



Delta3D Version 2.4.0

# **dtDIS::**

## **Reference Manual**



# Contents

---

<b>1</b>	<b>Main Page</b>	<b>1</b>
<b>2</b>	<b>Todo List</b>	<b>3</b>
<b>3</b>	<b>Directory Documentation</b>	<b>5</b>
3.1	inc/dtDIS/plugins/default/ Directory Reference . . . . .	5
3.2	src/dtDIS/plugins/default/ Directory Reference . . . . .	6
3.3	src/dtDIS/ Directory Reference . . . . .	7
3.4	inc/dtDIS/ Directory Reference . . . . .	8
3.5	inc/ Directory Reference . . . . .	9
3.6	inc/dtDIS/plugins/ Directory Reference . . . . .	10
3.7	src/dtDIS/plugins/ Directory Reference . . . . .	11
3.8	src/ Directory Reference . . . . .	12
<b>4</b>	<b>Namespace Documentation</b>	<b>13</b>
4.1	DIS Namespace Reference . . . . .	13
4.2	dtCore Namespace Reference . . . . .	14
4.3	dtDAL Namespace Reference . . . . .	15
4.4	dtDIS Namespace Reference . . . . .	16
4.4.1	Detailed Description . . . . .	17
4.4.2	Enumeration Type Documentation . . . . .	17
4.4.2.1	AcknowledgeFlag . . . . .	17
4.4.2.2	DomainType . . . . .	17
4.4.2.3	ResponseFlag . . . . .	18
4.4.3	Function Documentation . . . . .	18
4.4.3.1	CreateComponent . . . . .	18
4.4.3.2	DestroyComponent . . . . .	18
4.5	dtDIS::Articulation Namespace Reference . . . . .	19
4.5.1	Detailed Description . . . . .	19
4.5.2	Enumeration Type Documentation . . . . .	19
4.5.2.1	MotionType . . . . .	19
4.5.2.2	ParameterTypeDesignator . . . . .	19
4.5.2.3	PartType . . . . .	19
4.6	dtGame Namespace Reference . . . . .	20
4.7	dtUtil Namespace Reference . . . . .	21
4.7.1	Function Documentation . . . . .	21
4.7.1.1	find_multimap_pair . . . . .	21

<b>5</b>	<b>Class Documentation</b>	<b>23</b>
5.1	ActiveEntityControl Struct Reference . . . . .	23
5.1.1	Detailed Description . . . . .	23
5.1.2	Member Function Documentation . . . . .	23
5.1.2.1	AddEntity . . . . .	23
5.1.2.2	ClearAll . . . . .	23
5.1.2.3	GetActor . . . . .	23
5.1.2.4	GetEntity . . . . .	23
5.1.2.5	RemoveEntity . . . . .	24
5.2	ActorMapConfig Struct Reference . . . . .	25
5.2.1	Detailed Description . . . . .	25
5.2.2	Member Function Documentation . . . . .	25
5.2.2.1	AddActorMapping . . . . .	25
5.2.2.2	GetMappedActor . . . . .	25
5.2.2.3	RemoveActorMapping . . . . .	25
5.3	ActorUpdateToEntityState Class Reference . . . . .	26
5.3.1	Detailed Description . . . . .	26
5.3.2	Constructor & Destructor Documentation . . . . .	26
5.3.2.1	ActorUpdateToEntityState . . . . .	26
5.3.2.2	~ActorUpdateToEntityState . . . . .	26
5.3.3	Member Function Documentation . . . . .	26
5.3.3.1	Convert . . . . .	26
5.4	Connection Class Reference . . . . .	27
5.4.1	Detailed Description . . . . .	27
5.4.2	Member Function Documentation . . . . .	27
5.4.2.1	Connect . . . . .	27
5.4.2.2	Disconnect . . . . .	27
5.4.2.3	Receive . . . . .	27
5.4.2.4	Send . . . . .	27
5.5	ConnectionData Struct Reference . . . . .	28
5.5.1	Detailed Description . . . . .	28
5.5.2	Member Data Documentation . . . . .	28
5.5.2.1	application_id . . . . .	28
5.5.2.2	exercise_id . . . . .	28
5.5.2.3	ip . . . . .	28
5.5.2.4	MTU . . . . .	28
5.5.2.5	plug_dir . . . . .	28
5.5.2.6	port . . . . .	28
5.5.2.7	site_id . . . . .	28
5.6	ConnectionXMLHandler Class Reference . . . . .	29
5.6.1	Detailed Description . . . . .	29

5.6.2	Constructor & Destructor Documentation . . . . .	30
5.6.2.1	ConnectionXMLHandler . . . . .	30
5.6.2.2	~ConnectionXMLHandler . . . . .	30
5.6.3	Member Function Documentation . . . . .	30
5.6.3.1	characters . . . . .	30
5.6.3.2	endDocument . . . . .	30
5.6.3.3	endElement . . . . .	30
5.6.3.4	endPrefixMapping . . . . .	30
5.6.3.5	GetConnectionData . . . . .	30
5.6.3.6	ignorableWhitespace . . . . .	30
5.6.3.7	processingInstruction . . . . .	30
5.6.3.8	setDocumentLocator . . . . .	30
5.6.3.9	skippedEntity . . . . .	30
5.6.3.10	startDocument . . . . .	30
5.6.3.11	startElement . . . . .	30
5.6.3.12	startPrefixMapping . . . . .	30
5.6.4	Member Data Documentation . . . . .	30
5.6.4.1	DEFAULT_CONNECTION_DATA . . . . .	30
5.7	CreateEntityProcessor Class Reference . . . . .	31
5.7.1	Detailed Description . . . . .	31
5.7.2	Constructor & Destructor Documentation . . . . .	31
5.7.2.1	CreateEntityProcessor . . . . .	31
5.7.2.2	~CreateEntityProcessor . . . . .	31
5.7.3	Member Function Documentation . . . . .	31
5.7.3.1	Process . . . . .	31
5.8	DefaultPlugin Class Reference . . . . .	32
5.8.1	Detailed Description . . . . .	32
5.8.2	Constructor & Destructor Documentation . . . . .	32
5.8.2.1	DefaultPlugin . . . . .	32
5.8.2.2	~DefaultPlugin . . . . .	32
5.8.3	Member Function Documentation . . . . .	32
5.8.3.1	Finish . . . . .	32
5.8.3.2	Start . . . . .	32
5.9	EnginePropertyName Struct Reference . . . . .	33
5.9.1	Member Data Documentation . . . . .	33
5.9.1.1	ARTICULATION . . . . .	33
5.9.1.2	DEAD_RECKONING_ALGORITHM . . . . .	33
5.9.1.3	DOF_NODE_NAME . . . . .	33
5.9.1.4	ENTITY_LINEARY_VELOCITY . . . . .	33
5.9.1.5	ENTITY_LOCATION . . . . .	33
5.9.1.6	ENTITY_MARKING . . . . .	33

5.9.1.7	ENTITY_ORIENTATION . . . . .	33
5.9.1.8	GROUND_CLAMP . . . . .	33
5.9.1.9	LAST_KNOWN_LOCATION . . . . .	34
5.9.1.10	LAST_KNOWN_ORIENTATION . . . . .	34
5.9.1.11	RESOURCE_DAMAGE_DESTROYED . . . . .	34
5.9.1.12	RESOURCE_DAMAGE_OFF . . . . .	34
5.9.1.13	RESOURCE_DAMAGE_ON . . . . .	34
5.10	EntityMapXMLHandler Class Reference . . . . .	35
5.10.1	Detailed Description . . . . .	35
5.10.2	Constructor & Destructor Documentation . . . . .	35
5.10.2.1	EntityMapXMLHandler . . . . .	35
5.10.2.2	~EntityMapXMLHandler . . . . .	35
5.10.3	Member Function Documentation . . . . .	35
5.10.3.1	characters . . . . .	35
5.10.3.2	endDocument . . . . .	35
5.10.3.3	endElement . . . . .	35
5.10.3.4	endPrefixMapping . . . . .	35
5.10.3.5	ignorableWhitespace . . . . .	35
5.10.3.6	processingInstruction . . . . .	35
5.10.3.7	setDocumentLocator . . . . .	35
5.10.3.8	skippedEntity . . . . .	35
5.10.3.9	startDocument . . . . .	35
5.10.3.10	startElement . . . . .	35
5.10.3.11	startPrefixMapping . . . . .	35
5.11	EntityPropertyName Struct Reference . . . . .	36
5.11.1	Member Data Documentation . . . . .	36
5.11.1.1	APPEARANCE . . . . .	36
5.11.1.2	ENTITY_TYPE . . . . .	36
5.12	ESPduProcessor Class Reference . . . . .	37
5.12.1	Detailed Description . . . . .	37
5.12.2	Constructor & Destructor Documentation . . . . .	37
5.12.2.1	ESPduProcessor . . . . .	37
5.12.2.2	~ESPduProcessor . . . . .	37
5.12.3	Member Function Documentation . . . . .	37
5.12.3.1	AddActor . . . . .	37
5.12.3.2	ApplyFullUpdateToProxy . . . . .	37
5.12.3.3	Process . . . . .	37
5.12.3.4	SendPartialUpdate . . . . .	37
5.13	IDISPlugin Class Reference . . . . .	38
5.13.1	Detailed Description . . . . .	38
5.13.2	Constructor & Destructor Documentation . . . . .	38

5.13.2.1	~IDISPlugin	38
5.13.3	Member Function Documentation	38
5.13.3.1	Finish	38
5.13.3.2	Start	38
5.14	IMessageToPacketAdapter Class Reference	39
5.14.1	Detailed Description	39
5.14.2	Constructor & Destructor Documentation	39
5.14.2.1	~IMessageToPacketAdapter	39
5.14.3	Member Function Documentation	39
5.14.3.1	Convert	39
5.15	MasterComponent Class Reference	40
5.15.1	Detailed Description	41
5.15.2	Constructor & Destructor Documentation	41
5.15.2.1	MasterComponent	41
5.15.2.2	~MasterComponent	41
5.15.3	Member Function Documentation	41
5.15.3.1	DispatchNetworkMessage	41
5.15.3.2	GetIncomingMessage	41
5.15.3.3	GetIncomingMessage	41
5.15.3.4	GetOutgoingMessage	41
5.15.3.5	GetOutgoingMessage	41
5.15.3.6	GetSharedState	41
5.15.3.7	GetSharedState	41
5.15.3.8	LoadPlugins	42
5.15.3.9	OnAddedToGM	42
5.15.3.10	OnPluginLoaded	42
5.15.3.11	OnPluginUnloaded	42
5.15.3.12	OnRemovedFromGM	42
5.15.3.13	ProcessMessage	42
5.15.3.14	UnloadPlugins	42
5.16	NodeName Struct Reference	43
5.16.1	Detailed Description	43
5.16.2	Member Data Documentation	43
5.16.2.1	NODE_PRIMARY_GUN	43
5.16.2.2	NODE_PRIMARY_TURRET	43
5.16.2.3	NODE_SECONDARY_GUN	43
5.17	OutgoingMessage Class Reference	44
5.17.1	Detailed Description	44
5.17.2	Member Typedef Documentation	44
5.17.2.1	AdapterMap	44
5.17.3	Constructor & Destructor Documentation	44

5.17.3.1	OutgoingMessage . . . . .	44
5.17.4	Member Function Documentation . . . . .	44
5.17.4.1	AddAdaptor . . . . .	44
5.17.4.2	ClearData . . . . .	44
5.17.4.3	GetAdapters . . . . .	44
5.17.4.4	GetData . . . . .	44
5.17.4.5	Handle . . . . .	44
5.17.4.6	Handle . . . . .	44
5.17.4.7	RemoveAdaptor . . . . .	45
5.18	PluginManager Class Reference . . . . .	46
5.18.1	Detailed Description . . . . .	46
5.18.2	Member Typedef Documentation . . . . .	46
5.18.2.1	LibLoaderT . . . . .	46
5.18.2.2	LibraryRegistry . . . . .	46
5.18.2.3	PluginSignal . . . . .	47
5.18.2.4	RegistryEntry . . . . .	47
5.18.3	Member Function Documentation . . . . .	47
5.18.3.1	GetLoadedSignal . . . . .	47
5.18.3.2	GetRegistry . . . . .	47
5.18.3.3	GetRegistry . . . . .	47
5.18.3.4	GetUnloadedSignal . . . . .	47
5.18.3.5	LoadPlugin . . . . .	47
5.18.3.6	UnloadAllPlugins . . . . .	47
5.18.3.7	UnloadPlugin . . . . .	47
5.19	RemoveEntityProcessor Class Reference . . . . .	48
5.19.1	Detailed Description . . . . .	48
5.19.2	Constructor & Destructor Documentation . . . . .	48
5.19.2.1	RemoveEntityProcessor . . . . .	48
5.19.2.2	~RemoveEntityProcessor . . . . .	48
5.19.3	Member Function Documentation . . . . .	48
5.19.3.1	Process . . . . .	48
5.20	ResourceMapConfig Struct Reference . . . . .	49
5.20.1	Detailed Description . . . . .	49
5.20.2	Member Function Documentation . . . . .	49
5.20.2.1	AddResourceMapping . . . . .	49
5.20.2.2	GetMappedResource . . . . .	49
5.20.2.3	RemoveResourceMapping . . . . .	49
5.21	SharedState Class Reference . . . . .	50
5.21.1	Detailed Description . . . . .	50
5.21.2	Constructor & Destructor Documentation . . . . .	51
5.21.2.1	SharedState . . . . .	51

