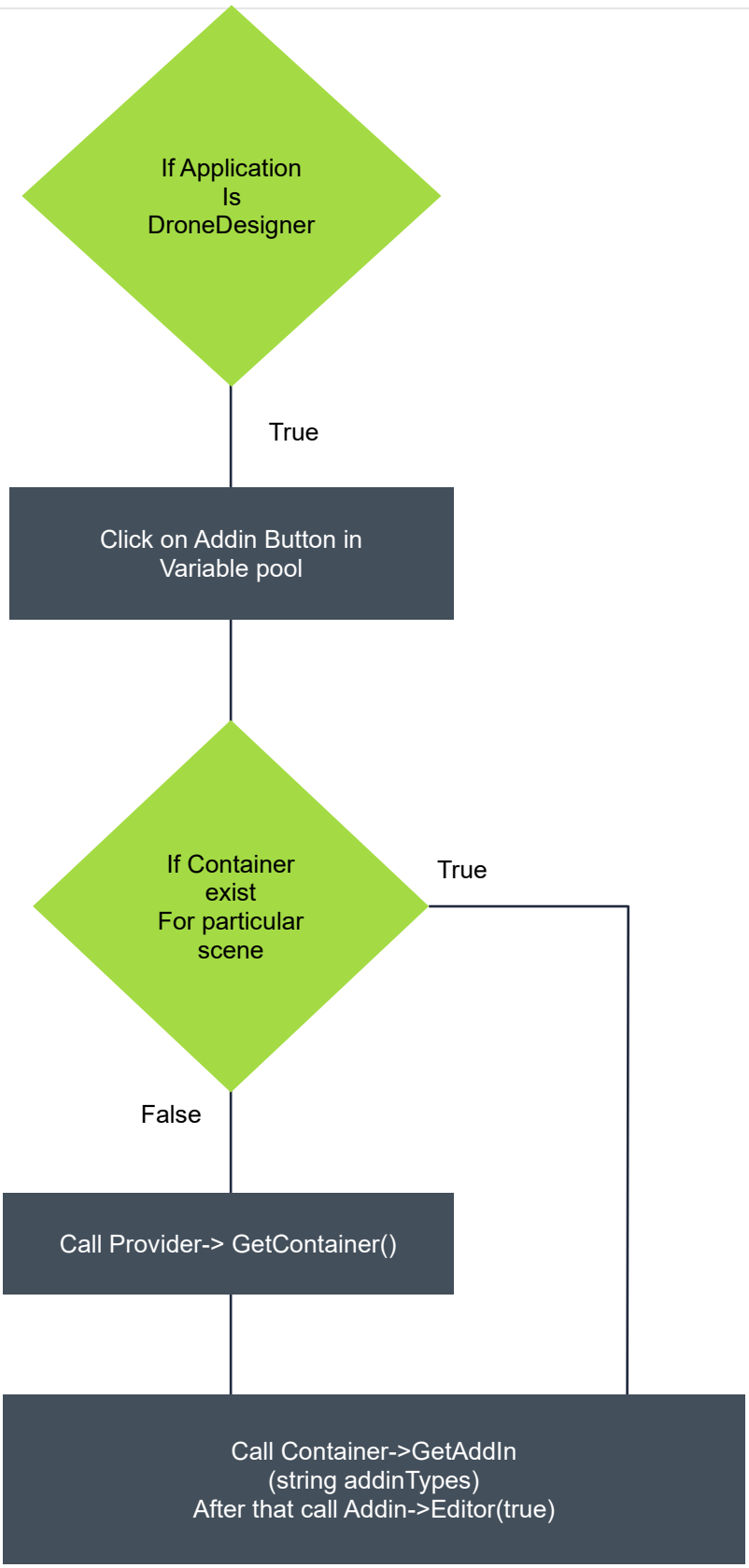

WProvider
When Engine is unload or when application is closing
When Scene graph is unload or deleted
True
True
Shutdown() method is called
DeleteContainer (IWContainer container) method is called

