# Window Manager Diagram I

## MetaplaceClient
**+swfRef:** WindowManager
...

## WindowManagerController
+**model**: Model
+**view**: View
+**assetManager**: AssetManager
- **windowControllers**: Vector<WindowController>

+**getWindowController**(target:Str):WindowController
+addWindow(**window**: IWindow):void
+removeWindow(**winCtrlr**: WindowController):void
+getLayer(layerName:String):Sprite
+sendResponsePacket(type:String, json:Object):void
+givePayloadToWindow(targetWindow: String):void

See *Window Manager Diagram II*

## JsonParser
See *Window Manager Diagram II*

## AssetManager
See *AssetManager Diagram*

## WindowController
+**view**: IWindow
+**windowVO**: WindowVO

+sendPayloadEvent(event:payloadEvent)
+receivePayload(event:PaytloadEvent)
-updatePlayerVO(event:Event):void
+sendResponsePacket(type:String);
+closeWindow():void

## <<interface>> IWindow, Window
+**modal**: ModelBlockerVO
+movable: Boolean
+persist: Boolean
+unique: Boolean
+**player**: IPlayerVO
+**assetManager**: AssetManager
+initializationPayload:Object
+assetPath:String

+setupWindow():void
+sendPayload(json:Object):void
+sendPayloadToWindow(json, window:String)
+receivedPayload(**event**: PayloadEvent):void
+closeWindow():void
+destroyWindow()

## LocalizerHelper
+defaultFont: String
See *Localizer Diagram*

## ToolTip
+font: String, +fontSize: uint
See *Window Manager Diagram II*

## WindowVO
See *Window Manager Diagram II*

## <<event>> PayloadEvent
+type: String
+**player**: IPlayerVO
+jsonPayload:Object

## WindowManager
- **model:** WindowManagerIModel
- **view:** WindowManagerView
+**controller:** WindowManagerController
- **jsonParser:** JsonParser
+windowManagerBounds: Rectangle
+**instance:** WindowManager

+setLocalTableCss():void
+setLoadingStatus():void
+setRendererInfo():void
- sendWindowManagerReady():void
+sendJson():void
+receivedJson():void

## WindowManagerView
+addWindow(**window**: IWindow) :void
+removeWindow(**window**: IWindow): void
+addLayer(layerName:String):void
+getLayer(layerName:String): void
+showBlocker(color:uint): void
+hideBlocker(): void

## <<event>> PlayerVOUpdatedEvent
+type: String

## Configuration
+LOG_LEVEL:String
+WEB_SCREEN_WIDTH:Number
+WEB_SCREEN_HEIGHT:Number
+DEFAULT_FONT:String
+TOOLTIP_FONT:String
...
+screenWidth:Number
+screenHeight:Number

+xPixel(xPercent:Number):Number
+yPixel(yPercent:Number):Number

## WindowManagerModel
+**player**: IPlayerVO
+layers:Vector <Sprite>

## <<interface>> IPlayerVO, PlayerVO
See *Window Manager Diagram II*

## GameCoupler
-**game**: IGame

+receivedPayload(**event**: PayloadEvent):void
...

## PlayFrames
- **playFrames**: Dictionary<PlayFrames>

+**play**(clip: MovieClip): PlayFrames

## ProgressBar
+value, +minValue, +maxValue: Number
See *AssetManager Diagram*

## RoomToRoomScreen
- **progressBar**: ProgressBar
+percentDone: Number

+showProgressBar():void
+removeProgressBar():void

## LoadProgressScreen
- **progressBar**: ProgressBar
+percentDone: Number
+loadingText: String

+showProgressBar():void
+removeProgressBar():void

## ButtonController
+view: MovieClip
+enabled: Boolean,
+selected: Boolean
+text: String

+**ButtonController**(view: MovieClip): ButtonCtrl
+clickButton(pressDuradtion: uint):void
+onClick(funct: Function, args): void

## Animatem
+**instance:** Animatem
*See Animatem Diagram*

## SoundFxManager
+**instance**: SoundManager
See *AssetManager Diagram*

## <<interface>> IGame, Game
+**modal**: ModelBlockerVO
+movable: Boolean
+persist: Boolean
+unique: Boolean
+**player**: IPlayerVO
+**assetManager**: AssetManager
+initializationPayload:Object
+assetPath:String

+setupGame():void
+sendPayload(json:Object):void
+sendPayloadToWindow(json, window:String)
+receivedPayload(**event**: PayloadEvent):void
+closeGame():void
+destroyGame():void