5.21.2.2	~SharedState . . . . .	51
5.21.3	Member Function Documentation . . . . .	51
5.21.3.1	GetActiveEntityControl . . . . .	51
5.21.3.2	GetActiveEntityControl . . . . .	51
5.21.3.3	GetActorMap . . . . .	51
5.21.3.4	GetActorMap . . . . .	51
5.21.3.5	GetApplicationID . . . . .	51
5.21.3.6	GetConnectionData . . . . .	51
5.21.3.7	GetCoordinateConverter . . . . .	51
5.21.3.8	GetCoordinateConverter . . . . .	51
5.21.3.9	GetResourceMap . . . . .	51
5.21.3.10	GetResourceMap . . . . .	51
5.21.3.11	GetSiteID . . . . .	51
5.21.3.12	SetApplicationID . . . . .	51
5.21.3.13	SetConnectionData . . . . .	51
5.21.3.14	SetCoordinateConverter . . . . .	51
5.21.3.15	SetSiteID . . . . .	51
5.22	ValueMap Struct Reference . . . . .	52
5.22.1	Detailed Description . . . . .	52
5.22.2	Member Function Documentation . . . . .	52
5.22.2.1	GetArticulationMotionPropertyName . . . . .	52
5.22.2.2	GetArticulationNodeName . . . . .	52
5.22.2.3	GetDeadReckoningModelPropertyValue . . . . .	52
5.22.2.4	GetRequiresGroundClamping . . . . .	53
5.23	XMLFiles Struct Reference . . . . .	54
5.23.1	Detailed Description . . . . .	54
5.23.2	Member Data Documentation . . . . .	54
5.23.2.1	XML_CONNECTION_SCHEMA_FILE . . . . .	54
5.23.2.2	XML_MAPPING_SCHEMA_FILE . . . . .	54
<b>6</b>	<b>File Documentation</b>	<b>55</b>
6.1	acknowledgeconstants.h File Reference . . . . .	55
6.2	activeentitycontrol.cpp File Reference . . . . .	56
6.3	activeentitycontrol.h File Reference . . . . .	57
6.4	actorupdatetoentitystate.cpp File Reference . . . . .	58
6.5	actorupdatetoentitystate.h File Reference . . . . .	59
6.6	articulationconstants.cpp File Reference . . . . .	60
6.7	articulationconstants.h File Reference . . . . .	61
6.8	connection.cpp File Reference . . . . .	62
6.9	connection.h File Reference . . . . .	63
6.10	containerutils.h File Reference . . . . .	64

6.11	createdestroypolicy.h File Reference . . . . .	65
6.12	createdestroypolicy.inl File Reference . . . . .	66
6.13	createentityprocessor.cpp File Reference . . . . .	67
6.14	createentityprocessor.h File Reference . . . . .	68
6.15	defaultplugin.cpp File Reference . . . . .	69
6.16	defaultplugin.h File Reference . . . . .	70
6.17	disxml.cpp File Reference . . . . .	71
6.18	disxml.h File Reference . . . . .	72
6.19	dllfinder.h File Reference . . . . .	73
6.20	dtdisdefaultpluginexport.h File Reference . . . . .	74
	6.20.1 Define Documentation . . . . .	74
	6.20.1.1 DT_DIS_DEFAULT_EXPORT . . . . .	74
6.21	dtdisexport.h File Reference . . . . .	75
	6.21.1 Define Documentation . . . . .	75
	6.21.1.1 DT_DIS_EXPORT . . . . .	75
6.22	entityidcompare.cpp File Reference . . . . .	76
6.23	entityidcompare.h File Reference . . . . .	77
6.24	entitytypeconstants.h File Reference . . . . .	78
6.25	espduapplicator.cpp File Reference . . . . .	79
6.26	espduapplicator.h File Reference . . . . .	80
6.27	espduprocessor.cpp File Reference . . . . .	81
6.28	espduprocessor.h File Reference . . . . .	82
6.29	hasproperty.cpp File Reference . . . . .	83
6.30	hasproperty.h File Reference . . . . .	84
6.31	idisplugin.cpp File Reference . . . . .	85
6.32	idisplugin.h File Reference . . . . .	86
6.33	imessagetopacketadapter.cpp File Reference . . . . .	87
6.34	imessagetopacketadapter.h File Reference . . . . .	88
6.35	libraryregistry.h File Reference . . . . .	89
6.36	mainpage.h File Reference . . . . .	90
	6.36.1 Detailed Description . . . . .	90
6.37	mastercomponent.cpp File Reference . . . . .	91
6.38	mastercomponent.h File Reference . . . . .	92
6.39	outgoingmessage.cpp File Reference . . . . .	93
6.40	outgoingmessage.h File Reference . . . . .	94
6.41	pluginmanager.cpp File Reference . . . . .	95
6.42	pluginmanager.h File Reference . . . . .	96
6.43	pluginsymbols.cpp File Reference . . . . .	97
6.44	pluginsymbols.h File Reference . . . . .	98
6.45	propertyname.cpp File Reference . . . . .	99
6.46	propertyname.h File Reference . . . . .	100

---

6.47	removeentityprocessor.cpp File Reference . . . . .	101
6.48	removeentityprocessor.h File Reference . . . . .	102
6.49	sharedstate.cpp File Reference . . . . .	103
6.50	sharedstate.h File Reference . . . . .	104
6.51	valuemaps.cpp File Reference . . . . .	105
6.52	valuemaps.h File Reference . . . . .	106



## Main Page

---

Delta3D is an Open Source engine which can be used for games, simulations, or other graphical applications.

The Delta3D framework exists as a number of modules, each sitting in its own library, enclosed within its own namespace. At the very core lies the **dtCore** (p. 14) library. This contains basic, low-level functionality which is mostly required for all 3D applications written in C++.

Around and alongside this sit other supporting libraries, such as **dtUtil** (p. 21) (containing reusable features which are useful for most applications), dtTerrain (for rendering terrain databases), **dtGame** (p. 20), dtNet, etc.

Extensive online documentation is available from the Delta3D Docs section to help in using Delta3D.

The project's original reference guides generated by Doxygen from the source code may be viewed at the Delta3D API Documentation section.

To download source code, binaries, dependencies and sample datasets visit the Delta3D Downloads page.

For more about dependencies see the Delta3D Dependencies page.

The documentation you are looking at can be downloaded from [www.3draum.ch](http://www.3draum.ch).

Enjoy!



**Todo List**

---



# Directory Documentation

---

### 3.1 inc/dtDIS/plugins/default/ Directory Reference

#### Files

- file `actorupdatetoentitystate.h`
- file `createentityprocessor.h`
- file `defaultplugin.h`
- file `dtdefaultpluginexport.h`
- file `espduapplicator.h`
- file `espduprocessor.h`
- file `pluginsymbols.h`
- file `removeentityprocessor.h`

## 3.2 src/dtDIS/plugins/default/ Directory Reference

### Files

- file **actorupdatetoentitystate.cpp**
- file **createentityprocessor.cpp**
- file **defaultplugin.cpp**
- file **espduapplicator.cpp**
- file **espduprocessor.cpp**
- file **pluginsymbols.cpp**
- file **removeentityprocessor.cpp**

## 3.3 src/dtDIS/ Directory Reference

### Directories

- directory **plugins**

### Files

- file **activeentitycontrol.cpp**
- file **articulationconstants.cpp**
- file **connection.cpp**
- file **disxml.cpp**
- file **entityidcompare.cpp**
- file **hasproperty.cpp**
- file **idisplugin.cpp**
- file **imessagetopacketadapter.cpp**
- file **mastercomponent.cpp**
- file **outgoingmessage.cpp**
- file **pluginmanager.cpp**
- file **propertyname.cpp**
- file **sharedstate.cpp**
- file **valuemaps.cpp**

## 3.4 inc/dtDIS/ Directory Reference

### Directories

- directory **plugins**

### Files

- file **acknowledgeconstants.h**
- file **activeentitycontrol.h**
- file **articulationconstants.h**
- file **connection.h**
- file **containerutils.h**
- file **createdestroypolicy.h**
- file **createdestroypolicy.inl**
- file **disxml.h**
- file **dllfinder.h**
- file **dtdisexport.h**
- file **entityidcompare.h**
- file **entitytypeconstants.h**
- file **hasproperty.h**
- file **idisplugin.h**
- file **imessagetopacketadapter.h**
- file **libraryregistry.h**
- file **mainpage.h**
- file **mastercomponent.h**
- file **outgoingmessage.h**
- file **pluginmanager.h**
- file **propertyname.h**
- file **sharedstate.h**
- file **valuemaps.h**

## 3.5 inc/ Directory Reference

### Directories

- directory dtDIS

## 3.6 inc/dtDIS/plugins/ Directory Reference

### Directories

- directory **default**

## 3.7 src/dtDIS/plugins/ Directory Reference

### Directories

- directory **default**

## 3.8 src/ Directory Reference

### Directories

- directory dtDIS

# Namespace Documentation

---

## 4.1 DIS Namespace Reference

## 4.2 dtCore Namespace Reference

## 4.3 dtDAL Namespace Reference

## 4.4 dtDIS Namespace Reference

the Delta3D support framework for the **DIS** (p. 13) network protocol.

### Namespaces

- namespace **Articulation**  
*the scope for articulation specific constants, lacking in the **DIS** (p. 13) dependency, and other tools.*

### Classes

- struct **ActiveEntityControl**  
*provides a single point for associating known entities & actors.*
- struct **ActorMapConfig**  
*a structure to maintain the one-to-one relationship between the **DIS::EntityID** and the **dtDAL::ActorType**.*
- class **ActorUpdateToEntityState**  
*responsible for translating **ActorUpdateMessage** instances to **EntityStatePdu** instances.*
- class **Connection**  
*Makes a multicast connection to support **DIS** (p. 13) networks.*
- struct **ConnectionData**  
*the information needed to connect to the **DIS** (p. 13) network.*
- class **ConnectionXMLHandler**  
*Reads the connection info from a config file.*
- class **CreateEntityProcessor**  
*the plugin's model of how to respond to the **DIS::CreateEntityPdu** message.*
- class **DefaultPlugin**  
*the plugin to support various **DIS::Pdu** types.*
- struct **EnginePropertyName**
- class **EntityMapXMLHandler**  
*Stores configuration data loaded from a file for the purposes of mapping **DIS** (p. 13) to Delta3D.*
- struct **EntityPropertyName**
- class **ESPduProcessor**  
*the interface class responsible for applying data from a **DIS::EntityStatePdu** packet.*
- class **IDISPlugin**  
*the interface for all **DIS** (p. 13) plugins*
- class **IMessageToPacketAdapter**  
*the interface for translating Delta3D messages into **DIS** (p. 13) packets.*
- class **MasterComponent**  
*Supports a framework to translate **DIS** (p. 13) PDUs into **dtGame::Message** instances.*
- class **OutgoingMessage**  
*A framework for translating **dtGame::Message** instances to PDUs.*
- class **PluginManager**

*manages a container of available plugins.*

- class **RemoveEntityProcessor**  
*the plugin's model of how to respond to the DIS::RemoveEntityPdu message.*
- struct **ResourceMapConfig**  
*a structure to maintain the relationship between the DIS::EntityID and the resources available.*
- class **SharedState**  
*The data to be shared among plugins.*
- struct **ValueMap**  
*a name scope for all sorts of mappings*
- struct **XMLFiles**  
*constant definitions for the official schema files.*

## Enumerations

- enum **AcknowledgeFlag** { **ACKNOWLEDGE\_CREATE\_ENTITY** = 1, **ACKNOWLEDGE\_REMOVE\_ENTITY** = 2, **ACKNOWLEDGE\_START\_RESUME** = 3, **ACKNOWLEDGE\_STOP\_FREEZE** = 4 }
- enum **DomainType** {  
**DOMAIN\_LAND** = 1, **DOMAIN\_AIR** = 2, **DOMAIN\_SURFACE** = 3, **DOMAIN\_SUBSURFACE** = 4,  
**DOMAIN\_SPACE** = 5 }
- enum **ResponseFlag** { **RESPONSE\_OTHER** = 0, **RESPONSE\_ABLE\_TO\_COMPLY** = 1, **RESPONSE\_UNABLE\_TO\_COMPLY** = 2 }

## Functions

- DT\_DIS\_DEFAULT\_EXPORT **IDISPlugin \* CreateComponent** ()
- DT\_DIS\_DEFAULT\_EXPORT void **DestroyComponent** (**IDISPlugin \*plugin**)

### 4.4.1 Detailed Description

the Delta3D support framework for the **DIS** (p. 13) network protocol.

### 4.4.2 Enumeration Type Documentation

#### 4.4.2.1 enum AcknowledgeFlag

Enumerator:

**ACKNOWLEDGE\_CREATE\_ENTITY**  
**ACKNOWLEDGE\_REMOVE\_ENTITY**  
**ACKNOWLEDGE\_START\_RESUME**  
**ACKNOWLEDGE\_STOP\_FREEZE**

#### 4.4.2.2 enum DomainType

Enumerator:

**DOMAIN\_LAND**  
**DOMAIN\_AIR**  
**DOMAIN\_SURFACE**  
**DOMAIN\_SUBSURFACE**  
**DOMAIN\_SPACE**

#### 4.4.2.3 enum ResponseFlag

Enumerator:

*RESPONSE\_OTHER*

*RESPONSE\_ABLE\_TO\_COMPLY*

*RESPONSE\_UNABLE\_TO\_COMPLY*

#### 4.4.3 Function Documentation

4.4.3.1 dtDIS::IDISPlugin \* CreateComponent ()

4.4.3.2 void DestroyComponent (dtDIS::IDISPlugin \* *plugin*)

## 4.5 dtDIS::Articulation Namespace Reference

the scope for articulation specific constants, lacking in the **DIS** (p. 13) dependency, and other tools.

### Classes

- struct **NodeName**

*a static definition of the scene graph node names to be used for articulated parts.*

### Enumerations

- enum **MotionType** { **MOTION\_AZIMUTH** = 11, **MOTION\_AZIMUTH\_RATE** = 12, **MOTION\_ELEVATION** = 13 }
- enum **ParameterTypeDesignator** { **ARTICULATED\_PART** = 0, **ATTACHED\_PART** = 1 }
- enum **PartType** { **PART\_PRIMARY\_TURRET** = 4096, **PART\_PRIMARY\_GUN** = 4416, **PART\_SECONDARY\_GUN** = 6016 }

#### 4.5.1 Detailed Description

the scope for articulation specific constants, lacking in the **DIS** (p. 13) dependency, and other tools.