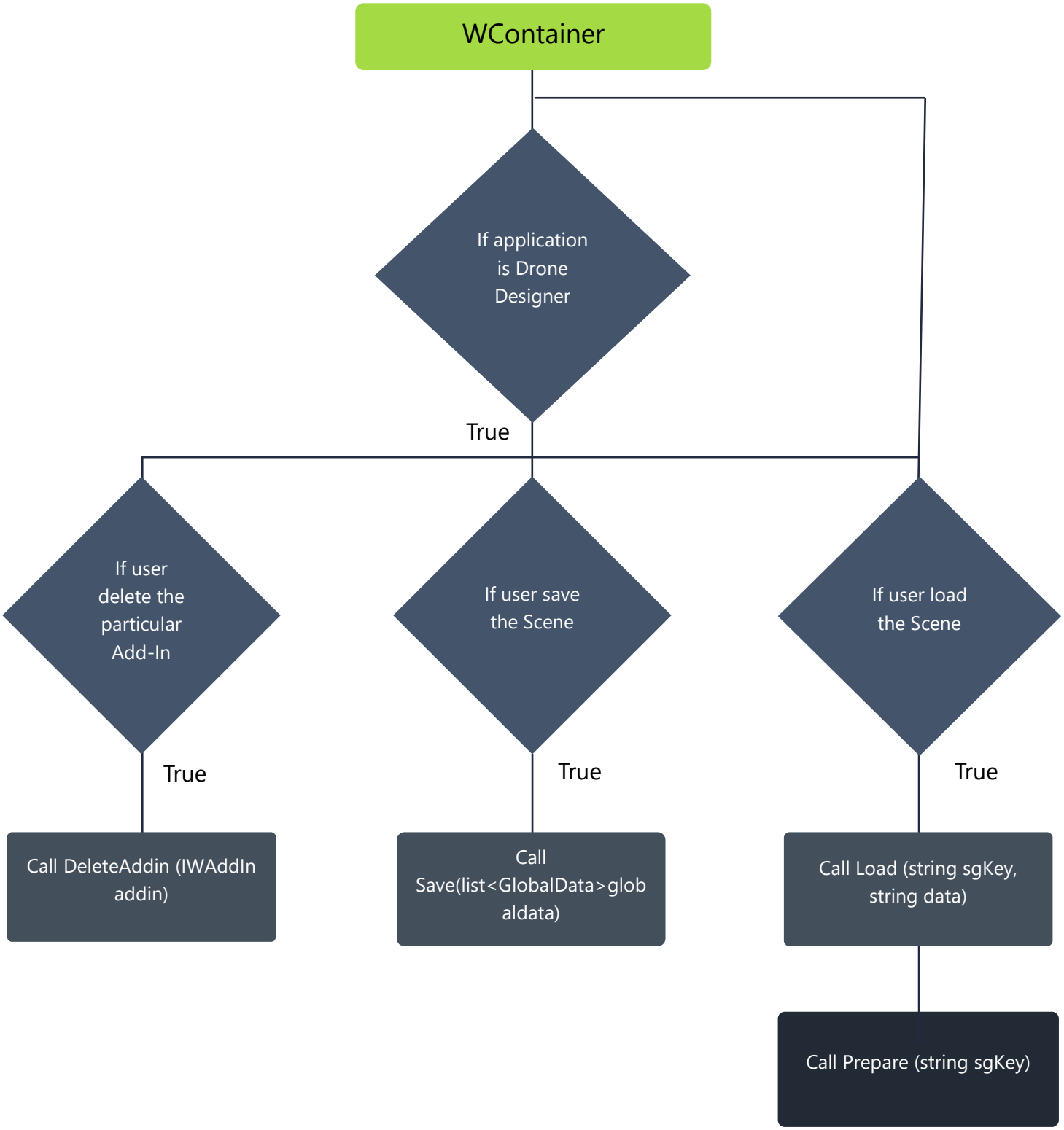
Container

<ContainerClass> implements IWContainer

- Instance of this class created by Provider Class.
- This class contains the instance of all IWAddIn classes that are created in GetAddin() method and maintains the list of addins.
- public override IWHost Host property is assigned only in DroneDesigner.which provide some method to get the data related to host
- On calling GetActionDescriptor(), it returns the dictionary containing action name and IWActionHelper instance(if any).
- On calling GetEventDescriptors(), it returns the dictionary containing event name and IWEventHelper instance(if any).
- IWAddIn GetAddin(string subtype) create the instance of IWAddIn, and return it.
- GetWAddins() return the array of instance of IWAddIn classes, created by container.
- Load (string data) loads the data to its IWAddIn classes, it contains. This method is called when scene is load
- string Save (List<globaldata>) will collect the data/config from all the Addin,Event,Action and these string formatted data save with scene.
- Prepare (string sgKey) will call when all the Add-In is prepared, and user will load or Initialize addin dataset here.
- ShutDown() will release all the resources and unbind the events of container and of all IWAddIn Classes(if any).


WContainer
If Container is local
False
True
Assign Scene graph property
If application is Drone Designer
Call GetAddin (string addintype) method
This method returns IWAddin or IWDataSet or IWDataTable type object
Assign IWHost property


If Application
Is
DroneDesigner
True
Click on Addin Button in
Variable pool
If Container
exist
For particular
scene
True
False
Call Provider-> GetContainer()
Call Container->GetAddIn
(string addinTypes)
After that call Addin->Editor(true)

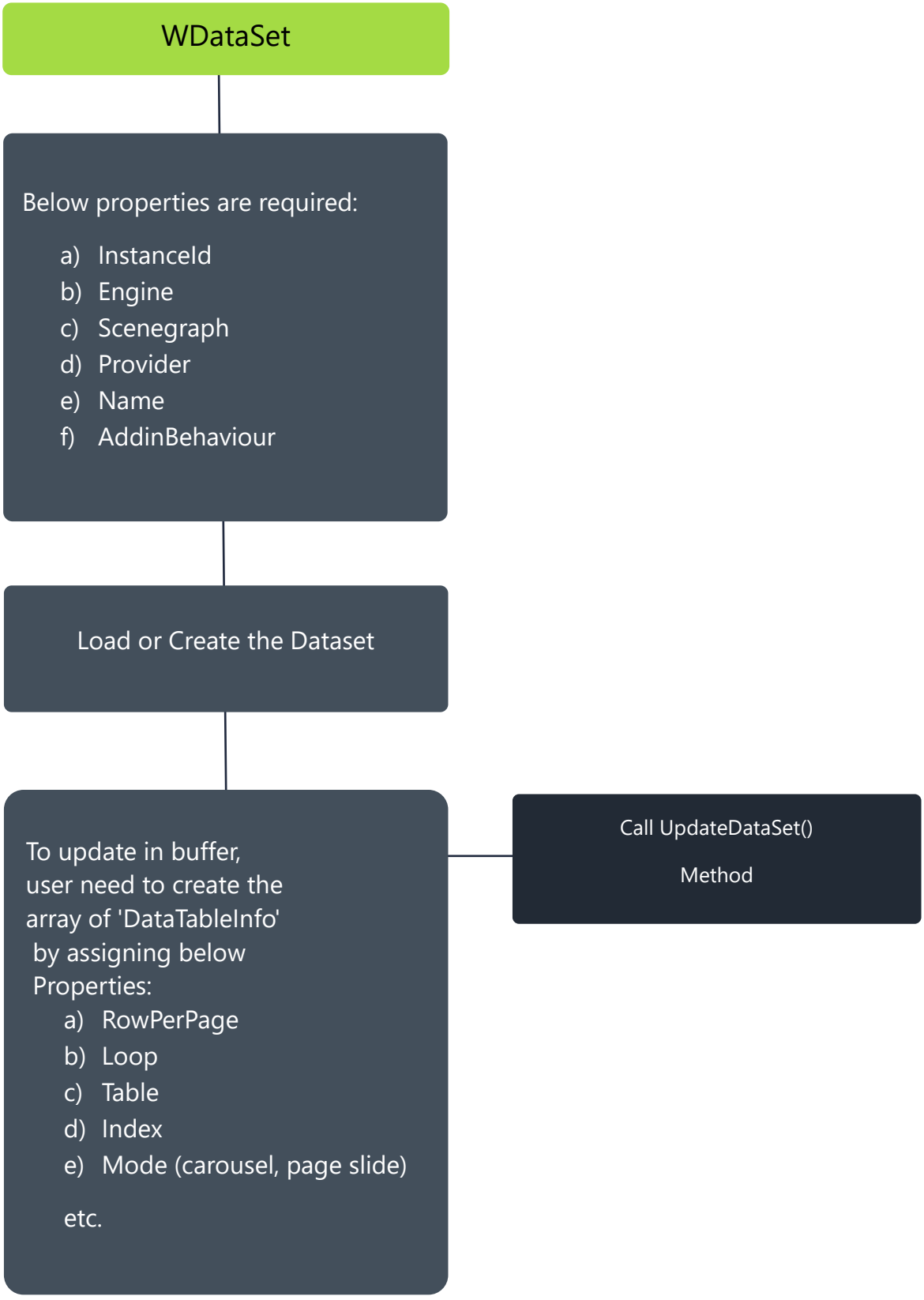

WContainer
If application is Drone Designer
True
If user delete the particular Add-In
If user save the Scene
If user load the Scene
True
True
True
Call DeleteAddin (IWAddIn addin)
Call Save(list<GlobalData>globaldata)
Call Load (string sgKey, string data)
Call Prepare (string sgKey)

Add-ins:

<AddinClass> implements <BaseClass>(WDataSet/WDataTable/WVariables/WAddIn)

- This class handles the dataset,datatable ,variables etc.
- Implementing the base class required public and override methods, which the client can use according to the addin requirement.
- AddinClass has to override the methods (Init(), Editor(),ShutDown() etc.) required for addin.