## <<interface>> IMultigame, Multigame
+gameLoaded():void
+startMultigame(**playerVOs**:<IPlayerVO>):void
+playerExited(**playerVO**: IPlayerVO):void
+sendMultigamePayload(json:Object)

## LogAs
**+instance:** LogAs

+**warning**(message: String): LogAsEvent
+**info**(message: String): LogAsEvent
+**debug**(message: String): LogAsEvent

## LogHandler
+handleLog(**event**: LogAsEvent): void

## <<event>> LogAsEvent
+message: String
+level: uint
+stackTrace: String

## Alert
- **alerts**: Dictionary<Alert>

+**show**(message: String, icon: String): Alert

***Logging and Alerts***
***All packages use LogAs***

**Key to Objects**
- Metaplace
- WindowManager
- WM Shared Lib.
- UIFramework Utility

# Window Manager Diagram II

See *WindowManager Diagram I*

**WindowManager**
See *WindowManager Diagram I*

**WindowManagerController**
See *WindowManager Diagram I*

1 — 1

1

**JsonParser**
- **actionQueue**: ActionQueue

+parse(json:Object):void

1

**WindowControllerActionFactory**
+**createAction**(json:Object):WindowControllerAction

1

*Actions placed in priority queue*

**ActionQueue**
See *ActionQueue Diagram*

0..*

<<interface>> IAction,
**Action**
+actionType: String
+processed: Boolean
- timeOutDelay: Delay

+Action(actionType:String):void
+playAction():void
- timeoutError():void
+removeTimeout():void
+finishAction():void

*Incoming Json validated with JsonPacker*

**JsonPacker**
+getString(json:Object, field:String, ...):String
+getBoolean(json:Object, field:String, ...):Boolean
+getNumber(json:Object, field:String, ...):Number
+getUint(json:Object, field:String, ...):uint
+getObject(json:Object, field:String, ...):Object
+stripEmptyValues(json:Object):Object
+setValue(json:Object, field:String, ...):Object
+setObject(json:Object, field:String, ...):Object

**ClassPacker**
+fromJson(json:Object):*
+toJson(instance:*):Object
+toString(instance:*): String

**WindowControllerAction**
- **controller**: WindowManagerController

**LoadGameAction**
- **windowVO**: WindowVO

+**create**(json:Object):LoadGameAction, +playAction():void, ...

**LoadWindowAction**
- **windowVO**: WindowVO

+**create**(json:Object): LoadWindowAction, +playAction():void, ...

**CloseWindowAction**
- targetWindow: String

+**create**(json:Object):CloseWindowAction, +playAction():void

**MoveWindowAction**
- targetWindow:String, xPosition:Number, yPosition:Number, tween:String, ...

+**create**(json:Object):MoveWindowAction, +playAction():void, ...

**ShowTooltipAction**
- key:String, xPosition:Number, yPosition:Number

+**create**(json:Object), +playAction():void, +showToolTip(key, x, y), +hideTooltip()

**SkinTooltipAction**
- url:String, className:String, fps:Number

+**create**(json:Object):SkinTooltipAction, +playAction():void, ...

**AddLayerAction**
- layerName:String, addLayerAt:String

+**create**(json:Object):AddLayerAction, +playAction():void

**InitializeBusinessIntelligenceAction**
- uid:String, appScribeTag:String, networkID:String, playerLevel:uint, ...

+**create**(json:Object): InitializeBusinessIntelligenceAction, +playAction():void

**StartMultigameAction**
- targetWindow:String, **players**: Vector<IPlayerVO>

+**create**(json:Object):StartMultigameAction, +playAction():void

2..*

**UpdatePlayerVOAction**
- **player**: IPlayerVO

+**create**(json:Object): UpdatePlayerVOAction, +playAction():void

1 — 1

**PlayerExitedAction**
- **player**: IPlayerVO, targetWindow:String

+**create**(json:Object): PlayerExitedAction, +playAction():void

1

**SkinProgressBarAction**
- url:String, className:String, fullScreen:Boolean

+**create**(json:Object):SkinProgressBarAction, +playAction():void, ...