#### 4.5.2 Enumeration Type Documentation

##### 4.5.2.1 enum MotionType

Enumerator:

**MOTION\_AZIMUTH**  
**MOTION\_AZIMUTH\_RATE**  
**MOTION\_ELEVATION**

##### 4.5.2.2 enum ParameterTypeDesignator

Enumerator:

**ARTICULATED\_PART**  
**ATTACHED\_PART**

##### 4.5.2.3 enum PartType

Enumerator:

**PART\_PRIMARY\_TURRET**  
**PART\_PRIMARY\_GUN**  
**PART\_SECONDARY\_GUN**

## 4.6 dtGame Namespace Reference

## 4.7 dtUtil Namespace Reference

### Functions

- template<class MType >  
MType::iterator **find\_multimap\_pair** (MType &mmap, const typename MType::key\_type &key, const  
typename MType::mapped\_type &value)  
*find the iterator for the unique key-value pair.*

### 4.7.1 Function Documentation

#### 4.7.1.1 MType::iterator dtUtil::find\_multimap\_pair (MType & *mmap*, const typename MType::key\_type & *key*, const typename MType::mapped\_type & *value*) [inline]

find the iterator for the unique key-value pair. Parameters

***mmap*** the container instance to be searched.

***key*** the container's key to be found.

***the*** container's value matching key to be found.

Returns the found iterator, or the end of the container.



# Class Documentation

---

## 5.1 ActiveEntityControl Struct Reference

provides a single point for associating known entities & actors.

```
#include <inc/dtDIS/activeentitycontrol.h>
```

### Public Member Functions

- bool **AddEntity** (const DIS::EntityID &eid, const dtCore::Uniqueld &id)  
*relate an EntityID with an Actor*
- void **ClearAll** ()  
*remove all state data.*
- const dtCore::Uniqueld \* **GetActor** (const DIS::EntityID &eid) const  
*finds the associated Actor*
- const DIS::EntityID \* **GetEntity** (const dtCore::Uniqueld &uid) const  
*finds the associated Entity*
- bool **RemoveEntity** (const DIS::EntityID &eid, const dtCore::Uniqueld &id)  
*remove a relation for the EntityID and Actor*

### 5.1.1 Detailed Description

provides a single point for associating known entities & actors. Provides quick look-ups given either an entity or an actor by keeping 2 maps in sync.

### 5.1.2 Member Function Documentation

#### 5.1.2.1 bool AddEntity (const DIS::EntityID & *eid*, const dtCore::Uniqueld & *id*)

relate an EntityID with an Actor Returns 'false' when any relation previously existed; 'true' if the relation was added.

#### 5.1.2.2 void ClearAll ()

remove all state data.

#### 5.1.2.3 const dtCore::Uniqueld \* GetActor (const DIS::EntityID & *eid*) const

finds the associated Actor Returns NULL when the EntityID has no matching Actor

#### 5.1.2.4 const DIS::EntityID \* GetEntity (const dtCore::Uniqueld & *uid*) const

finds the associated Entity Returns NULL when the proxy has no matching EntityID

**5.1.2.5 bool RemoveEntity (const DIS::EntityID & *eid*, const dtCore::Uniqueld & *id*)**

remove a relation for the EntityID and Actor Returns 'false' when no relation previously existed; 'true' if the relation was removed.

The documentation for this struct was generated from the following files:

- **activeentitycontrol.h**
- **activeentitycontrol.cpp**

## 5.2 ActorMapConfig Struct Reference

a structure to maintain the one-to-one relationship between the DIS::EntityID and the dtDAL::ActorType.

```
#include <inc/dtDIS/sharedstate.h>
```

### Public Member Functions

- **bool AddActorMapping** (const DIS::EntityType &eid, const dtDAL::ActorType \*at)  
*Adds a mapping from the DIS::EntityID to the dtDAL::ActorType if no mapping exists.*
- **bool GetMappedActor** (const DIS::EntityType &eid, const dtDAL::ActorType \*&toWrite)  
*Introduces the ActorType mapped to the EntityID.*
- **bool RemoveActorMapping** (const DIS::EntityType &eid)  
*Takes the one to one mapping from the container for the supplied key.*

### 5.2.1 Detailed Description

a structure to maintain the one-to-one relationship between the DIS::EntityID and the dtDAL::ActorType.

### 5.2.2 Member Function Documentation

#### 5.2.2.1 bool AddActorMapping (const DIS::EntityType & eid, const dtDAL::ActorType \* at)

Adds a mapping from the DIS::EntityID to the dtDAL::ActorType if no mapping exists. Returns true if no previous mapping existed. false if a mapping existed already.

#### 5.2.2.2 bool GetMappedActor (const DIS::EntityType & eid, const dtDAL::ActorType \*& toWrite)

Introduces the ActorType mapped to the EntityID. Parameters

**toWrite** Overwritten with the instance of the mapped ActorType.

**eid** The key being stored.

Returns true if eid was found in the map. false if it was not found.

#### 5.2.2.3 bool RemoveActorMapping (const DIS::EntityType & eid)

Takes the one to one mapping from the container for the supplied key. Parameters

**eid** the key to the mapping.

Returns true if eid was found in the map. false if it was not found.

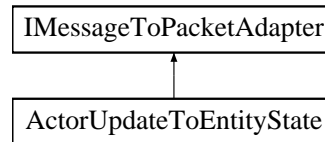
The documentation for this struct was generated from the following files:

- **sharedstate.h**
- **sharedstate.cpp**

## 5.3 ActorUpdateToEntityState Class Reference

responsible for translating ActorUpdateMessage instances to EntityStatePdu instances.

#include <inc/dtDIS/plugins/default/actorupdatetoentitystate.h>Inheritance diagram for ActorUpdateToEntityState::



### Public Member Functions

- **ActorUpdateToEntityState** (dtDIS::SharedState \*config, dtGame::GameManager \*gm)
- **~ActorUpdateToEntityState** ()
- DIS::Pdu \* **Convert** (const dtGame::Message &source)

*The interface for converting from a dtGame::Message to a **DIS** (p. 13) packet.*

#### 5.3.1 Detailed Description

responsible for translating ActorUpdateMessage instances to EntityStatePdu instances.

#### 5.3.2 Constructor & Destructor Documentation

**5.3.2.1 ActorUpdateToEntityState** (dtDIS::SharedState \* *config*, dtGame::GameManager \* *gm*)

**5.3.2.2 ~ActorUpdateToEntityState** ()

#### 5.3.3 Member Function Documentation

**5.3.3.1 DIS::Pdu \* Convert** (const dtGame::Message & *source*) [virtual]

The interface for converting from a dtGame::Message to a **DIS** (p. 13) packet. Parameters

**source** The data container

Returns the serialized packet instance

Implements **IMessageToPacketAdapter** (p. 39).

The documentation for this class was generated from the following files:

- **actorupdatetoentitystate.h**
- **actorupdatetoentitystate.cpp**

## 5.4 Connection Class Reference

Makes a multicast connection to support **DIS** (p. 13) networks.

```
#include <inc/dtDIS/connection.h>
```

### Public Member Functions

- void **Connect** (unsigned int port, const char \*host)  
*makes a socket connection for the specified network.*
- void **Disconnect** ()  
*closes the socket connection.*
- size\_t **Receive** (char \*buf, size\_t numbytes)  
*allocates buf with size numbytes.*
- void **Send** (const char \*buf, size\_t numbytes)  
*publishes the data to the network.*

#### 5.4.1 Detailed Description

Makes a multicast connection to support **DIS** (p. 13) networks. Note requires the HawkNL socket library.  
<http://www.hawksoft.com/hawknl/>

#### 5.4.2 Member Function Documentation

##### 5.4.2.1 void Connect (unsigned int *port*, const char \* *host*)

makes a socket connection for the specified network. Parameters

***port*** the serial port for the connection's network host.

***host*** the name of the network host.

##### 5.4.2.2 void Disconnect ()

closes the socket connection.

#### Todo

is this the ideal NL call?

##### 5.4.2.3 size\_t Receive (char \* *buf*, size\_t *numbytes*)

allocates buf with size numbytes. Parameters

***buf*** the buffer to be written to with network bytes

***numbytes*** the maximum index used for the buffer (buf)

Returns the number of bytes read from the connection

##### 5.4.2.4 void Send (const char \* *buf*, size\_t *numbytes*)

publishes the data to the network. Parameters

***buf*** the buffer to be written to with network bytes.

***numbytes*** the number of bytes contained in the the buffer.

The documentation for this class was generated from the following files:

- **connection.h**
- **connection.cpp**

## 5.5 ConnectionData Struct Reference

the information needed to connect to the **DIS** (p. 13) network.

```
#include <inc/dtDIS/sharedstate.h>
```

### Public Attributes

- unsigned short **application\_id**  
*the ID of the sending application*
- unsigned char **exercise\_id**  
*the ID for the local simulation client*
- std::string **ip**
- unsigned int **MTU**
- std::string **plug\_dir**
- unsigned int **port**
- unsigned short **site\_id**  
*The ID of the sending site.*

### 5.5.1 Detailed Description

the information needed to connect to the **DIS** (p. 13) network.

### 5.5.2 Member Data Documentation

#### 5.5.2.1 unsigned short application\_id

the ID of the sending application

#### 5.5.2.2 unsigned char exercise\_id

the ID for the local simulation client

#### 5.5.2.3 std::string ip

#### 5.5.2.4 unsigned int MTU

#### 5.5.2.5 std::string plug\_dir

#### 5.5.2.6 unsigned int port

#### 5.5.2.7 unsigned short site\_id

The ID of the sending site.

The documentation for this struct was generated from the following file:

- **sharedstate.h**

## 5.6 ConnectionXMLHandler Class Reference

Reads the connection info from a config file.

```
#include <inc/dtDIS/disxml.h>
```

### Public Member Functions

- **ConnectionXMLHandler** ()
- **~ConnectionXMLHandler** ()
- void **characters** (const XMLCh \*const chars, const unsigned int length)
- void **endDocument** ()
- void **endElement** (const XMLCh \*const uri, const XMLCh \*const localname, const XMLCh \*const qname)
- void **endPrefixMapping** (const XMLCh \*const prefix)
- const **dtDIS::ConnectionData & GetConnectionData** () const
- void **ignorableWhitespace** (const XMLCh \*const chars, const unsigned int length)
- void **processingInstruction** (const XMLCh \*const target, const XMLCh \*const data)
- void **setDocumentLocator** (const XERCES\_CPP\_NAMESPACE\_QUALIFIER Locator \*const locator)
- void **skippedEntity** (const XMLCh \*const name)
- void **startDocument** ()
- void **startElement** (const XMLCh \*const uri, const XMLCh \*const localname, const XMLCh \*const qname, const XERCES\_CPP\_NAMESPACE\_QUALIFIER Attributes &attrs)
- void **startPrefixMapping** (const XMLCh \*const prefix, const XMLCh \*const uri)

### Static Public Attributes

- static const **dtDIS::ConnectionData DEFAULT\_CONNECTION\_DATA**

#### 5.6.1 Detailed Description

Reads the connection info from a config file.

## 5.6.2 Constructor & Destructor Documentation

5.6.2.1 ConnectionXMLHandler ()

5.6.2.2 ~ConnectionXMLHandler ()

## 5.6.3 Member Function Documentation

5.6.3.1 void characters (const XMLCh \*const *chars*, const unsigned int *length*)

5.6.3.2 void endDocument ()

5.6.3.3 void endElement (const XMLCh \*const *uri*, const XMLCh \*const *localname*, const XMLCh \*const *qname*)

5.6.3.4 void endPrefixMapping (const XMLCh \*const *prefix*)

5.6.3.5 const dtDIS::ConnectionData & GetConnectionData () const

5.6.3.6 void ignorableWhitespace (const XMLCh \*const *chars*, const unsigned int *length*)

5.6.3.7 void processingInstruction (const XMLCh \*const *target*, const XMLCh \*const *data*)

5.6.3.8 void setDocumentLocator (const XERCES\_CPP\_NAMESPACE\_QUALIFIER Locator \*const *locator*)

5.6.3.9 void skippedEntity (const XMLCh \*const *name*)

5.6.3.10 void startDocument ()

5.6.3.11 void startElement (const XMLCh \*const *uri*, const XMLCh \*const *localname*, const XMLCh \*const *qname*, const XERCES\_CPP\_NAMESPACE\_QUALIFIER Attributes & *attrs*)

5.6.3.12 void startPrefixMapping (const XMLCh \*const *prefix*, const XMLCh \*const *uri*)

## 5.6.4 Member Data Documentation

5.6.4.1 const dtDIS::ConnectionData DEFAULT\_CONNECTION\_DATA [static]

Initial value:

```
{ 62040,
    9.1.2.3",
    ../../dtDIS_trunk/bin/plugins",
    0 }
    "23
    ".
    1,
    150
```

The documentation for this class was generated from the following files:

- disxml.h
- disxml.cpp

## 5.7 CreateEntityProcessor Class Reference

the plugin's model of how to respond to the DIS::CreateEntityPdu message.

```
#include <inc/dtDIS/plugins/default/createentityprocessor.h>
```

### Public Member Functions

- **CreateEntityProcessor** (**OutgoingMessage** \*omsg, **SharedState** \*config)
- **~CreateEntityProcessor** ()
- void **Process** (const DIS::Pdu &p)

#### 5.7.1 Detailed Description

the plugin's model of how to respond to the DIS::CreateEntityPdu message.

#### 5.7.2 Constructor & Destructor Documentation

**5.7.2.1 CreateEntityProcessor (OutgoingMessage \* omsg, SharedState \* config)**

**5.7.2.2 ~CreateEntityProcessor ()**

#### 5.7.3 Member Function Documentation

**5.7.3.1 void Process (const DIS::Pdu & p)**

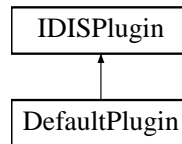
The documentation for this class was generated from the following files:

- **createentityprocessor.h**
- **createentityprocessor.cpp**

## 5.8 DefaultPlugin Class Reference

the plugin to support various DIS::Pdu types.

#include <inc/dtDIS/plugins/default/defaultplugin.h> Inheritance diagram for DefaultPlugin::



### Public Member Functions

- **DefaultPlugin** ()
- **~DefaultPlugin** ()
- void **Finish** (DIS::IncomingMessage &imsg, dtDIS::OutgoingMessage &omsg)  
*plugin should forget the state of the game.*
- void **Start** (DIS::IncomingMessage &imsg, dtDIS::OutgoingMessage &omsg, dtGame::GameManager \*gm, dtDIS::SharedState \*config)  
*attaches the IPacketProcessor and IMessageToPacketAdapter (p. 39)*

### 5.8.1 Detailed Description

the plugin to support various DIS::Pdu types.