1. **Create Addin of Variable Type (<AddinClass>:WVariables)**
   - To Create addin of variable type, AddinClass implements 'Wvariables'.
   - Base class provide public method to perform actions on Variables. CreateVariable(),UpdateVariable(),DeleteVariable() etc.)
   - AddinClass has to override the methods to get variable events. (OnVariableAdd(),OnVariableDelete(),OnVariableUpdate() etc.)
2. **Create Addin of DataTable Type (<AddinClass>:WDataTable)**
   - To Create addin of DataTable type, AddinClass implements 'WdataTable'.
   - AddinClass also has the functionality of all methods provided in WVariables, so this addin also creates variables and perform action on it.
3. **Create Addin of DataSet Type (<AddinClass>:WDataSet)**
   - To Create addin of DataSet type, AddinClass implements 'WDataSet'.
   - AddinClass also has the functionality of all methods provided in WVariables, so this AddinClass also creates variables and perform action on it.
4. **Create Simple Addin (<AddinClass>:WAddIn/IWAddIn)**
   - To create simple addin, AddInClass implements “WAddIn”
   - AddinClass has the functionality of IWAddIn and does not provide any variable or dataset functionality.


WDataSet
Below properties are required:
a) InstanceId
b) Engine
c) Scenegraph
d) Provider
e) Name
f) AddinBehaviour
Load or Create the Dataset
To update in buffer, user need to create the array of 'DataTableInfo' by assigning below Properties:
a) RowPerPage
b) Loop
c) Table
d) Index
e) Mode (carousel, page slide)
etc.
Call UpdateDataSet()
Method

EVENTS and ACTIONS in AddIns

- AddIns has their own Events and Actions.
- Events are to be raised
- Events and Actions has its own Control which are displayed in Designer.
- Events can be selected from the context menu displayed on AddIn node in Designer.
- Actions are displayed in Action Popup, open on Designer Events/AddIns Events like OnTrigger, OnVariableChange etc.

1. **Create Addin Events (<EventsClass>: IWEventHelper)**
   - To create events to be displayed on Addins created under a container.
   - Create Events instance in Container class, and its UI (if any) instance at Provider Class, which pass UI instance to container, to then Event Class.

2. **Create Addin Actions (<ActionClass>: IWActionHelper)**
   - To create actions in AddIns, action class implements IWActionHelper.
   - AddIn Actions execute an action on AddIn Data, and its UI (if any) be created at Provider Class, which pass UI instance to container, to then Action Class.

***Note**: Implementation of Base Class handles the Buffer Management, otherwise user must maintain buffer data of its own.*

Interfaces and Its related Base Class

1) IWProvider

Methods	Properties	Description
	string ProviderId { get; set; }	Gets or sets the id of the IWProvider • Set by the Engine when instance of IWProvider created.
	string Name { get; set; }	Gets or Sets the Name of the Provider, used to display in Designer to identify the addin.
	WProviderType Type	Gets the type of the provider (e.g. Action,DataSet,Event etc). This helps to identify the types of addins the IWProvider contains.
		Enum WproviderType: • Action =1, DataSet=2, DataTable=4, Event=8, Variables=16, Utility=32
	AddinsInfo[] AddinsType	Gets the array of AddInsInfo. Helps to know the addin type the Provider contains.
		Struct AddinsInfo: • bool Button: determine whether to display addin button in application or not. • String AddInType: describe the addintype,helps to create addin data according to it. • String addinDescription : helps to describe the description of addin provider creates.
	bool Singleton {get;}	Gets a value to determine whether to create multiple instances of IWAddIn or not.
	string Description {get;}	Gets the description about the IWProvider, which details the information of IWProvider.
	string Version { get; }	Get a value to set the version of the IWProvider.
void StartUp();		Calls after the creation of IWProvider. • On StartUp, user can create the instances of global resources and shared the same in all Add-in instances.
IWContainer GetContainer();		Create an instance of IWContainer and return it. • Provide the new instance of IWContainer and maintain that instance in the List<IWContainer>. • Set the Engine instance in the IWContainer.

void ShutDown();		Calls when the application close. • user has to release all the resources contain in the IWProvider. Use the IWContainer list and call the ShutDown of each IWContainer in the list.
CPlayoutEngine Engine		Set the Engine property of IWProvider by Playout.
void DeleteContainer(IWContainer container);		Calls when the scene will unload or deleted. • user has to remove the instance of IWContainer from the List< IWContainer>. • Call Shutdown() method of IWContainer instance.

1.1) WProvider Base class: Implement IWProvider

Methods	Properties	Description
	public string Name { get; set; }	Gets or sets the name of the provider
	public string ProviderId { get; set; }	get or set the provider id
	public abstract AddinsInfo[] AddinsType { get; }	Gets the array of addinsinfo created in provider. This tells the information of addins created.

		public struct AddinsInfo
		{
		/// <summary>
		/// Gets or sets to determine whether to create the addin button on designer or not
		/// </summary>
		public bool Button {get; set;}
		
		/// <summary>
		/// Gets or sets the type of the addiin
		/// </summary>
		public string AddinsType {get; set;}
		
		/// <summary>
		/// Gets/Set the description for AddIns
		/// </summary>
		string Description {get; set;}
		
		}

	public abstract WProviderType Type {get;}	Gets the type of Provider. This helps to know which type of addin, the current provider contains.

		public enum WProviderType
		{
		Action =1,
		DataSet=2,
		DataTable=4,
		Event=8,
		Variables=16,
		Utility=32
		}

	public virtual string Description { get; }	Gets the description of the Provider
	public virtual string Version { get; }	Gets the Provider Version
	public virtual bool Singleton { get; }	Gets a value to determine whether Addin has Single object or not
	public virtual CPlayOutEngine Engine {get; set; }	Get / Set the Engine to the Provider Provider will set the Engine to the Container created under it accordingly.
protected virtual void Initialize (string description, string version, bool singleton)		Initialize the provider with the given parameters data.
public abstract void StartUp()		Call on the creation of IWProvider
public abstract IWContainer GetContainer();		Get the IWContainer
public abstract void DeleteContainer		Delete the IWContainer from the current provider.
public abstract void ShutDown()		Call when the application close