For a list of more actions see
https://docs.wdig.com/display/CPPIM/Window+Manager+Documentation

1 — 1

**WindowVO**
+windowUrl: String
+layerName:String
+assetPath:String
+assetName:String
+windowId:String
+initializationPayload:Object
+initialX:Number
+initialY:Number
+scale:Number
+displayWidth:Number
+displayHeight:Number

**Animatem**
+**instance**: Animatem
See *Animatem Diagram*

**ToolTip**
+font:String
- **toolTips**: Dictionary<ToolTip>

+**addToolTip**(displayObject): ToolTip
+**removeToolTip**(displayObject): ToolTip
+**getToolTip**(displayObject): ToolTip
+showToolTip(): void
+hideToolTip(): void

**SkinToolTip**
+**addToolTip**(displayObject): SkinToolTip

**BI**
+**instance:** BI
-**tracker**: Tracker

+initialize(uid:String, appScribeTag:String)
+logDataLoadingStart(stepLabel:String)
+logDataLoadingEnd(bytesLoaded: uint)
+logSocialCallRequest(label:String):void
+logPopup(step: String, code: String)
...

1

**Tracker [ Playdom BI ]**
+initialize(uid:String, appScribeTag:String)
...

<<interface>> IPlayerVO,
**PlayerVO**
+name: String
+uid:String
+color:uint
+position:int
+accountType:String
+age:uint
+member:Boolean

**LoadProgressScreen**
+setLoadProgressScreen(MovieClip)
+setLoadProgressFullScreen(MovieClip)
See *WindowManager Diagram I*

**Key to Objects**

- Playdom
- WindowManager
- WM Shared Lib.
- UIFramework Utility

## Window Manager Diagram Notes

For those of you unfamiliar with UML,

- Unified Modeling Language (UML) is a standardized language for diagramming object-oriented software.
- Solid lines with arrows represent interaction between classes where a reference to the object is maintained.
- Dotted lines represent calls to static classes or singleton instances where a reference by the calling object need not be maintained.
- The position of the filled diamond shows which class owns a reference to what object, the number next to the diamond represents how many of that object.
- "0..*" means from zero to many of an object. Where the "*" wildcard denotes many. Similarly "2..4" would represent from 2 to 4.
- Static classes and methods are shown underlined.
- An underlined instance static variable represents a Singleton design pattern.
- An open arrow/triangle represents inheritance. e.g. Multigame extends Game.
- Variables and methods are shown with a prefix, where '+' denotes public, and '-' private.
- On these diagrams a solid arrow with an open circle at the beginning denotes direction of event dispatching and listening. This is not a UML convention.
- Any variables that are an instance of a class marked on the diagram, or methods that return an instance of a class marked on the diagram, are shown in bold.

Other matters of interest concerning the Window Manager,

- Utilities are shown in blue and are "stand-alone" packages entirely decoupled from the rest of the codebase. Generally (but not always) they will be a Singleton or even entirely static.
- Where possible, calls to utilities operate on a "fire and forget" bases, in that the utility entirely handles and cleans up processes asked of it. The instigating class usually is not required to maintain a reference to the utility.
- Utilities maintain options for *both* function callback and event dispatching. When using a utility there are generally helper methods that involve function callbacks, but the developer may instead listen for events dispatched.
- There is an emphasis on modular design encouraging refactoring on a piece by piece basis. The utilities are stand alone and decoupled for this reason, and also for reasons of reusability in other projects.
- Window Manager Actions are "plug-ins" that are also intended to be highly modular, with each action being self contained as much as possible. When extending WindowManager it should be done using a modular WindowManager action which can easily be added or removed on an as needed basis.
- Only the main twelve WindowManagerAction are displayed on the diagram.[1]
- Flash works on a "1st come 1st served" basis when it comes to classes compiled into .swf files. Whichever version of a class is called first is the version that must be used by all instances thereafter. By having all UIFramework Utilities compiled into WindowManager, which is loaded first, it ensures the environment runs with the latest version of the classes.
- The Game class intentionally doesn't extend Window. This was done as we wanted to hide underlying Window implementation from external game developers. A GameCoupler class is used to couple the Game class to Window, allowing Game to be treated generically as a Window by the WindowManager.
- Notice that there are two LoadProgressScreens. One is fullscreen the other regular.
- All packages call the LogAs utility for logging purposes.

---

[1] The other actions are DisplayCoinsAwardedAction, AttachMetaplaceObjectAction, JsonPayloadAction, LoadFontAction, SkinRoomToRoomAction, ChangeLanguageAction, DetachMetaplaceObjectAction, HideEdgeBlockerAction, HideLayerAction, HideTooltipAction, HideWindowAction, JsonPayloadUnfinishedAction, LinkWindowCacheAction, MousePositionAction, PushDevonDataToMpAction, PushLocalizationDataToMPAction, SetCursorAction, SetFontPathAction, SetWorldIdAction, ShowEdgeBlockerAction, ShowLayerAction, ShowWindowAction, and SuppressLogAlertsAction.