### 5.8.2 Constructor & Destructor Documentation

#### 5.8.2.1 DefaultPlugin ()

#### 5.8.2.2 ~DefaultPlugin ()

### 5.8.3 Member Function Documentation

#### 5.8.3.1 void Finish (DIS::IncomingMessage & *incoming*, dtDIS::OutgoingMessage & *outgoing*) [virtual]

plugin should forget the state of the game. called before unloading.

Implements **IDISPlugin** (p. 38).

#### 5.8.3.2 void Start (DIS::IncomingMessage & *imsg*, dtDIS::OutgoingMessage & *omsg*, dtGame::GameManager \* *gm*, dtDIS::SharedState \* *config*) [virtual]

attaches the IPacketProcessor and **IMessageToPacketAdapter** (p. 39)

Implements **IDISPlugin** (p. 38).

The documentation for this class was generated from the following files:

- **defaultplugin.h**
- **defaultplugin.cpp**

## 5.9 EnginePropertyName Struct Reference

```
#include <inc/dtDIS/propertyname.h>
```

### Static Public Attributes

- static dtUtil::RefString **ARTICULATION**
- static dtUtil::RefString **DEAD\_RECKONING\_ALGORITHM**
- static dtUtil::RefString **DOF\_NODE\_NAME**  
*this value is not a typo, it needs to be this because of the DVTE support.*
- static dtUtil::RefString **ENTITY\_LINEARY\_VELOCITY**
- static dtUtil::RefString **ENTITY\_LOCATION**  
*Outgoing DIS (p. 13) packets read from this Property for the entity's XYZ translation.*
- static dtUtil::RefString **ENTITY\_MARKING**
- static dtUtil::RefString **ENTITY\_ORIENTATION**  
*Outgoing DIS (p. 13) packets read from this Property for the entity's XYZ axis rotation (not HPR).*
- static dtUtil::RefString **GROUND\_CLAMP**
- static dtUtil::RefString **LAST\_KNOWN\_LOCATION**  
*Incoming DIS (p. 13) packets apply entity XYZ translation to this Property.*
- static dtUtil::RefString **LAST\_KNOWN\_ORIENTATION**  
*Incoming DIS (p. 13) packets apply entity orientation to this Property.*
- static dtUtil::RefString **RESOURCE\_DAMAGE\_DESTROYED**
- static dtUtil::RefString **RESOURCE\_DAMAGE\_OFF**
- static dtUtil::RefString **RESOURCE\_DAMAGE\_ON**

### 5.9.1 Member Data Documentation

#### 5.9.1.1 dtUtil::RefString ARTICULATION [static]

##### Todo

this only uses the substring 'Parameter' because dvte::IG::Entity needs it.

#### 5.9.1.2 dtUtil::RefString DEAD\_RECKONING\_ALGORITHM [static]

#### 5.9.1.3 dtUtil::RefString DOF\_NODE\_NAME [static]

this value is not a typo, it needs to be this because of the DVTE support.

#### 5.9.1.4 dtUtil::RefString ENTITY\_LINEARY\_VELOCITY [static]

#### 5.9.1.5 dtUtil::RefString ENTITY\_LOCATION [static]

Outgoing DIS (p. 13) packets read from this Property for the entity's XYZ translation.

#### 5.9.1.6 dtUtil::RefString ENTITY\_MARKING [static]

#### 5.9.1.7 dtUtil::RefString ENTITY\_ORIENTATION [static]

Outgoing DIS (p. 13) packets read from this Property for the entity's XYZ axis rotation (not HPR).

#### 5.9.1.8 dtUtil::RefString GROUND\_CLAMP [static]

##### Todo

this only named "Flying" for now because of the dvte::IG::Entity actor.

**5.9.1.9 dtUtil::RefString LAST\_KNOWN\_LOCATION [static]**

Incoming **DIS** (p.13) packets apply entity XYZ translation to this Property. This is used by the dtGame::DeadReckoningComponent to update the Delta3D actor's position. If no DeadReckoningComponent is used, set this variable to be the same as ENTITY\_LOCATION.

**5.9.1.10 dtUtil::RefString LAST\_KNOWN\_ORIENTATION [static]**

Incoming **DIS** (p.13) packets apply entity orientation to this Property. This is used by the dtGame::DeadReckoningComponent to update the Delta3D actor's rotation. If no DeadReckoningComponent is used, set this variable to be the same as ENTITY\_ORIENTATION.

**5.9.1.11 dtUtil::RefString RESOURCE\_DAMAGE\_DESTROYED [static]****5.9.1.12 dtUtil::RefString RESOURCE\_DAMAGE\_OFF [static]****5.9.1.13 dtUtil::RefString RESOURCE\_DAMAGE\_ON [static]**

The documentation for this struct was generated from the following files:

- **propertyname.h**
- **propertyname.cpp**

## 5.10 EntityMapXMLHandler Class Reference

Stores configuration data loaded from a file for the purposes of mapping **DIS** (p. 13) to Delta3D.

```
#include <inc/dtDIS/disxml.h>
```

### Public Member Functions

- **EntityMapXMLHandler** (**SharedState** \*config)
- **~EntityMapXMLHandler** ()
- void **characters** (const XMLCh \*const chars, const unsigned int length)
- void **endDocument** ()
- void **endElement** (const XMLCh \*const uri, const XMLCh \*const localname, const XMLCh \*const qname)
- void **endPrefixMapping** (const XMLCh \*const prefix)
- void **ignorableWhitespace** (const XMLCh \*const chars, const unsigned int length)
- void **processingInstruction** (const XMLCh \*const target, const XMLCh \*const data)
- void **setDocumentLocator** (const XERCES\_CPP\_NAMESPACE\_QUALIFIER Locator \*const locator)
- void **skippedEntity** (const XMLCh \*const name)
- void **startDocument** ()
- void **startElement** (const XMLCh \*const uri, const XMLCh \*const localname, const XMLCh \*const qname, const XERCES\_CPP\_NAMESPACE\_QUALIFIER Attributes &attrs)
- void **startPrefixMapping** (const XMLCh \*const prefix, const XMLCh \*const uri)

### 5.10.1 Detailed Description

Stores configuration data loaded from a file for the purposes of mapping **DIS** (p. 13) to Delta3D. Populates the content within the EntityMapControl. Assumes the dtGame::GameManager has all ActorTypes registered already.

### 5.10.2 Constructor & Destructor Documentation

#### 5.10.2.1 EntityMapXMLHandler (SharedState \* config)

#### 5.10.2.2 ~EntityMapXMLHandler ()

### 5.10.3 Member Function Documentation

#### 5.10.3.1 void characters (const XMLCh \*const chars, const unsigned int length)

#### 5.10.3.2 void endDocument ()

#### 5.10.3.3 void endElement (const XMLCh \*const uri, const XMLCh \*const localname, const XMLCh \*const qname)

#### 5.10.3.4 void endPrefixMapping (const XMLCh \*const prefix)

#### 5.10.3.5 void ignorableWhitespace (const XMLCh \*const chars, const unsigned int length)

#### 5.10.3.6 void processingInstruction (const XMLCh \*const target, const XMLCh \*const data)

#### 5.10.3.7 void setDocumentLocator (const XERCES\_CPP\_NAMESPACE\_QUALIFIER Locator \*const locator)

#### 5.10.3.8 void skippedEntity (const XMLCh \*const name)

#### 5.10.3.9 void startDocument ()

#### 5.10.3.10 void startElement (const XMLCh \*const uri, const XMLCh \*const localname, const XMLCh \*const qname, const XERCES\_CPP\_NAMESPACE\_QUALIFIER Attributes & attrs)

#### 5.10.3.11 void startPrefixMapping (const XMLCh \*const prefix, const XMLCh \*const uri)

The documentation for this class was generated from the following files:

- disxml.h
- disxml.cpp

## 5.11 EntityPropertyName Struct Reference

```
#include <inc/dtDIS/propertyname.h>
```

### Static Public Attributes

- static dtUtil::RefString **APPEARANCE**  
*Contains the official ActorProperty names for DIS::EntityStatePdu attributes.*
- static dtUtil::RefString **ENTITY\_TYPE**

### 5.11.1 Member Data Documentation

#### 5.11.1.1 dtUtil::RefString APPEARANCE [static]

Contains the official ActorProperty names for DIS::EntityStatePdu attributes.

#### 5.11.1.2 dtUtil::RefString ENTITY\_TYPE [static]

The documentation for this struct was generated from the following files:

- **propertyname.h**
- **propertyname.cpp**

## 5.12 ESPduProcessor Class Reference

the interface class responsible for applying data from a DIS::EntityStatePdu packet.

```
#include <inc/dtDIS/plugins/default/espduprocessor.h>
```

### Public Member Functions

- **ESPduProcessor** (dtGame::GameManager \*gm, **SharedState** \*config)
- **~ESPduProcessor** ()
- void **Process** (const DIS::Pdu &p)

### Protected Member Functions

- void **AddActor** (const DIS::EntityStatePdu &pdu, dtDAL::ActorProxy \*proxy)
- void **ApplyFullUpdateToProxy** (const DIS::EntityStatePdu &pdu, dtGame::GameActorProxy &proxy)
- void **SendPartialUpdate** (const DIS::EntityStatePdu &pdu, const dtDAL::ActorProxy &actor)

#### 5.12.1 Detailed Description

the interface class responsible for applying data from a DIS::EntityStatePdu packet.

#### 5.12.2 Constructor & Destructor Documentation

5.12.2.1 **ESPduProcessor** (dtGame::GameManager \* *gm*, SharedState \* *config*)

5.12.2.2 **~ESPduProcessor** ()

#### 5.12.3 Member Function Documentation

5.12.3.1 void **AddActor** (const DIS::EntityStatePdu & *pdu*, dtDAL::ActorProxy \* *proxy*) [protected]

5.12.3.2 void **ApplyFullUpdateToProxy** (const DIS::EntityStatePdu & *pdu*, dtGame::GameActorProxy & *proxy*) [protected]

5.12.3.3 void **Process** (const DIS::Pdu & *p*)

5.12.3.4 void **SendPartialUpdate** (const DIS::EntityStatePdu & *pdu*, const dtDAL::ActorProxy & *actor*) [protected]

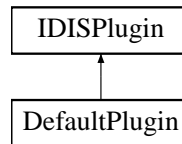
The documentation for this class was generated from the following files:

- **espduprocessor.h**
- **espduprocessor.cpp**

## 5.13 IDISPlugin Class Reference

the interface for all **DIS** (p. 13) plugins

#include <inc/dtDIS/idisplugin.h>Inheritance diagram for IDISPlugin::



### Public Member Functions

- virtual `~IDISPlugin ()`
- virtual void **Finish** (`DIS::IncomingMessage &incoming`, `dtDIS::OutgoingMessage &outgoing`)=0  
*plugin should forget the state of the game.*
- virtual void **Start** (`DIS::IncomingMessage &incoming`, `dtDIS::OutgoingMessage &outgoing`, `dtGame::GameManager *gm`, `dtDIS::SharedState *config`)=0  
*plugin is informed of the state of the game.*

#### 5.13.1 Detailed Description

the interface for all **DIS** (p. 13) plugins

#### 5.13.2 Constructor & Destructor Documentation

##### 5.13.2.1 `~IDISPlugin ()` [virtual]

#### 5.13.3 Member Function Documentation

##### 5.13.3.1 virtual void **Finish** (`DIS::IncomingMessage & incoming`, `dtDIS::OutgoingMessage & outgoing`) [pure virtual]

plugin should forget the state of the game. called before unloading.

Implemented in **DefaultPlugin** (p. 32).

##### 5.13.3.2 virtual void **Start** (`DIS::IncomingMessage & incoming`, `dtDIS::OutgoingMessage & outgoing`, `dtGame::GameManager * gm`, `dtDIS::SharedState * config`) [pure virtual]

plugin is informed of the state of the game. called after being successfully loaded. Parameters

**incoming** used to attach packet handlers.

**outgoing** used to attach message handlers.

**gm** used to deliver the state of the game.

**config** used to share connection and setup data.

Implemented in **DefaultPlugin** (p. 32).

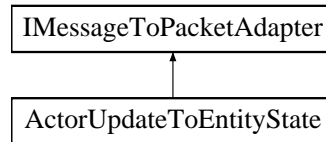
The documentation for this class was generated from the following files:

- **idisplugin.h**
- **idisplugin.cpp**

## 5.14 IMessageToPacketAdapter Class Reference

the interface for translating Delta3D messages into **DIS** (p. 13) packets.

#include <inc/dtDIS/imessagetopacketadapter.h> Inheritance diagram for IMessageToPacketAdapter::



### Public Member Functions

- virtual `~IMessageToPacketAdapter ()`
- virtual `DIS::Pdu * Convert (const dtGame::Message &source)=0`  
*The interface for converting from a dtGame::Message to a **DIS** (p. 13) packet.*

#### 5.14.1 Detailed Description

the interface for translating Delta3D messages into **DIS** (p. 13) packets.

#### 5.14.2 Constructor & Destructor Documentation

##### 5.14.2.1 `~IMessageToPacketAdapter ()` [virtual]

#### 5.14.3 Member Function Documentation

##### 5.14.3.1 `virtual DIS::Pdu* Convert (const dtGame::Message & source)` [pure virtual]

The interface for converting from a dtGame::Message to a **DIS** (p. 13) packet. Parameters

**source** The data container

Returns the serialized packet instance

Implemented in **ActorUpdateToEntityState** (p. 26).