2) IWContainer

Methods	Properties	Description
	IWHost Host {get; set; }	Get and Set the IWHost to the container. • Through this user get the IWService instance, which get the designer services like list of variables/nodes/dataset etc. • User has to set the IWService instance to the IWAddIns/IWEventHelper/IWActionHelper creates under the current IWContainer.
	Scene SceneGraph {get; set;}	Calls after the creation of IWContainer instance. This property will set through playout engine. • Container will set the SceneGraph property of all addins that are created under it.
void StartUp();		Calls after the creation of IWContainer instance. • In this user can create a list containing IWAddin instance..
List<EventsInfo> GetEventDescriptors();		Gets the list of EventsInfo created for the addin • EventsInfo contains, array of addintype under which this event should be display and IWEventHelper instance.
		Struct EventsInfo : public string[] AddInType { get; set; } public IWEventHelper Event { get; set; }
List<ActionsInfo> GetActionDescriptor();		Gets the list of ActionsInfo created for the addin • ActionsInfo contains, ActionCategory which specifies the type of action used in designer and IWActionHelper instance.
		Struct ActionsInfo : public string Category { get; set; } public IWActionHelper Action { get; set;
void Load (string data);		Load the data to the addins/actions/events. Calls while loading the scene for the current IWContainer. • This data contains all the information that IWContainer gave during save time. IWContainer has to set the data to the addins/actions/events accordingly.
string Save(List<GlobalData> globalData);		Return the data containing all information, which has to be save for the current IWContainer like Add-In config, Event data, Action data that are used in the current scene (cumulate all the data and return that string). Calls when user save the scene. • In case of global scope: List<GlobalData> param contains the information of Global variable which are used for current scene. • In case of local scope : List< GlobalData> is always null.

		Struct GlobalData : public string AddInId { get; set; } public string[] VariableName { get; set;
IWAddIn GetAddin(string addinType);		Return the instance of IWAddIn on the basis of given addin type. • User has to set the SceneGraph, Engine and IWService to this new IWAddIn instance. • Add this IWAddIn instance to the addin list created at StartUp().
void DeleteAddin(IWAddIn addin);		Delete the IWAddIn from the IWContainer • Call the ShutDown of the IWAddIn to be deleted. • Remove the IWAddin instance from list of addin.
void ShutDown();		Calls when the scene will unload or close. Release all the resource that are occupied by IWContainer. • User call the ShutDown of all the IWAddIn contains in current IWContainer.

2.1 : **IWContainerEx**

Methods	Properties	Description
Void Prepare(string sgKey);		This method is Called after Container ->Load method. This method is used to update the addin data

2.2) WContainer Base class (Implement IWContainer and IWContainerEx)

Methods	Properties	Description
	public virtual IWHost Host { get; set; }	Get or set the IWHost,which provide Services(nodes,variables,addins etc) from Designer.
	public virtual CPlayOutEngine Engine { get; set; }	Get or set the Engine object to the container.
	public abstract Scene SceneGraph { set; get; }	Get or Set the SceneGraph object to the container.
public abstract void StartUp();		This method is called after the creation of container object.
public abstract IWAddIn GetAddin		Returns the object of the IWAddin based on the addintype given.

public abstract void DeleteAddin		Delete the IWAddIn from the current container.
public abstract void Load		Load the data (events/actions/variables etc)
public abstract string Save		Returns the data(events,actions,variables etc) to save [for global addin list is not null and for local addin list is null]
public abstract void ShutDown();		Free the resources handled by container
public virtual List<ActionsInfo> GetActionDescriptor()		Returns the list of ActionsInfo created in container
public virtual List<EventsInfo> GetEventDescriptors()		Returns the list of EventsInfo created in container
	public IWProvider Provider { get; set; }	Set the object of provider
Void Prepare(string sgKey);		This method is called after Container ->Load method. This method is used to update the addin data

3. IWAddIn

Methods	Properties	Description
	IWProvider Provider {get; set;}	Get and Set the IWProvider under which this IWAddIn create. • This helps to get the IWProvider information to the Addin.
	string InstanceId {get; set;}	Get and Set the id of the AddIn instance. This is a unique GUID id, which sets through designer while creating the IWAddIn. • This ID helps to create the buffer, as buffer name contains the addinid to maintain the uniqueness. For Global Singleton AddIn, user has to set InstanceId of its own.(GUID Id)
	string Name {get; set;}	Get and Set the name of the current IWAddIn instance.Name will be assigned by designer while creating the IWAddIn on button click from variable pool.
	string AddinType {get;}	Get the type of the IWAddIn created. • This helps to get the type of addin (eg Integer,Float,Excel etc) • This also helps to mention in AddInsInfo in IWProvider, and to display the event according to the addin type.
	string Description {get; }	Get the description of the IWAddIn to know the details of this IWAddIn.
	string Config {get; set; }	Get and set the config data for the current IWAddIn instance. This data contains all the information required for this IWAddIn instance.
void StartUp();		Calls after the creation of IWAddIn instance. • Set the IWAddin properties with its default value (if any) • Create default variables, if implemented with IWInterface.
DialogResult Editor (bool showDialog);		Return the DialogResult to execute the addins operation on the basis of return result. • This mainly used to show a dialog box to set the Config of the addin instance data through GUI., and return the result as per the selection. • User can show the dialog only if bool showDialog is true, else don't show the dialog popup (if any). • Initialize the IWAddin with the data contained in the Config. • User has to create the buffer according to the addin data. • Call the Scene.CreateDataSource()/ CPlayoutEngine.CreateDataSource() to create buffer according to scope(global/local) of addin.
	AddInOptionsBehaviour AddInBehaviour {get;}	Get the behaviour of IWAddIn to perform like AllowDelete,AllowRename etc.

		• Based on this behaviour, operations like delete, rename etc are performed on the IWAddIn • By default it takes as None.
		enum AddInOptionsBehaviour : None = 0, AllowDelete = 1, AllowRename = 2, AllowScopeConversion = 4, AllowUpdate = 8, AllowVisible = 16, AllowAll = AllowDelete | AllowRename | AllowScopeConversion | AllowUpdate | AllowVisible
	int RefCount {get; }	Get the reference count of addin, used for global addin. • Call the IWAddIn shutdown (), if count is 0 • It is used to keep the used IWAddIn instance.
void AddRef();		Increment the RefCount of IWAddIn. • Used for global IWAddIn. • Call after the instance of IWAddIn created
void DeleteRef();		Decrement the RefCount of IWAddIn. • Used for global IWAddIn. • Call on scene unload
void ShutDown();		Calls when the addin delete. • In this, user has to release occupied resource(if any) and unbind the events.

3.1 : **IWAddInEx**

Methods	Properties	Description
	bool HideDefaultEvent { get; }	Return true/false to show the Default event like Variable change, Page Change.