The documentation for this class was generated from the following files:

- **imessagetopacketadapter.h**
- **imessagetopacketadapter.cpp**

## 5.15 MasterComponent Class Reference

Supports a framework to translate **DIS** (p. 13) PDUs into dtGame::Message instances.

```
#include <inc/dtDIS/mastercomponent.h>
```

### Public Member Functions

- **MasterComponent (SharedState \*config)**  
*supply the configuration files needed to support **DIS** (p. 13).*
- virtual void **DispatchNetworkMessage** (const dtGame::Message &message)  
*Time to convert the Message into **DIS** (p. 13) PDU's and send out.*
- const DIS::IncomingMessage & **GetIncomingMessage** () const  
*obtain the IncomingMessage for attaching and removing processors.*
- DIS::IncomingMessage & **GetIncomingMessage** ()  
*obtain the IncomingMessage for attaching and removing processors.*
- const **OutgoingMessage** & **GetOutgoingMessage** () const  
*obtain the **OutgoingMessage** (p. 44) for attaching and removing adapters.*
- **OutgoingMessage** & **GetOutgoingMessage** ()  
*obtain the **OutgoingMessage** (p. 44) for attaching and removing adapters.*
- const **SharedState** \* **GetSharedState** () const  
*obtain the **SharedState** (p. 50) to know the current state of entity management and configurations.*
- **SharedState** \* **GetSharedState** ()  
*obtain the **SharedState** (p. 50) to know the current state of entity management and configurations.*
- void **OnAddedToGM** ()  
*connects to the socket.*
- void **OnRemovedFromGM** ()  
*disconnects from the socket.*
- void **ProcessMessage** (const dtGame::Message &msg)  
*executed by the GameManager for observing any Message passing.*

### Protected Member Functions

- **~MasterComponent** ()
- void **LoadPlugins** (const std::string &directory)
- void **OnPluginLoaded** (PluginManager::LibraryRegistry::value\_type &entry)  
*a slot to be executed when loading each plugin.*
- void **OnPluginUnloaded** (PluginManager::LibraryRegistry::value\_type &plugin)  
*a slot to be executed when unloading each plugin.*
- void **UnloadPlugins** ()

### 5.15.1 Detailed Description

Supports a framework to translate **DIS** (p. 13) PDUs into dtGame::Message instances. The component will connect to a **DIS** (p. 13) network and load plugins to handle different **DIS** (p. 13) packet types. All you need to do is add this component to the dtGame::GameManager instance, and it will handle everything dealing with **DIS** (p. 13) for you, which includes:

- connecting to the **DIS** (p. 13) network, as specified in the connection file
- loading **DIS** (p. 13) packet plugins, also found in the connection file
- mapping DIS::EntityType instances to the ActorType instances specified in the mapping file.

### 5.15.2 Constructor & Destructor Documentation

#### 5.15.2.1 MasterComponent (SharedState \* config)

supply the configuration files needed to support **DIS** (p. 13). Parameters

**config** the result of reading data files needed for this component to work. This class does not assume ownership of the memory.

**connection\_file** The XML file that shows the **ConnectionData** (p. 28).

**mapping\_file** The XML file that shows the mappings from DIS::EntityIDs to dtDAL::ActorTypes.

#### Todo

what should set the network stream's endian type? the SharedState's connection data?

#### 5.15.2.2 ~MasterComponent () [protected]

### 5.15.3 Member Function Documentation

#### 5.15.3.1 void DispatchNetworkMessage (const dtGame::Message & message) [virtual]

Time to convert the Message into **DIS** (p. 13) PDU's and send out.

#### 5.15.3.2 const DIS::IncomingMessage & GetIncomingMessage () const

obtain the IncomingMessage for attaching and removing processors. Returns the IncomingMessage instance.

#### 5.15.3.3 DIS::IncomingMessage & GetIncomingMessage ()

obtain the IncomingMessage for attaching and removing processors. Returns the IncomingMessage instance.

#### 5.15.3.4 const OutgoingMessage & GetOutgoingMessage () const

obtain the **OutgoingMessage** (p. 44) for attaching and removing adapters. Returns the **OutgoingMessage** (p. 44) instance.

#### 5.15.3.5 OutgoingMessage & GetOutgoingMessage ()

obtain the **OutgoingMessage** (p. 44) for attaching and removing adapters. Returns the **OutgoingMessage** (p. 44) instance.

#### 5.15.3.6 const SharedState \* GetSharedState () const

obtain the **SharedState** (p. 50) to know the current state of entity management and configurations. Returns the **SharedState** (p. 50) instance.

#### 5.15.3.7 SharedState \* GetSharedState ()

obtain the **SharedState** (p. 50) to know the current state of entity management and configurations. Returns the **SharedState** (p. 50) instance.

**5.15.3.8 void LoadPlugins (const std::string & *directory*) [protected]****5.15.3.9 void OnAddedToGM ()**

connects to the socket. executed by the GameManager after being "added" to the GameManager. Note This can be ignored by client developers.

**5.15.3.10 void OnPluginLoaded (PluginManager::LibraryRegistry::value\_type & *entry*) [protected]**

a slot to be executed when loading each plugin. Note This can be ignored by client developers.

**5.15.3.11 void OnPluginUnloaded (PluginManager::LibraryRegistry::value\_type & *plugin*) [protected]**

a slot to be executed when unloading each plugin. Note This can be ignored by client developers.

**5.15.3.12 void OnRemovedFromGM ()**

disconnects from the socket. executed by the GameManager after being "removed" from the GameManager. Note This can be ignored by client developers.

**Todo**

remove the controlled entities.

**5.15.3.13 void ProcessMessage (const dtGame::Message & *msg*)**

executed by the GameManager for observing any Message passing. Note This can be ignored by client developers.

**Todo**

should it handle a pause message, by not updating the incoming or outgoing network classes?

**5.15.3.14 void UnloadPlugins () [protected]**

The documentation for this class was generated from the following files:

- **mastercomponent.h**
- **mastercomponent.cpp**

## 5.16 NodeName Struct Reference

a static definition of the scene graph node names to be used for articulated parts.

```
#include <inc/dtDIS/articulationconstants.h>
```

### Static Public Attributes

- static const char **NODE\_PRIMARY\_GUN** [] = { "dof\_gun\_01\0" }
- static const char **NODE\_PRIMARY\_TURRET** [] = { "dof\_turret\_01\0" }
- static const char **NODE\_SECONDARY\_GUN** [] = { "dof\_gun\_02\0" }

### 5.16.1 Detailed Description

a static definition of the scene graph node names to be used for articulated parts.

### 5.16.2 Member Data Documentation

**5.16.2.1** const char **NODE\_PRIMARY\_GUN** = { "dof\_gun\_01\0" } [static]

**5.16.2.2** const char **NODE\_PRIMARY\_TURRET** = { "dof\_turret\_01\0" } [static]

**5.16.2.3** const char **NODE\_SECONDARY\_GUN** = { "dof\_gun\_02\0" } [static]

The documentation for this struct was generated from the following files:

- **articulationconstants.h**
- **articulationconstants.cpp**

## 5.17 OutgoingMessage Class Reference

A framework for translating dtGame::Message instances to PDUs.

```
#include <inc/dtDIS/outgoingmessage.h>
```

### Public Types

- typedef std::multimap< const dtGame::MessageType \*, IMessageToPacketAdapter \* > **AdapterMap**  
*the container type for message adapters.*

### Public Member Functions

- **OutgoingMessage** (DIS::Endian stream, unsigned char exercise)  
*setup the network support.*
- void **AddAdaptor** (const dtGame::MessageType \*mt, IMessageToPacketAdapter \*adapter)
- void **ClearData** ()
- const AdapterMap & **GetAdapters** ()
- const DIS::DataStream & **GetData** () const
- void **Handle** (const DIS::Pdu &pdu)
- void **Handle** (const dtGame::Message &msg)
- void **RemoveAdaptor** (const dtGame::MessageType \*mt, IMessageToPacketAdapter \*adapter)

#### 5.17.1 Detailed Description

A framework for translating dtGame::Message instances to PDUs. Register adapters to handle specific dtGame::MessageTypes.

#### 5.17.2 Member Typedef Documentation

**5.17.2.1** typedef std::multimap<const dtGame::MessageType\*,IMessageToPacketAdapter\*>  
**AdapterMap**

the container type for message adapters.

#### 5.17.3 Constructor & Destructor Documentation

**5.17.3.1** **OutgoingMessage** (DIS::Endian *stream*, unsigned char *exercise*)

setup the network support. Parameters

**stream** the endian type to be used for the network stream.

**exercise** the DIS (p. 13) exercise identifier for this simulator.

#### 5.17.4 Member Function Documentation

**5.17.4.1** void **AddAdaptor** (const dtGame::MessageType \* *mt*, IMessageToPacketAdapter \* *adapter*)

**5.17.4.2** void **ClearData** ()

**5.17.4.3** const AdapterMap& **GetAdapters** () [inline]

**5.17.4.4** const DIS::DataStream & **GetData** () const

**5.17.4.5** void **Handle** (const DIS::Pdu & *pdu*)

**5.17.4.6** void **Handle** (const dtGame::Message & *msg*)

#### Todo

use the rest of the adapters in the multimap container.

**5.17.4.7 void RemoveAdaptor (const dtGame::MessageType \* *mt*, IMessageToPacketAdapter \* *adapter*)**

The documentation for this class was generated from the following files:

- **outgoingmessage.h**
- **outgoingmessage.cpp**

## 5.18 PluginManager Class Reference

manages a container of available plugins.

```
#include <inc/dtDIS/pluginmanager.h>
```

### Public Types

- typedef dtDIS::details::CreateDestroyPolicy< **IDISPlugin** > **LibLoaderT**  
*convenience typedef.*
- typedef std::map< std::string, **RegistryEntry** > **LibraryRegistry**  
*a mapping from the library path loaded to an object with defined entry points.*
- typedef sigslot::signal2< const std::string &, **RegistryEntry** & > **PluginSignal**  
*specifies the required function signature for slots observing plugin events.*
- typedef LibLoaderT::LibraryInterface **RegistryEntry**  
*the stored type, also passed to observers upon notification.*

### Public Member Functions

- **PluginSignal** & **GetLoadedSignal** ()  
*get the signal instance in order to connect or disconnct to the loaded event.*
- **LibraryRegistry** & **GetRegistry** ()  
*get the plugin container*
- const **LibraryRegistry** & **GetRegistry** () const  
*get the plugin container*
- **PluginSignal** & **GetUnloadedSignal** ()  
*get the signal instance in order to connect or disconnct to the unloaded event.*
- bool **LoadPlugin** (const std::string &path)  
*searches for the file, notifies observers of the new plugin.*
- void **UnloadAllPlugins** ()  
*will call this->UnloadPlugin for all of the known plugins.*
- bool **UnloadPlugin** (const std::string &path)  
*notifies observers that the memory will be released if the path is a loaded library, releases the memory for library.*

#### 5.18.1 Detailed Description

manages a container of available plugins. notifies observers when plugins are added or removed to the container.

#### 5.18.2 Member Typedef Documentation

**5.18.2.1** typedef dtDIS::details::CreateDestroyPolicy<IDISPlugin> LibLoaderT  
convenience typedef.

**5.18.2.2** typedef std::map<std::string,RegistryEntry> LibraryRegistry  
a mapping from the library path loaded to an object with defined entry points.

**5.18.2.3 typedef sigslot::signal2<const std::string&, RegistryEntry&> PluginSignal**

specifies the required function signature for slots observing plugin events.

**5.18.2.4 typedef LibLoaderT::LibraryInterface RegistryEntry**

the stored type, also passed to observers upon notification.

**5.18.3 Member Function Documentation****5.18.3.1 PluginManager::PluginSignal & GetLoadedSignal ()**

get the signal instance in order to connect or disconnect to the loaded event.

**5.18.3.2 PluginManager::LibraryRegistry & GetRegistry ()**

get the plugin container

**5.18.3.3 const PluginManager::LibraryRegistry & GetRegistry () const**

get the plugin container

**5.18.3.4 PluginManager::PluginSignal & GetUnloadedSignal ()**

get the signal instance in order to connect or disconnect to the unloaded event.

**5.18.3.5 bool LoadPlugin (const std::string & *path*)**

searches for the file, notifies observers of the new plugin. Parameters

***path*** the file path for the library to be loaded.

Returns true if the plugin was successfully added to the registry

**5.18.3.6 void UnloadAllPlugins ()**

will call this->UnloadPlugin for all of the known plugins.

**5.18.3.7 bool UnloadPlugin (const std::string & *path*)**

notifies observers that the memory will be released if the path is a loaded library, releases the memory for library. Parameters

***path*** the file path for the library to be loaded.

Returns true if the plugin path was found in the registry.

The documentation for this class was generated from the following files:

- **pluginmanager.h**
- **pluginmanager.cpp**

## 5.19 RemoveEntityProcessor Class Reference

the plugin's model of how to respond to the DIS::RemoveEntityPdu message.

```
#include <inc/dtDIS/plugins/default/removeentityprocessor.h>
```

### Public Member Functions

- **RemoveEntityProcessor** (**OutgoingMessage** \*omsg, **SharedState** \*config)
- **~RemoveEntityProcessor** ()
- void **Process** (const DIS::Pdu &p)

#### 5.19.1 Detailed Description

the plugin's model of how to respond to the DIS::RemoveEntityPdu message.

#### 5.19.2 Constructor & Destructor Documentation

**5.19.2.1 RemoveEntityProcessor (OutgoingMessage \* omsg, SharedState \* config)**

**5.19.2.2 ~RemoveEntityProcessor ()**

#### 5.19.3 Member Function Documentation

**5.19.3.1 void Process (const DIS::Pdu & p)**

The documentation for this class was generated from the following files:

- **removeentityprocessor.h**
- **removeentityprocessor.cpp**

## 5.20 ResourceMapConfig Struct Reference

a structure to maintain the relationship between the DIS::EntityID and the resources available.

```
#include <inc/dtDIS/sharedstate.h>
```

### Public Member Functions

- bool **AddResourceMapping** (const DIS::EntityType &eid, const dtDAL::ResourceDescriptor &resource)  
*Adds a mapping from the DIS::EntityID to the resource identifier if no mapping exists.*
- bool **GetMappedResource** (const DIS::EntityType &eid, const dtDAL::ResourceDescriptor \*&toWrite) const  
*Introduces the ActorType mapped to the EntityID.*
- bool **RemoveResourceMapping** (const DIS::EntityType &eid)  
*Takes the one to one mapping from the container for the supplied key.*

### 5.20.1 Detailed Description

a structure to maintain the relationship between the DIS::EntityID and the resources available.

### 5.20.2 Member Function Documentation

#### 5.20.2.1 bool AddResourceMapping (const DIS::EntityType & eid, const dtDAL::ResourceDescriptor & resource)

Adds a mapping from the DIS::EntityID to the resource identifier if no mapping exists. Returns true if no previous mapping existed. false if a mapping existed already.