3.2) WAddin (Implement IWAddin ,IWAddinEx)

Methods	Properties	Description
	public string InstanceId { get; set; }	Get and Set the id of the AddIn instance. This is a unique GUID id, which sets through designer while creating the IWAddIn. • This ID helps to create the buffer, as buffer name contains the addinid to maintain the uniqueness. For Global Singleton AddIn, user has to set InstanceId of its own.(GUID

	public string Name { get; set; }	Get and Set the Addin Name
	public Scene SceneGraph { get; set; }	Get and Set scenegraph object
	public CPlayOutEngine Engine { get; set; }	Get and Set engine object
	public IWProvider Provider { get; set; }	Get and Set the instance of Provider
	public virtual string AddinType { get; }	Get the Addin Type
	public virtual string Description { get; }	Get the Description about Addin
	public abstract string Config { get; set; }	Get and Set the Config Value of Addin
public abstract void StartUp();		Call on the creation of Addin object
public abstract DialogResult Editor		This method is used to show a dialog box to set an config required by Add-in.
public abstract void ShutDown();		Calls when the addin delete. In this, user has to release occupied resource(if any) and unbind the events.
protected virtual void Initialize		Initialize the Addin
	public virtual AddInOptionsBehaviour AddInBehaviour { get; }	Get the Addin Options Behaviour that decides the addin access to user.
public void AddRef()		Increment the reference count of addin

public void DeleteRef()		Decrement the reference count of addin
	public int RefCount { get; }	Get the reference count of addin
	bool HideDefaultEvent { get; }	Return true/false to show the Default event like Variable change, Page Change. etc

4) IWVariables

Methods	Properties	Description
	VAR_TYPE Type { get; set; }	Get and Set the BeeSys.Wasp3D. Hosting, VAR_TYPE. • This helps to create the addin variable on the basis of this type.

		enum VAR_TYPE:
		{
		VAR_INT = 0,
		VAR_FLOAT = 1,
		VAR_UINT = 2,
		VAR_BSTR = 3,
		VAR_VECTOR2 = 4,
		VAR_VECTOR3 = 5,
		VAR_VECTOR4 = 6,
		VAR_ALL = 7,
		VAR_COLOR = 8,
		VAR_BOOL = 9,
		VAR_FILE = 10,
		}

bool CreateVariable(string name, string data, VAR_TYPE type, VariableOptionsBehaviour optionBehaviour = VariableOptionsBehaviour.AllowAll, List<SubParams> SubParams =null);		Create the variable for this addin with the given name,data, type and VariableBehaviour as optional parameter, • Return true or false on the basis of variable created or not • Users has to write the data in buffer. • User has to raise the SceneGraph/Engine RaiseVariableChange event to reflects the effect in designer. • VariableOptionsBehaviour tells the behaviour of variable I.e the operations to be performed on this variable, on the basis of this behaviour • By default, variable behaviour be taken as All. • List<SubParams> SubParams tells the name of SubChild for the given variable (if any)

		enum VariableOptionsBehaviour
		{
		None = 0,
		AllowUpdate = 1,
		AllowDelete = 2,
		AllowRename = 4,
		AllowScopeConversion = 8,
		AllowVisible = 16,
		AllowAll = AllowUpdate | AllowDelete | AllowRename | AllowScopeConversion | AllowVisible
		}

		public struct SubParams
		{
		public string Name;
		public string DisplayName;
		}

bool DeleteVariable(string name);		Delete the variable from the IWVariables. • Return true or false on the basis of variable delete or not • User has to update the buffer data with the empty value of this variable. • User has to raise the SceneGraph/Engine RaiseVariableChange event to reflects the effect in designer.
bool RenameVariable(string oldName, string name);		Rename the old variable name with the given name. • Return true or false on the basis of variable rename or not • User has to raise the SceneGraph/Engine RaiseVariableChange event to reflects the effect in designer.
VarInfo GetVariable(string name);		Gets the VarInfo, contains the information of variable on the basis of provided name (that are created in this IWVariables).

		struct VarInfo
		{
		public string Name;
		public string Value;
		public VAR_TYPE Type;
		public VariableOptionsBehaviour OptionsBehaviour;
		}

VarInfo[] GetVariable();		Gets the array of VarInfo, contains the information of all variables created in this IWVariables. • VarInfo contains the variable information like: Variable Name,Variable Value, type of variable and variablebehaviour.
bool UpdateVariable(string name, string value);		Update the variable values on the basis on provided variable name and return true/false to determine whether the value is updated or not.
bool UpdateVariable(string name, string value,bool updateReadBuffer);		Update the variable values on the basis on provided variable name and return true/false to determine whether the value is updated or not. • To update the buffer immediately [on the current frame] pass updateReadBuffer as true otherwise false.

4.1 : **IWVariableEx**

Methods	Properties	Description
bool UpdateVariableBehaviour(string name, VariableOptionsBehaviour optionBehaviour);		this method updates the behaviour of particular variable /// <param name="name">variable name</param> /// <param name="optionBehaviour">behaviour of variable</param>

4.2 : **IWVariableEx2**

Methods	Properties	Description
bool UpdateVariables (Dictionary<string, string> variableNaneValeCollection, bool updateReadBuffer, bool raiseEvent);		this method is used to update the value of collection of variables /// <param name="variables">name value collections </param> /// <param name="updateReadBuffer">immediate update</param> /// <param name="raiseEvent">Need to raise VaribleChange event</param>

4.3) WVariables (Implement WAddIn, IWVariables,IWVariablesEx, IWVariablesEx2)

Methods	Properties	Description
	public VAR_TYPE Type { get; set; }	Get/Set the Variable Type
public WVariables()		Constructor
protected internal virtual bool OnBeforeUpdateVariable		Occurs before the variable is updated in buffer. When the returning value is true then base class will update the data in buffer else do nothing.
protected internal virtual void OnBeforeAddVariable		Occurs before the variable is updated in buffer.
protected internal virtual void OnBeforeDeleteVariable		Occurs before the variable is updated in buffer.
public abstract override void StartUp();		Call on the creation of Addin object
public abstract override System.Windows.Forms.DialogResult Editor(bool showDialog)		Return the DialogResult to execute the addins operation on the basis of return result. • This mainly used to show a dialog box to set the Config of the addin instance data through GUI., and return the result as per the selection. • User can show the dialog only if bool showDialog is true, else don't show the dialog popup (if any). • Initialize the IWAddin with the data contained in the Config. • User has to create the buffer according to the addin data. Call the Scene.CreateDataSource()/ CPlayoutEngine.CreateDataSource() to create buffer according to scope(global/local) of addin.
public sealed override void ShutDown()		Call when Addin is deleted.

protected internal abstract void OnShutDown()		Calls when Addin is deleted.
public bool CreateVariable		Creates the variable with given VariableOptionsBehaviour
public bool DeleteVariable		Delete the variable from the IWVariables. • Return true or false on the basis of variable delete or not • User has to update the buffer data with the empty value of this variable. • User has to raise the SceneGraph/Engine RaiseVariableChange event to reflects the effect in designer.
public bool UpdateVariable		Update the variable values on the basis on provided variable name and return true/false to determine whether the value is updated or not.
public bool RenameVariable		Rename the old variable name with the given name. • Return true or false on the basis of variable rename or not • User has to raise the SceneGraph/Engine RaiseVariableChange event to reflects the effect in designer.
public VarInfo GetVariable		Gets the VarInfo, contains the information of variable on the basis of provided name (that are created in this IWVariables).
public VarInfo[] GetVariable()		Gets the array of VarInfo, contains the information of all variables created in this IWVariables. • VarInfo contains the variable information like: Variable Name,Variable Value, type of variable and variablebehaviour.
	public override string Config {get; set; }	Get and Set the addin data.
	public new CPlayOutEngine Engine { get; set; }	Get and Set the Engine to the variable class and set IsEngine bool Property
	public new Scene SceneGraph { get; set; }	Get and Set the SceneGraph to the variable class and set IsEngine bool Property
bool		this method is used to update the value of collection of

UpdateVariables(Dictionary<string, string> variableNaneValeCollection, bool updateReadBuffer, bool raiseEvent);		varibles /// <param name="variables">name value collections</param> /// <param name="updateReadBuffer">immediate update</param> /// <param name="raiseEvent">Need to raise VaribleChange event</param>
bool UpdateVariableBehaviour(string name, VariableOptionsBehaviour optionBehaviour);		this method updates the behaviour of particular variable /// <param name="name">variable name</param> /// <param name="optionBehaviour">behaviour of variable</param>

5) IWDataTable

Methods	Properties	Description
	int ActiveIndex {get; set;}	Get and Set the active index for the current IWDataTable addin. • This helps to set the active page and its related data on the basis of RowsPerPage. • User must update the bufferdata on the basis of this value.
	bool Loop {get; set;}	Get and Set to determine whether to apply loop functionality on data of IWDataTable. • This helps to determine whether to loop the table data or not.
	int Pages {get;}	Get the total pages for the Table in IWDataTable. • Total page calculation will be dependent on Mode and RowPerPage property • Ex: If Mode is 'page', then total pages will be calculated as (TotalRows in DataTable/RowsPerPage)
	string Filter {get; set; }	Get and Set a value to get the data from the DataTable on the basis of given filter,
	Mode Mode {get; set; }	Get and Set the TableMode of the table in IWDataTable. • Mode contains "Page", "Carousel", "Slide" • In Page/Slide mode, each page contains the Rows count on the basis of RowPerPage • In carousel mode, each page contains single row. • This property is helpful to get or set the Next, Previous buffer data.

		enum Mode
		{
		carousel = 1,
		page = 2,
		slide=3,
		}

	int RowsPerPage { get; set; }	Gets or set the number of rows to be display per page. • User has to create the buffer according to the RowsPerPage and update data accordingly. • If RowPerPage =-1, means we write all the data in single page

	string Sort { get; set; }	Sort the table on the basis of column
	bool IsLastPage { get; }	Return true/false if current page of table is Last page
	bool Override { get; set; }	When this property is true, that means we override the WDataSet property like RowPerPage, Loop, Mode, etc for individual table. Otherwise Table RowperPage, Loop, Mode etc property value is same as WDataSet.
int GetRecords();		Get the total record exist in the DataTable in current IWDataTable.
int Next();		Move the record to the next page. Return the active index set according to it. • User update the buffer data on the basis of data contains in next page.
int Previous();		Move the record to the previous page. Return the active index set according to it. • User update the buffer data on the basis of data contains in previous page.
DataTable GetTable(bool activeIndex);		Get the DataTable containing data for current IWDataTable on the basis of bool value. • Bool indicates whether to get the data for current page or full data.
void SkipPages(List<int > index);		Skip the page for the given index value. While performing the next and previous operation on IWDataTable data.
	string TableName {get; set; }	Gets or sets the name of the table used in IWDataTable.

	int Index {get; set; }	Gets or sets the index of the Table used in IWDataTable. • This index is used to create the buffer, as each table contains unique index.
void Sync (DataTable dataTable);		Sync the buffer data with the given datatable data. • User call this to create new data table or when data update in table.
object GetValue(int row, int col);		Get the value of particular row and column.
bool SetValue(int row, int col, string value);		Set the value in the datatable at the given row and col position. • Call the update table to update the buffer data with the updated value.
void Refresh();		Refresh the IWDataTable with the datatable data. • Update the buffer information
event DataChangedHandler DataChanged;		Raise the event, if any data changed in buffer data.
event PageChangedHandler PageChanged;		Raise the event, when the active page changed. • Occurs when performing operation like Next(),Previous() or when set the activeindex value.
void BeginUpdate();		This method will help to cumulate the No. of operation like filter, sort etc. By using this method cumulative operation is not performed. To execute cumulative operation in single call, user need to invoke EndUpdate method. Note:: Calling of EndUpdate method is must after BeginUpdate.
void EndUpdate();		Calling of EndUpdate method is must after BeginUpdate method. This is used to execute cumulative operation in single call.

5.1 : **IWDataTableEx**

Methods	Properties	Description
public event EventHandler<DataRowChangeArgs>RowValuesChanged;		occur for each row that are updated in current page buffer.

5.2 : **IWDataReset**

Methods	Properties	Description
bool ResetOnCue { get; set; }		When user do the cue then set the ActivePageIndex =1 if ResetOnCue is true. By default, this property is true.

5.3 : **IWDataUpdateBuffer**

Methods	Properties	Description
bool UpdateBuffer { get; set; }		If Update Buffer property is true, then it will write into buffer to show the data into viewport
void OnUpdateBuffer()		This method is used to forcefully update the data into buffer

5.4) WDataTable Implement WVariables, IWDataTable, IWDataTableEx, IWDataReset, IWDataUpdateBuffer

Methods	Properties	Description
public event EventHandler<DataRowChangeArgs>RowValuesChanged;		Occur for each row that are updated in current page buffer.

public event PageChangedHandler PageChanged;		Occurs when current page Index gets changed.
public event DataChangedHandler DataChanged;		Occurs when any record is updated or the datasource gets changed.
public WDataTable()		Constructor
	protected internal string DataSetName { get; set; }	Get/Set the DataSetName
	public bool IsLastRecord { get; }	Gets a value whether the data table is on the last record or not.
	public int Index { get; set; }	Get and set the table index
	public Mode Mode {get; set; }	Get and Set the mode of the IWDataTable.
	public string Filter {get; set; }	Get and Set data from the datasource based on the set filter criteria.
	public int Pages { get; }	Obtains the number of pages, which is (total number of rows/number of row per page).
	public int RowsPerPage { get; set; }	Gets/Sets the number of rows to display in a page.
	public string TableName { get; set; }	Gets or sets the table name of DataTable
	string Sort { get; set; }	Sort the table data on the basis of column
	bool IsLastPage { get; }	Return true/false if current page of table is Last page
	bool Override { get; set; }	Get or sets value, to determine whether the given data table use override properties or not.