#### 5.20.2.2 bool GetMappedResource (const DIS::EntityType & eid, const dtDAL::ResourceDescriptor \*& toWrite) const

Introduces the ActorType mapped to the EntityID. Parameters

**toWrite** Overwritten with the instance of the mapped resource identifier.

**eid** The key being stored.

Returns true if eid was found in the map. false if it was not found.

#### 5.20.2.3 bool RemoveResourceMapping (const DIS::EntityType & eid)

Takes the one to one mapping from the container for the supplied key. Parameters

**eid** the key to the mapping.

Returns true if eid was found and removed from the map. false if it was not removed.

The documentation for this struct was generated from the following files:

- **sharedstate.h**
- **sharedstate.cpp**

## 5.21 SharedState Class Reference

The data to be shared among plugins.

```
#include <inc/dtDIS/sharedstate.h>
```

### Public Member Functions

- **SharedState** (const std::string &connectionXMLFile="", const std::string &entityMappingXMLFile="")
- **~SharedState** ()
- const **ActiveEntityControl** & **GetActiveEntityControl** () const
- **ActiveEntityControl** & **GetActiveEntityControl** ()
- const **ActorMapConfig** & **GetActorMap** () const
- **ActorMapConfig** & **GetActorMap** ()
- unsigned short **GetApplicationID** () const
- const **ConnectionData** & **GetConnectionData** () const
- dtUtil::Coordinates & **GetCoordinateConverter** ()
- const dtUtil::Coordinates & **GetCoordinateConverter** () const
- const **ResourceMapConfig** & **GetResourceMap** () const
- **ResourceMapConfig** & **GetResourceMap** ()
- unsigned short **GetSiteID** () const
- void **SetApplicationID** (unsigned short ID)
- void **SetConnectionData** (const **ConnectionData** &data)
- void **SetCoordinateConverter** (const dtUtil::Coordinates &coordConverter)
- void **SetSiteID** (unsigned short ID)

#### 5.21.1 Detailed Description

The data to be shared among plugins.

#### Todo

it would be good to have a file that maps the 'appearance' values to an asset handle, rather than full actor "types" just to change the asset.

## 5.21.2 Constructor & Destructor Documentation

5.21.2.1 SharedState (const std::string & *connectionXMLFile* = "", const std::string & *entityMappingXMLFile* = "")

5.21.2.2 ~SharedState ()

## 5.21.3 Member Function Documentation

5.21.3.1 const ActiveEntityControl & GetActiveEntityControl () const

5.21.3.2 ActiveEntityControl & GetActiveEntityControl ()

5.21.3.3 const ActorMapConfig & GetActorMap () const

5.21.3.4 ActorMapConfig & GetActorMap ()

5.21.3.5 unsigned short GetApplicationID () const

5.21.3.6 const ConnectionData & GetConnectionData () const

5.21.3.7 dtUtil::Coordinates & GetCoordinateConverter ()

5.21.3.8 const dtUtil::Coordinates & GetCoordinateConverter () const

5.21.3.9 const ResourceMapConfig & GetResourceMap () const

5.21.3.10 ResourceMapConfig & GetResourceMap ()

5.21.3.11 unsigned short GetSiteID () const

5.21.3.12 void SetApplicationID (unsigned short *ID*)

5.21.3.13 void SetConnectionData (const ConnectionData & *data*)

5.21.3.14 void SetCoordinateConverter (const dtUtil::Coordinates & *coordConverter*)

5.21.3.15 void SetSiteID (unsigned short *ID*)

The documentation for this class was generated from the following files:

- sharedstate.h
- sharedstate.cpp

## 5.22 ValueMap Struct Reference

a name scope for all sorts of mappings

```
#include <inc/dtDIS/valuemaps.h>
```

### Static Public Member Functions

- static bool **GetArticulationMotionPropertyName** (unsigned int *motiontype*, std::string &*propertyname*)  
*find the actor property name value for the supported motiontype associated with articulation.*
- static bool **GetArticulationNodeName** (unsigned int *parttype*, std::string &*nodename*)  
*find the supported node name for the part type identifier.*
- static bool **GetDeadReckoningModelPropertyValue** (unsigned char *drm*, std::string &*property*)  
*maps the provided dead reckoning model to the supported Delta3D property value.*
- static bool **GetRequiresGroundClamping** (const DIS::EntityType &*etype*, bool &*requires*)  
*informs clients if the EntityType should support ground clamping.*

#### 5.22.1 Detailed Description

a name scope for all sorts of mappings

#### 5.22.2 Member Function Documentation

##### 5.22.2.1 bool GetArticulationMotionPropertyName (unsigned int *motiontype*, std::string & *propertyname*) [static]

find the actor property name value for the supported motiontype associated with articulation. Parameters

***motiontype*** the DIS (p. 13) standard type of motion.

***propertyname*** the corresponding name of the property for the motiontype.

Returns true if the motiontype is supported

##### 5.22.2.2 bool GetArticulationNodeName (unsigned int *parttype*, std::string & *nodename*) [static]

find the supported node name for the part type identifier. Parameters

***parttype*** the part identifier in the DIS::ArticulationParameter

***nodename*** the string value of the scenegraph node to be searched when performing articulation.

Returns true if parttype is supported, false if it is not supported.

##### 5.22.2.3 bool GetDeadReckoningModelPropertyValue (unsigned char *drm*, std::string & *property*) [static]

maps the provided dead reckoning model to the supported Delta3D property value. Parameters

***drm*** the dead reckoning model.

***property*** the value to be assigned which is mapped to the drm.

Returns true if the value of drm is supported.

**5.22.2.4 bool GetRequiresGroundClamping (const DIS::EntityType & *etype*, bool & *requires*) [static]**

informs clients if the EntityType should support ground clamping. Parameters

***requires*** modified to true when the EntityType should be ground clamped.

***etype*** the EntityType interested to know if ground clamping is desired.

Returns true if there exists a known interpretation of the EntityType

The documentation for this struct was generated from the following files:

- **valuemaps.h**
- **valuemaps.cpp**

## 5.23 XMLFiles Struct Reference

constant definitions for the official schema files.

```
#include <inc/dtDIS/disxml.h>
```

### Static Public Attributes

- static const char **XML\_CONNECTION\_SCHEMA\_FILE** [] = {"dis\_connection.xsd\0"}
- static const char **XML\_MAPPING\_SCHEMA\_FILE** [] = {"dis\_mapping.xsd\0"}

#### 5.23.1 Detailed Description

constant definitions for the official schema files.

#### 5.23.2 Member Data Documentation

**5.23.2.1 XERCES\_CPP\_NAMESPACE\_USE** const char **XML\_CONNECTION\_SCHEMA\_FILE** = {"dis\_connection.xsd\0"} [static]

**5.23.2.2** const char **XML\_MAPPING\_SCHEMA\_FILE** = {"dis\_mapping.xsd\0"} [static]

The documentation for this struct was generated from the following files:

- **disxml.h**
- **disxml.cpp**

## File Documentation

---

### 6.1 acknowledgeconstants.h File Reference

#### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

#### Enumerations

- enum **AcknowledgeFlag** { **ACKNOWLEDGE\_CREATE\_ENTITY** = 1, **ACKNOWLEDGE\_REMOVE\_ENTITY** = 2, **ACKNOWLEDGE\_START\_RESUME** = 3, **ACKNOWLEDGE\_STOP\_FREEZE** = 4 }
- enum **ResponseFlag** { **RESPONSE\_OTHER** = 0, **RESPONSE\_ABLE\_TO\_COMPLY** = 1, **RESPONSE\_UNABLE\_TO\_COMPLY** = 2 }

## 6.2 activeentitycontrol.cpp File Reference

```
#include <dtDIS/activeentitycontrol.h>  
#include <dtDAL/actorproxy.h>
```

## 6.3 activeentitycontrol.h File Reference

```
#include <dtDIS/dtdisexport.h>
#include <dtDIS/entityidcompare.h>
#include <DIS/EntityID.h>
#include <dtCore/uniqueid.h>
#include <dtCore/refptr.h>
#include <map>
```

### Classes

- struct **ActiveEntityControl**  
*provides a single point for associating known entities & actors.*

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.4 actorupdatetoentitystate.cpp File Reference

```
#include <dtDIS/plugins/default/actorupdatetoentitystate.h>
#include <dtDIS/plugins/default/espduapplicator.h>
#include <dtDIS/propertyname.h>
#include <DIS/EntityStatePdu.h>
#include <DIS/EntityType.h>
#include <dtDAL/enginepropertytypes.h>
#include <dtGame/actorupdatemessage.h>
#include <dtDIS/sharedstate.h>
#include <dtGame/gamemanager.h>
#include <dtUtil/stringutils.h>
```

## 6.5 actorupdatetoentitystate.h File Reference

```
#include <dtDIS/imessagetopacketadapter.h>
#include <dtCore/refptr.h>
#include <dtDIS/plugins/default/dtdisdefaultpluginexport.h>
```

### Classes

- class **ActorUpdateToEntityState**  
*responsible for translating ActorUpdateMessage instances to EntityStatePdu instances.*

### Namespaces

- namespace **DIS**
- namespace **dtCore**
- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*
- namespace **dtGame**

## 6.6 articulationconstants.cpp File Reference

```
#include <dtDIS/articulationconstants.h>
```

## 6.7 articulationconstants.h File Reference

```
#include <string>
```

```
#include <dtDIS/dtdisexport.h>
```

### Classes

- struct **NodeName**  
*a static definition of the scene graph node names to be used for articulated parts.*

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*
- namespace **dtDIS::Articulation**  
*the scope for articulation specific constants, lacking in the **DIS** (p. 13) dependency, and other tools.*

### Enumerations

- enum **MotionType** { **MOTION\_AZIMUTH** = 11, **MOTION\_AZIMUTH\_RATE** = 12, **MOTION\_ELEVATION** = 13 }
- enum **ParameterTypeDesignator** { **ARTICULATED\_PART** = 0, **ATTACHED\_PART** = 1 }
- enum **PartType** { **PART\_PRIMARY\_TURRET** = 4096, **PART\_PRIMARY\_GUN** = 4416, **PART\_SECONDARY\_GUN** = 6016 }

## 6.8 connection.cpp File Reference

```
#include <dtDIS/connection.h>  
#include <dtUtil/log.h>  
#include <sstream>
```

## 6.9 connection.h File Reference

```
#include <nl.h>
#include <cstddef>
#include <dtDIS/dtdisexport.h>
```

### Classes

- class **Connection**  
*Makes a multicast connection to support **DIS** (p. 13) networks.*

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.10 containerutils.h File Reference

### Namespaces

- namespace **dtUtil**

### Functions

- template<class MMTypе >  
MMType::iterator **find\_multimap\_pair** (MMType &mmap, const typename MMTypе::key\_type &key, const  
typename MMTypе::mapped\_type &value)  
*find the iterator for the unique key-value pair.*

## 6.11 createdestroypolicy.h File Reference

```
#include <dtDIS/libraryregistry.h>
#include <dtUtil/librarysharingmanager.h>
#include <dtUtil/log.h>
#include <cstdlib>
#include <dtDIS/createdestroypolicy.inl>
```

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.12 createdestroypolicy.ini File Reference

## 6.13 createentityprocessor.cpp File Reference

```
#include <dtDIS/plugins/default/createentityprocessor.h>
#include <dtDIS/outgoingmessage.h>
#include <dtDIS/sharedstate.h>
#include <DIS/CreateEntityPdu.h>
#include <DIS/AcknowledgePdu.h>
#include <dtDIS/hasproperty.h>
#include <dtDIS/plugins/default/espduapplicator.h>
#include <dtDIS/propertyname.h>
#include <dtDIS/acknowledgeconstants.h>
```

## 6.14 createentityprocessor.h File Reference

```
#include <DIS/IPacketProcessor.h>
```

```
#include <dtDIS/plugins/default/dtdisdefaultpluginexport.h>
```

### Classes

- class **CreateEntityProcessor**  
*the plugin's model of how to respond to the DIS::CreateEntityPdu message.*

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.15 defaultplugin.cpp File Reference

```
#include <dtDIS/plugins/default/defaultplugin.h>
#include <DIS/EntityStatePdu.h>
#include <DIS/IncomingMessage.h>
#include <dtDIS/outgoingmessage.h>
#include <dtGame/gamemanager.h>
#include <vector>
#include <dtCore/refptr.h>
#include <dtDAL/actortype.h>
#include <DIS/PDUType.h>
#include <cstdint>
#include <dtDIS/plugins/default/espduprocessor.h>
#include <dtDIS/plugins/default/createentityprocessor.h>
#include <dtDIS/plugins/default/removeentityprocessor.h>
#include <dtDIS/plugins/default/actorupdatetoentitystate.h>
```

## 6.16 defaultplugin.h File Reference

```
#include <dtDIS/idisplugin.h>
```

```
#include <dtDIS/plugins/default/dtdisdefaultpluginexport.h>
```

### Classes

- class **DefaultPlugin**  
*the plugin to support various DIS::Pdu types.*

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.17 disxml.cpp File Reference

```
#include <dtDIS/disxml.h>
#include <dtDIS/sharedstate.h>
#include <dtDIS/propertyname.h>
#include <dtUtil/log.h>
#include <dtUtil/stringutils.h>
#include <dtDAL/librarymanager.h>
#include <xercesc/util/XercesDefs.hpp>
#include <xercesc/util/XMLString.hpp>
```

## 6.18 disxml.h File Reference

```
#include <dtDIS/dtdisexport.h>
#include <string>
#include <xercesc/sax2/ContentHandler.hpp>
#include <xercesc/sax2/Attributes.hpp>
#include <stack>
#include <DIS/EntityType.h>
#include <dtCore/refptr.h>
#include <dtDIS/sharedstate.h>
```

### Classes

- class **ConnectionXMLHandler**  
*Reads the connection info from a config file.*
- class **EntityMapXMLHandler**  
*Stores configuration data loaded from a file for the purposes of mapping **DIS** (p. 13) to Delta3D.*
- struct **XMLFiles**  
*constant definitions for the official schema files.*

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.19 dllfinder.h File Reference

```
#include <osgDB/FileUtils>
#include <dtUtil/fileutils.h>
#include <osgDB/FileNameUtils>
```

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.20 dtdisdefaultpluginexport.h File Reference

### Defines

- #define DT\_DIS\_DEFAULT\_EXPORT

#### 6.20.1 Define Documentation

##### 6.20.1.1 #define DT\_DIS\_DEFAULT\_EXPORT

## 6.21 dtDisexport.h File Reference

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

### Defines

- #define **DT\_DIS\_EXPORT**

#### 6.21.1 Define Documentation

##### 6.21.1.1 #define DT\_DIS\_EXPORT

## 6.22 entityidcompare.cpp File Reference

```
#include <dtDIS/entityidcompare.h>
```

```
#include <DIS/EntityID.h>
```

## 6.23 entityidcompare.h File Reference

```
#include <dtDIS/dtdisexport.h>
```

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.24 entitytypeconstants.h File Reference

```
#include <dtDIS/dtdisexport.h>
```

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

### Enumerations

- enum **DomainType** {  
**DOMAIN\_LAND** = 1, **DOMAIN\_AIR** = 2, **DOMAIN\_SURFACE** = 3, **DOMAIN\_SUBSURFACE** = 4,  
**DOMAIN\_SPACE** = 5 }

## 6.25 espduapplicator.cpp File Reference

```
#include <dtDIS/plugins/default/espduapplicator.h>
#include <DIS/EntityStatePdu.h>
#include <DIS/Conversion.h>
#include <DIS/StreamUtils.h>
#include <DIS/Orientation.h>
#include <dtGame/actorupdatemessage.h>
#include <dtDIS/sharedstate.h>
#include <dtDIS/valuemaps.h>
#include <dtDIS/articulationconstants.h>
#include <dtDAL/datatype.h>
#include <dtUtil/stringutils.h>
#include <dtDAL/resourcedescriptor.h>
#include <dtDIS/propertyname.h>
#include <dtDAL/actorproxy.h>
#include <dtDAL/namedparameter.h>
#include <dtGame/deadreckoningcomponent.h>
#include <dtUtil/coordinates.h>
#include <sstream>
```

## 6.26 espduapplicator.h File Reference

```
#include <string>
#include <dtDIS/plugins/default/dtdisdefaultpluginexport.h>
#include <osg/Vec3>
```

### Namespaces

- namespace **dtDAL**
- namespace **dtDIS**
  - the Delta3D support framework for the **DIS** (p. 13) network protocol.*
- namespace **dtGame**

## 6.27 esduprocessor.cpp File Reference

```
#include <dtDIS/plugins/default/esduprocessor.h>
#include <dtDIS/plugins/default/espduapplicator.h>
#include <dtDIS/sharedstate.h>
#include <dtGame/gamemanager.h>
#include <dtGame/actorupdatemessage.h>
#include <DIS/EntityStatePdu.h>
```

## 6.28 espduprocessor.h File Reference

```
#include <DIS/IPacketProcessor.h>
#include <DIS/EntityID.h>
#include <map>
#include <dtCore/observerptr.h>
#include <dtDIS/entityidcompare.h>
#include <dtGame/messagefactory.h>
#include <dtDIS/plugins/default/dtdisdefaultpluginexport.h>
```

### Classes

- class **ESPduProcessor**  
*the interface class responsible for applying data from a DIS::EntityStatePdu packet.*

### Namespaces

- namespace **dtDAL**
- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*
- namespace **dtGame**

## 6.29 hasproperty.cpp File Reference

```
#include <dtDIS/hasproperty.h>  
#include <cstdlib>
```

## 6.30 hasproperty.h File Reference

```
#include <vector>
#include <dtCore/refptr.h>
#include <dtDAL/actorproxy.h>
#include <string>
#include <dtDIS/dtdisexport.h>
```

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.31 idisplugin.cpp File Reference

```
#include <dtDIS/idisplugin.h>
```

## 6.32 idisplugin.h File Reference

```
#include <vector>
#include <dtCore/refptr.h>
#include <dtGame/gameactor.h>
#include <dtDIS/dtdisexport.h>
```

### Classes

- class **IDISPlugin**  
*the interface for all **DIS** (p. 13) plugins*

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*
- namespace **dtGame**

## 6.33 imessagetopacketadapter.cpp File Reference

```
#include <dtDIS/imessagetopacketadapter.h>
```

## 6.34 imessagetopacketadapter.h File Reference

```
#include <dtDIS/dtdisexport.h>
```

### Classes

- class **IMessageToPacketAdapter**  
*the interface for translating Delta3D messages into **DIS** (p. 13) packets.*

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*
- namespace **dtGame**

## 6.35 libraryregistry.h File Reference

```
#include <dtUtil/librarysharingmanager.h>
```

```
#include <dtCore/refptr.h>
```

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.36 mainpage.h File Reference

### 6.36.1 Detailed Description

This file contains Doxygen special commands and text for the **Main Page** (p. ??) and some other minor aspects of this documentation. It is not part of Delta3D.

## 6.37 mastercomponent.cpp File Reference

```
#include <dtDIS/mastercomponent.h>
#include <dtDIS/sharedstate.h>
#include <DIS/PDUType.h>
#include <dtUtil/functor.h>
#include <list>
#include <dtDIS/dllfinder.h>
#include <dtDIS/plugins/default/defaultplugin.h>
#include <dtActors/engineactorregistry.h>
#include <dtActors/coordinateconfigactor.h>
```

## 6.38 mastercomponent.h File Reference

```
#include <dtGame/gmcomponent.h>
#include <dtDIS/pluginmanager.h>
#include <dtDIS/connection.h>
#include <dtDIS/outgoingmessage.h>
#include <DIS/IncomingMessage.h>
#include <string>
#include <dtDIS/dtdisexport.h>
```

### Classes

- class **MasterComponent**

*Supports a framework to translate **DIS** (p. 13) PDUs into dtGame::Message instances.*

### Namespaces

- namespace **dtDIS**

*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.39 outgoingmessage.cpp File Reference

```
#include <dtDIS/outgoingmessage.h>
#include <dtDIS/connection.h>
#include <dtDIS/imessagetopacketadapter.h>
#include <DIS/Pdu.h>
#include <dtGame/message.h>
#include <dtUtil/log.h>
#include <dtDIS/containerutils.h>
```

## 6.40 outgoingmessage.h File Reference

```
#include <map>
#include <dtGame/messagetype.h>
#include <dtDIS/dtdisexport.h>
#include <DIS/DataStream.h>
```

### Classes

- class **OutgoingMessage**  
*A framework for translating dtGame::Message instances to PDUs.*

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*
- namespace **dtGame**

## 6.41 pluginmanager.cpp File Reference

```
#include <dtDIS/pluginmanager.h>
#include <dtUtil/log.h>
#include <dtUtil/librarysharingmanager.h>
#include <osgDB/FileNameUtils>
#include <dtUtil/stringutils.h>
```

## 6.42 pluginmanager.h File Reference

```
#include <string>
#include <map>
#include <dtDIS/createdestroypolicy.h>
#include <dtDIS/idisplugin.h>
#include <dtCore/sigslot.h>
#include <dtDIS/dtdisexport.h>
```

### Classes

- class **PluginManager**  
*manages a container of available plugins.*

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.43 pluginsymbols.cpp File Reference

```
#include <dtDIS/plugins/default/pluginsymbols.h>  
#include <dtDIS/plugins/default/defaultplugin.h>  
#include <dtGame/gamemanager.h>  
#include <dtDAL/actortype.h>
```

## 6.44 pluginsymbols.h File Reference

```
#include <dtDIS/plugins/default/dtdisdefaultpluginexport.h>
```

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

### Functions

- DT\_DIS\_DEFAULT\_EXPORT IDISPlugin \* **CreateComponent** ()
- DT\_DIS\_DEFAULT\_EXPORT void **DestroyComponent** (IDISPlugin \*plugin)

## 6.45 propertyname.cpp File Reference

```
#include <dtDIS/propertyname.h>
```

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.46 propertyname.h File Reference

```
#include <dtDIS/dtdisexport.h>
#include <string>
#include <dtUtil/refstring.h>
```

### Classes

- struct **EnginePropertyName**
- struct **EntityPropertyName**

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.47 removeentityprocessor.cpp File Reference

```
#include <dtDIS/plugins/default/removeentityprocessor.h>
#include <dtDIS/outgoingmessage.h>
#include <dtDIS/sharedstate.h>
#include <DIS/RemoveEntityPdu.h>
#include <DIS/AcknowledgePdu.h>
#include <dtDIS/acknowledgeconstants.h>
#include <dtDAL/actorproxy.h>
```

## 6.48 removeentityprocessor.h File Reference

```
#include <DIS/IPacketProcessor.h>
```

```
#include <dtDIS/plugins/default/dtdisdefaultpluginexport.h>
```

### Classes

- class **RemoveEntityProcessor**  
*the plugin's model of how to respond to the DIS::RemoveEntityPdu message.*

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.49 sharedstate.cpp File Reference

```
#include <dtDIS/sharedstate.h>  
#include <dtDAL/actorproxy.h>  
#include <dtUtil/xercesparser.h>  
#include <dtDIS/disxml.h>  
#include <cstdint>
```

## 6.50 sharedstate.h File Reference

```
#include <dtDIS/dtdisexport.h>
#include <DIS/EntityID.h>
#include <DIS/EntityType.h>
#include <dtDAL/actortype.h>
#include <dtDAL/resourcedescriptor.h>
#include <dtCore/uniqueid.h>
#include <dtUtil/coordinates.h>
#include <map>
#include <dtDIS/entityidcompare.h>
#include <dtDIS/activeentitycontrol.h>
```

### Classes

- struct **ActorMapConfig**  
*a structure to maintain the one-to-one relationship between the DIS::EntityID and the dtDAL::ActorType.*
- struct **ConnectionData**  
*the information needed to connect to the **DIS** (p. 13) network.*
- struct **ResourceMapConfig**  
*a structure to maintain the relationship between the DIS::EntityID and the resources available.*
- class **SharedState**  
*The data to be shared among plugins.*

### Namespaces

- namespace **dtCore**
- namespace **dtDAL**
- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

## 6.51 valuemaps.cpp File Reference

```
#include <dtDIS/valuemaps.h>
#include <dtDIS/articulationconstants.h>
#include <dtGame/deadreckoningcomponent.h>
#include <DIS/EntityType.h>
#include <dtDIS/entitytypeconstants.h>
```

## 6.52 valuemaps.h File Reference

```
#include <dtDIS/dtdisexport.h>
```

```
#include <string>
```

### Classes

- struct **ValueMap**  
*a name scope for all sorts of mappings*

### Namespaces

- namespace **dtDIS**  
*the Delta3D support framework for the **DIS** (p. 13) network protocol.*

# Index

---

## - Symbols -

- ~ActorUpdateToEntityState
  - dtDIS::ActorUpdateToEntityState, 26
- ~ConnectionXMLHandler
  - dtDIS::ConnectionXMLHandler, 30
- ~CreateEntityProcessor
  - dtDIS::CreateEntityProcessor, 31
- ~DefaultPlugin
  - dtDIS::DefaultPlugin, 32
- ~ESPduProcessor
  - dtDIS::ESPduProcessor, 37
- ~EntityMapXMLHandler
  - dtDIS::EntityMapXMLHandler, 35
- ~IDISPlugin
  - dtDIS::IDISPlugin, 38
- ~IMessageToPacketAdapter
  - dtDIS::IMessageToPacketAdapter, 39
- ~MasterComponent
  - dtDIS::MasterComponent, 41
- ~RemoveEntityProcessor
  - dtDIS::RemoveEntityProcessor, 48
- ~SharedState
  - dtDIS::SharedState, 51

## - A -

- ACKNOWLEDGE\_CREATE\_ENTITY
  - dtDIS, 17
- ACKNOWLEDGE\_REMOVE\_ENTITY
  - dtDIS, 17
- ACKNOWLEDGE\_START\_RESUME
  - dtDIS, 17
- ACKNOWLEDGE\_STOP\_FREEZE
  - dtDIS, 17
- acknowledgeconstants.h, 55
- AcknowledgeFlag
  - dtDIS, 17
- activeentitycontrol.cpp, 56
- activeentitycontrol.h, 57
- ActorUpdateToEntityState
  - dtDIS::ActorUpdateToEntityState, 26
- actorupdatetoentitystate.cpp, 58
- actorupdatetoentitystate.h, 59
- AdapterMap
  - dtDIS::OutgoingMessage, 44
- AddActor
  - dtDIS::ESPduProcessor, 37
- AddActorMapping
  - dtDIS::ActorMapConfig, 25
- AddAdaptor
  - dtDIS::OutgoingMessage, 44
- AddEntity
  - dtDIS::ActiveEntityControl, 23
- AddResourceMapping
  - dtDIS::ResourceMapConfig, 49
- APPEARANCE
  - dtDIS::EntityPropertyName, 36
- application\_id
  - dtDIS::ConnectionData, 28
- ApplyFullUpdateToProxy
  - dtDIS::ESPduProcessor, 37

- ARTICULATED\_PART
  - dtDIS::Articulation, 19
- ARTICULATION
  - dtDIS::EnginePropertyName, 33
- articulationconstants.cpp, 60
- articulationconstants.h, 61
- ATTACHED\_PART
  - dtDIS::Articulation, 19

## - C -

- characters
  - dtDIS::ConnectionXMLHandler, 30
  - dtDIS::EntityMapXMLHandler, 35
- ClearAll
  - dtDIS::ActiveEntityControl, 23
- ClearData
  - dtDIS::OutgoingMessage, 44
- Connect
  - dtDIS::Connection, 27
- connection.cpp, 62
- connection.h, 63
- ConnectionXMLHandler
  - dtDIS::ConnectionXMLHandler, 30
- containerutils.h, 64
- Convert
  - dtDIS::ActorUpdateToEntityState, 26
  - dtDIS::IMessageToPacketAdapter, 39
- CreateComponent
  - dtDIS, 18
- createdestroypolicy.h, 65
- createdestroypolicy.inl, 66
- CreateEntityProcessor
  - dtDIS::CreateEntityProcessor, 31
- createentityprocessor.cpp, 67
- createentityprocessor.h, 68