public abstract override void StartUp();		
public abstract override System.Windows.Forms.DialogResult Editor(bool showDialog)		Return the DialogResult to execute the addins operation on the basis of return result. • This mainly used to show a dialog box to set the Config of the addin instance data through GUI., and return the result as per the selection. • User can show the dialog only if bool showDialog is true, else don't show the dialog popup (if any). • Initialize the IWAddin with the data contained in the Config. • User has to create the buffer according to the addin data. • Call the Scene.CreateDataSource()/ CPlayoutEngine.CreateDataSource() to create buffer according to scope(global/local) of addin.
protected internal abstract override void OnShutDown();		
public int GetRecords()		Get the number of records in datasource.
public DataTable GetTable(bool activeIndex)		Get the datatable on the basis of activeindex data or not.
public DataTable GetTable(bool activeIndex, bool skipPrimaryColumn)		Get the datatable on the basis of activeindex, also to skip the primarycolumn data or not.
public int Next()		Move to next page record

public int Previous()		Move to previous page record.
public void Sync(DataTable dataTable)		Update the data in the buffer based on given data table
public void SkipPages(List<int> index)		Skips the page whose index is same as given index
public void SetColumnSchema(ColumnSchema[] columnSchema)		Sets the column schema for custom buffer size. For the given schema array.
public bool SetValue(int row, int col, string value);		Set the value of the current IWDataTable at the given row and col location
	public bool Loop { get; set; }	Get and Set the value to determine whether to apply the loop for the current data table or not, while changing the page
	public int ActiveIndex { get; set; }	Get and Set the current page index
public virtual void Refresh()		Refresh particular table
bool ResetOnCue {get; set; }		When user do the cue then set the ActivePageIndex =1 if ResetOnCue is true.
bool UpdateBuffer {get; set; }		If UpdateBuffer is true, then it will write into buffer to shown the data into viewport

void OnUpdateBuffer()		This method is used to forcefully update the data into buffer
void BeginUpdate();		This method will help to cumulate the No. of operation like filter, sort etc. By using this method cumulative operation is not performed. To execute cumulative operation in single call, user need to invoke EndUpdate method. Note:: Calling of EndUpdate method is must after BeginUpdate.
void EndUpdate();		Calling of EndUpdate method is must after BeginUpdate method. This is used to execute cumulative operation in single call.
object GetValue(int row, int col);		Get the value of particular row and column.

6) IWDataSet

Methods	Properties	Description
	DataSetOptionsBehaviour TablesBehaviour { get;}	Get the DataSetBehaviour to be performed on the IWDataTable contains in the current IWDataSet. • By default it set as None

		DataSetOptionsBehaviour
		{
		None = 0,
		AllowDelete = 1,
		AllowRename = 2,
		AllowUpdate = 4,
		AllowAll = AllowDelete | AllowRename | AllowUpdate
		}

IWDataTable GetTable(int index);		Get the IWDatatable which has the table index same as the given index.
	IWDataTable[] Tables { get; }	Get the array of all IWDataTable creates under this IWDataSet.
	Dictionary<string, IWDataTable> TableCollection { get; }	Gets the dictionary containing IWDataTable table name as key and IWDataTable as its value.
DataSet GetDataSet();		Get the current dataset of IWDataSet.
bool DeleteTable(int index);		Delete the IWDataTable from IWDataSet, which has the same tableindex as the given index. • Return true if deleted successfully. • Raise the SceneGraph/Engine RaiseTableCollectionChanged according to the scope of addin. • Also maintain the TableCollection dictionary.
bool RenameTable(int index, string name);		Rename the IWDataTable table name with the given name, whose tableindex is same as the given index. • Return true if rename successfully. • Raise the SceneGraph/Engine RaiseTableCollectionChanged according to the scope of addin.

		• Also maintain the TableCollection dictionary.
void Refresh();		Refresh the IWDataSet data, ➢ Update the bufferdata.

6.1) WDataSet Implement WVariables, IWDataSet,IWDataReset, IWDataUpdateBuffer

Methods	Properties	Description
WDataSet()		Constructor
	public virtual DataSetOptionsBehaviour TablesBehaviour { get; }	Gets or Sets the DataSetOptionsBehaviour for the IWDataTable
	public IWDataTable[] Tables { get; }	Gets the array of the IWDataTable exist in current IWDataSet.
	public Dictionary<string, IWDataTable> TableCollection { get; }	Get the dictionary of IWDataTable Objects having TableName as key and IWDataTable as value.
public System.Data.DataSet GetDataSet()		Get the current active DataSet
protected internal bool AddTable(DataTable dataTable, int index, int rowPerPage, bool loop,Mode mode)		Adds the datatable to the dataset • data table to be added • index of the table • Row per page to be display in the designer table pool • bool loop value, whether to repeat the data after last page or not • mode of the table.
public bool RenameTable(int tableIndex,string name)		Rename the table from the dataset, array and dictionary and update the bufferdata for this datatable

public IWDataTable GetTable(int tableIndex);		Get the IWDataTable at given index
public bool DeleteTable(int tableIndex)		Delete the table for the given table index.
protected internal void UpdateDataSet(DataTableInfo[] dataTables)		Update/Sync the current DataSet into buffer. This method is used to create and update the collection of tables.
public virtual void Refresh()		Refresh the dataset and update the data into buffer.
public abstract override System.Windows.Forms.DialogResult Editor(bool showDialog);		Return the Dialog Result to execute the addins operation on the basis of return result. • This mainly used to show a dialog box to set the Config of the addin instance data through GUI., and return the result as per the selection. • User can show the dialog only if bool showDialog is true, else don't show the dialog popup (if any). • Initialize the IWAddin with the data contained in the Config. • User has to create the buffer according to the addin data. • Call the Scene.CreateDataSource()/ CPlayoutEngine.CreateDataSource() to create buffer according to scope(global/local) of addin.

7) IWEventHelper

Methods	Properties	Description
	string Id { get; }	Gets the id for the IWEventHelper. This Id is used to get and set the Event Params.
	string Name { get; }	Gets the name for the current Event. • This helps to display the name in Designer
WEventControl Editor();		Get the BeeSys.Wasp3D.Hosting.WEventControl to display in Designer. • This UI helps to take input required for IWEventHelper.
NameValueCollection GetData(int ationsetId);		Gets the NameValueCollection containing data for the given actionsetid. • This helps to maintain the Do/Undo of Events Data.
bool Wire(int ationsetId);		Map/Wire the actionsetid with selected data. • Calls when the first action added to the current IWEventHelper. • User has to save the GUI/EventData with this actionsetid.
bool Wire(int actionsetId, NameValueCollection data);		Map/Wire the actionsetid with the given NameValueCollection data • This occurs while doing Do/Uno for the given actionsetid. • NameValueCollection data is same as user gave in GetData().
bool UnWire(int actionsetId);		UnWire/Remove the mapped actionsetid from the IWEventHelper Data.
bool IsWired(string addinInstanceId);		Determine whether any actions added for the given addin instanceId in this Event or not.
event EventDeleteActionSet DeleteActionSet;		Raise when any actionset to be deleted from IWEventHelper internally.

event EventDataChangedHandler DataChanged;		Raise when the EventParams Changed in IWEventHelper.

8) IWActionHelper

Methods	Properties	Description
	string Name { get; }	Gets the name for the current Action. • This string displays the name in Designer
	string Help { get; }	Gets the Help string for the action class. • This help string will display in Action UI.
WActionControl Editor();		Get the BeeSys.Wasp3D.Hosting.WActionControl to display in Designer action Popup. • This UI helps to take input required for IWActionHelper.
void SetName(int actionId, string name);		Set the name of the action for the given actionId. • As action name set by designer, while adding the actions. That name is sent to IWActionHelper to save the data.
string GetName(int actionId);		Gets the name of the action for the given actionid. • The name should be the same, as given in SetName() method for the given actionid.
void SetInstance(int actionId);		Set the Action control for the given action id. Action Control will display data according to given actionid in IWActionHelper.
bool Add(ref int actionId);		Add the new action data to the IWActionHelper and set the action id for this. • Return true if action added. • Action id to be generate by IWActionHelper for each action added. • Check for action id given in param, if it is "-1" then generate a new action id for the current data, else update the action data for the given actionid.

bool Update(int actionId);		Update the action data with the current action data for the given action id. • Return true if data update else return false.
bool Remove(int actionId);		Remove the action data from the IWActionHelper data, whose action id is same as the given action id. • Return true if action removed else return false.
bool ExecuteAction(int actionId, string nodeId, int trackId, int animTrackId);		Execute the action for the given param data. The action id given in param should be exist in IWActionHelper data. • Return true if action executed successfully • This method is called By Scene Graph when Scene Plays.
int Copy(int actionId);		Copy the actiondata for given actionid, and create a new action with the same data, create a unique actionid for new actiondata and return it.
void Reset();		Reset the IWActionHelper and set the ActionUI(if any) to its initial state.
void Init();		Initialize the IWActionHelper and UI to initial state and fill the Action UI data.
NameValueCollection GetParam(int actionId);		Get the NameValueCollection containing action data for the given actionid. • This data is used mainly to maintain the Do/Uno of action.
bool SetParam(int actionId, NameValueCollection param);		Set the Name Value Collection data for the given action id. • Return true if action data set successfully else return false. • Name Value Collection data is same as the data user gave in GetParam().

9) IWService

Methods	Properties	Description
List<Node> GetNodes(ObjectType objectType);		Get the list of node created in Designer on the basis of given object type. • Object Type specifies the type of objects Designer provide like Rectangle, Motion etc. • Node contain the information of object node i.e. name and id of the node.
List<Tracks> GetTracksByNodeId(string nodeId);		Gets the list of Tracks for the given node id. • Tracks give the track name and track id.
List<Tracks> GetTracksByNodeName(string nodeName);		Gets the list of Tracks for the given node name. • Tracks give the track name and track id.
List<AnimationSet> GetAnimSet();		Get the list of Animation set created in Drone Designer. • Animation Set give name, id of the animation and also specifies the group to which it is associated.
List<Addins> GetAddins(AddInType addinType, bool global)		Gets the list of Addins on the basis of given addin type and addin scope. • Addins contain the information of AddIn data like name, id, provider id, addin behaviour etc. • AddIn Type contain the type of addin like DataSet, Data Table etc.
List<VarInfo> GetSystemVariables();		Get the list of Var Info containing variable information like name, value etc. of variables which are System Variables.
List<VarInfo> GetAddinVariables(string addinId, VAR_TYPE variableType);		Get the list of Var Info containing variable information like name, value etc. based on the provided addin id and type.
List<Material> GetMaterials(MaterialType materialType);		Get the list of Node containing material information on the basis of given Material Type. • Material Type specifies the type of material designer support like Standard, Chroma etc. • Material gives the name and id of the Material

Appearance GetAppearance();		• Get the appearance of Drone Designer UI like selection color, backcolor, forecolor etc. • This helps the user to set their UI look and feel similar to Drone Designer look and feel.
DataTable GetTable(string addinId, int tableIndex, bool activeIndex);		Get the Data Table for the given addin id, whose table index is same as given index. • bool active index specifies, whether to get Data Table data on the basis of active index in Drone Designer or not.
List<Shader> GetShaders(TextureType textureType);		Gets the list of shaders applying for given texture type in designer. • Texture Type tell the type of texture exist in designer like Blur, RT etc. • Shader give the name and value of shader
List<Texture> GetTextures(TextureType textureType);		Get the list of Texture for the given Texture type. ⅄ Texture gives the name and id of the texture
List<TableInfo> GetTableInfo(string addinId);		Get the list of table Info for the tables exist in IWAddIn whose addin id is same as the given addinid.
string GetLocale();		Gets the locale used in Designer for localization.

List<EventParam> GetEventParams();		Get the list of Events Params used in Designer.
		public struct EventParam
		{
		public string ParamName;
		public string BufferName;
		public string ParamType;
		}

List<Trigger> GetTriggers(TriggerType type);		Get the list of triggers used in Designer on the basis of given Trigger Type. public enum TriggerType

		{
		Predefined = 1,
		Custom = 2,
		Named = 4,
		All = TriggerType.Predefined | TriggerType.Named | TriggerType.Custom
		}

		public class Trigger
		{
		public string Id { get; set; }
		public string Name { get; set; }
		public TriggerType Type { get; set; }
		public bool Self { get; set; }
		}

List<Level> GetLevels();		Gets the levels used in Designer,
List<Node> GetChildNodes(string nodeid);		Get the list of child node created under the given node.

10) IWParam

Methods	Properties	Description
bool CreateParam(string name, string data, VAR_TYPE variableType);		This is used to create the event param , determine whether the param created with the given parameters or not.
bool UpdateParam(string name, string value);		This is used to update the event param value , determine whether the given param updated with the updated value or not.
ParamInfo[] GetParams();		Gets the array of Param Info containing information of params created for the given IWParam

		public class ParamInfo : Param
		{
		public string Value;
		}

		public class Param
		{
		public string Name;
		public VAR_TYPE Type;
		}

ParamInfo GetParam(string paramName);		Get the paraminfo for the given paramName