## - D -

- DEAD\_RECKONING\_ALGORITHM
  - dtDIS::EnginePropertyName, 33
- DEFAULT\_CONNECTION\_DATA
  - dtDIS::ConnectionXMLHandler, 30
- DefaultPlugin
  - dtDIS::DefaultPlugin, 32
- defaultplugin.cpp, 69
- defaultplugin.h, 70
- DestroyComponent
  - dtDIS, 18
- DIS, 13
- Disconnect
  - dtDIS::Connection, 27
- DispatchNetworkMessage
  - dtDIS::MasterComponent, 41
- disxml.cpp, 71
- disxml.h, 72
- dllfinder.h, 73
- DOF\_NODE\_NAME
  - dtDIS::EnginePropertyName, 33
- DOMAIN\_AIR
  - dtDIS, 17
- DOMAIN\_LAND

- dtDIS, 17
- DOMAIN\_SPACE
  - dtDIS, 17
- DOMAIN\_SUBSURFACE
  - dtDIS, 17
- DOMAIN\_SURFACE
  - dtDIS, 17
- DomainType
  - dtDIS, 17
- DT\_DIS\_DEFAULT\_EXPORT
  - dtdefaultpluginexport.h, 74
- DT\_DIS\_EXPORT
  - dtdefaultpluginexport.h, 75
- dtCore, 14
- dtDAL, 15
- dtDIS, 16
  - ACKNOWLEDGE\_CREATE\_ENTITY, 17
  - ACKNOWLEDGE\_REMOVE\_ENTITY, 17
  - ACKNOWLEDGE\_START\_RESUME, 17
  - ACKNOWLEDGE\_STOP\_FREEZE, 17
  - AcknowledgeFlag, 17
  - CreateComponent, 18
  - DestroyComponent, 18
  - DOMAIN\_AIR, 17
  - DOMAIN\_LAND, 17
  - DOMAIN\_SPACE, 17
  - DOMAIN\_SUBSURFACE, 17
  - DOMAIN\_SURFACE, 17
  - DomainType, 17
  - RESPONSE\_ABLE\_TO\_COMPLY, 18
  - RESPONSE\_OTHER, 18
  - RESPONSE\_UNABLE\_TO\_COMPLY, 18
  - ResponseFlag, 17
- dtDIS::ActiveEntityControl, 23
  - AddEntity, 23
  - ClearAll, 23
  - GetActor, 23
  - GetEntity, 23
  - RemoveEntity, 23
- dtDIS::ActorMapConfig, 25
  - AddActorMapping, 25
  - GetMappedActor, 25
  - RemoveActorMapping, 25
- dtDIS::ActorUpdateToEntityState, 26
  - ~ActorUpdateToEntityState, 26
  - ActorUpdateToEntityState, 26
  - Convert, 26
- dtDIS::Articulation, 19
  - ARTICULATED\_PART, 19
  - ATTACHED\_PART, 19
  - MOTION\_AZIMUTH, 19
  - MOTION\_AZIMUTH\_RATE, 19
  - MOTION\_ELEVATION, 19
  - MotionType, 19
  - ParameterTypeDesignator, 19
  - PART\_PRIMARY\_GUN, 19
  - PART\_PRIMARY\_TURRET, 19
  - PART\_SECONDARY\_GUN, 19
  - PartType, 19
- dtDIS::Articulation::NodeName, 43
  - NODE\_PRIMARY\_GUN, 43
  - NODE\_PRIMARY\_TURRET, 43
  - NODE\_SECONDARY\_GUN, 43
- dtDIS::Connection, 27
  - Connect, 27
  - Disconnect, 27
  - Receive, 27
  - Send, 27
- dtDIS::ConnectionData, 28
  - application\_id, 28
  - exercise\_id, 28
  - ip, 28
  - MTU, 28
  - plug\_dir, 28
  - port, 28
  - site\_id, 28
- dtDIS::ConnectionXMLHandler, 29
  - ~ConnectionXMLHandler, 30
  - characters, 30
  - ConnectionXMLHandler, 30
  - DEFAULT\_CONNECTION\_DATA, 30
  - endDocument, 30
  - endElement, 30
  - endPrefixMapping, 30
  - GetConnectionData, 30
  - ignorableWhitespace, 30
  - processingInstruction, 30
  - setDocumentLocator, 30
  - skippedEntity, 30
  - startDocument, 30
  - startElement, 30
  - startPrefixMapping, 30
- dtDIS::CreateEntityProcessor, 31
  - ~CreateEntityProcessor, 31
  - CreateEntityProcessor, 31
  - Process, 31
- dtDIS::DefaultPlugin, 32
  - ~DefaultPlugin, 32
  - DefaultPlugin, 32
  - Finish, 32
  - Start, 32
- dtDIS::EnginePropertyName, 33
  - ARTICULATION, 33
  - DEAD\_RECKONING\_ALGORITHM, 33
  - DOF\_NODE\_NAME, 33
  - ENTITY\_LINEAR\_VELOCITY, 33
  - ENTITY\_LOCATION, 33
  - ENTITY\_MARKING, 33
  - ENTITY\_ORIENTATION, 33
  - GROUND\_CLAMP, 33
  - LAST\_KNOWN\_LOCATION, 33
  - LAST\_KNOWN\_ORIENTATION, 34
  - RESOURCE\_DAMAGE\_DESTROYED, 34
  - RESOURCE\_DAMAGE\_OFF, 34
  - RESOURCE\_DAMAGE\_ON, 34
- dtDIS::EntityMapXMLHandler, 35
  - ~EntityMapXMLHandler, 35
  - characters, 35
  - endDocument, 35
  - endElement, 35
  - endPrefixMapping, 35
  - EntityMapXMLHandler, 35
  - ignorableWhitespace, 35
  - processingInstruction, 35
  - setDocumentLocator, 35
  - skippedEntity, 35
  - startDocument, 35
  - startElement, 35
  - startPrefixMapping, 35
- dtDIS::EntityPropertyName, 36
  - APPEARANCE, 36
  - ENTITY\_TYPE, 36

- dtDIS::ESPduProcessor, 37
    - ~ESPduProcessor, 37
    - AddActor, 37
    - ApplyFullUpdateToProxy, 37
    - ESPduProcessor, 37
    - Process, 37
    - SendPartialUpdate, 37
  - dtDIS::IDISPlugin, 38
    - ~IDISPlugin, 38
    - Finish, 38
    - Start, 38
  - dtDIS::IMessageToPacketAdapter, 39
    - ~IMessageToPacketAdapter, 39
    - Convert, 39
  - dtDIS::MasterComponent, 40
    - ~MasterComponent, 41
    - DispatchNetworkMessage, 41
    - GetIncomingMessage, 41
    - GetOutgoingMessage, 41
    - GetSharedState, 41
    - LoadPlugins, 41
    - MasterComponent, 41
    - OnAddedToGM, 42
    - OnPluginLoaded, 42
    - OnPluginUnloaded, 42
    - OnRemovedFromGM, 42
    - ProcessMessage, 42
    - UnloadPlugins, 42
  - dtDIS::OutgoingMessage, 44
    - AdapterMap, 44
    - AddAdaptor, 44
    - ClearData, 44
    - GetAdapters, 44
    - GetData, 44
    - Handle, 44
    - OutgoingMessage, 44
    - RemoveAdaptor, 44
  - dtDIS::PluginManager, 46
    - GetLoadedSignal, 47
    - GetRegistry, 47
    - GetUnloadedSignal, 47
    - LibLoaderT, 46
    - LibraryRegistry, 46
    - LoadPlugin, 47
    - PluginSignal, 46
    - RegistryEntry, 47
    - UnloadAllPlugins, 47
    - UnloadPlugin, 47
  - dtDIS::RemoveEntityProcessor, 48
    - ~RemoveEntityProcessor, 48
    - Process, 48
    - RemoveEntityProcessor, 48
  - dtDIS::ResourceMapConfig, 49
    - AddResourceMapping, 49
    - GetMappedResource, 49
    - RemoveResourceMapping, 49
  - dtDIS::SharedState, 50
    - ~SharedState, 51
    - GetActiveEntityControl, 51
    - GetActorMap, 51
    - GetApplicationID, 51
    - GetConnectionData, 51
    - GetCoordinateConverter, 51
    - GetResourceMap, 51
    - GetSiteID, 51
    - SetApplicationID, 51
    - SetConnectionData, 51
    - SetCoordinateConverter, 51
    - SetSiteID, 51
    - SharedState, 51
  - dtDIS::ValueMap, 52
    - GetArticulationMotionPropertyName, 52
    - GetArticulationNodeName, 52
    - GetDeadReckoningModelPropertyValue, 52
    - GetRequiresGroundClamping, 52
  - dtDIS::XMLFiles, 54
    - XML\_CONNECTION\_SCHEMA\_FILE, 54
    - XML\_MAPPING\_SCHEMA\_FILE, 54
  - dtDisdefaultpluginexport.h, 74
    - DT\_DIS\_DEFAULT\_EXPORT, 74
  - dtDisexport.h, 75
    - DT\_DIS\_EXPORT, 75
  - dtGame, 20
  - dtUtil, 21
    - find\_multimap\_pair, 21
- E -**
- endDocument
    - dtDIS::ConnectionXMLHandler, 30
    - dtDIS::EntityMapXMLHandler, 35
  - endElement
    - dtDIS::ConnectionXMLHandler, 30
    - dtDIS::EntityMapXMLHandler, 35
  - endPrefixMapping
    - dtDIS::ConnectionXMLHandler, 30
    - dtDIS::EntityMapXMLHandler, 35
  - ENTITY\_LINEAR\_VELOCITY
    - dtDIS::EnginePropertyName, 33
  - ENTITY\_LOCATION
    - dtDIS::EnginePropertyName, 33
  - ENTITY\_MARKING
    - dtDIS::EnginePropertyName, 33
  - ENTITY\_ORIENTATION
    - dtDIS::EnginePropertyName, 33
  - ENTITY\_TYPE
    - dtDIS::EntityPropertyName, 36
  - entityidcompare.cpp, 76
  - entityidcompare.h, 77
  - EntityMapXMLHandler
    - dtDIS::EntityMapXMLHandler, 35
  - entitytypeconstants.h, 78
  - espduapplicator.cpp, 79
  - espduapplicator.h, 80
  - ESPduProcessor
    - dtDIS::ESPduProcessor, 37
  - espduprocessor.cpp, 81
  - espduprocessor.h, 82
  - exercise\_id
    - dtDIS::ConnectionData, 28
- F -**
- find\_multimap\_pair
    - dtUtil, 21
  - Finish
    - dtDIS::DefaultPlugin, 32
    - dtDIS::IDISPlugin, 38
- G -**
- GetActiveEntityControl
    - dtDIS::SharedState, 51
  - GetActor

dtDIS::ActiveEntityControl, 23  
 GetActorMap  
   dtDIS::SharedState, 51  
 GetAdapters  
   dtDIS::OutgoingMessage, 44  
 GetApplicationID  
   dtDIS::SharedState, 51  
 GetArticulationMotionPropertyName  
   dtDIS::ValueMap, 52  
 GetArticulationNodeName  
   dtDIS::ValueMap, 52  
 GetConnectionData  
   dtDIS::ConnectionXMLHandler, 30  
   dtDIS::SharedState, 51  
 GetCoordinateConverter  
   dtDIS::SharedState, 51  
 GetData  
   dtDIS::OutgoingMessage, 44  
 GetDeadReckoningModelPropertyValue  
   dtDIS::ValueMap, 52  
 GetEntity  
   dtDIS::ActiveEntityControl, 23  
 GetIncomingMessage  
   dtDIS::MasterComponent, 41  
 GetLoadedSignal  
   dtDIS::PluginManager, 47  
 GetMappedActor  
   dtDIS::ActorMapConfig, 25  
 GetMappedResource  
   dtDIS::ResourceMapConfig, 49  
 GetOutgoingMessage  
   dtDIS::MasterComponent, 41  
 GetRegistry  
   dtDIS::PluginManager, 47  
 GetRequiresGroundClamping  
   dtDIS::ValueMap, 52  
 GetResourceMap  
   dtDIS::SharedState, 51  
 GetSharedState  
   dtDIS::MasterComponent, 41  
 GetSiteID  
   dtDIS::SharedState, 51  
 GetUnloadedSignal  
   dtDIS::PluginManager, 47  
 GROUND\_CLAMP  
   dtDIS::EnginePropertyName, 33

**- H -**

Handle  
   dtDIS::OutgoingMessage, 44  
 hasproperty.cpp, 83  
 hasproperty.h, 84

**- I -**

idisplugin.cpp, 85  
 idisplugin.h, 86  
 ignorableWhitespace  
   dtDIS::ConnectionXMLHandler, 30  
   dtDIS::EntityMapXMLHandler, 35  
 imessagetopacketadapter.cpp, 87  
 imessagetopacketadapter.h, 88  
 inc/ Directory Reference, 9  
 inc/dtDIS/ Directory Reference, 8  
 inc/dtDIS/plugins/ Directory Reference, 10  
 inc/dtDIS/plugins/default/ Directory Reference, 5

ip  
   dtDIS::ConnectionData, 28

**- L -**

LAST\_KNOWN\_LOCATION  
   dtDIS::EnginePropertyName, 33  
 LAST\_KNOWN\_ORIENTATION  
   dtDIS::EnginePropertyName, 34  
 LibLoaderT  
   dtDIS::PluginManager, 46  
 LibraryRegistry  
   dtDIS::PluginManager, 46  
 libraryregistry.h, 89  
 LoadPlugin  
   dtDIS::PluginManager, 47  
 LoadPlugins  
   dtDIS::MasterComponent, 41

**- M -**

mainpage.h, 90  
 MasterComponent  
   dtDIS::MasterComponent, 41  
 mastercomponent.cpp, 91  
 mastercomponent.h, 92  
 MOTION\_AZIMUTH  
   dtDIS::Articulation, 19  
 MOTION\_AZIMUTH\_RATE  
   dtDIS::Articulation, 19  
 MOTION\_ELEVATION  
   dtDIS::Articulation, 19  
 MotionType  
   dtDIS::Articulation, 19  
 MTU  
   dtDIS::ConnectionData, 28

**- N -**

NODE\_PRIMARY\_GUN  
   dtDIS::Articulation::NodeName, 43  
 NODE\_PRIMARY\_TURRET  
   dtDIS::Articulation::NodeName, 43  
 NODE\_SECONDARY\_GUN  
   dtDIS::Articulation::NodeName, 43

**- O -**

OnAddedToGM  
   dtDIS::MasterComponent, 42  
 OnPluginLoaded  
   dtDIS::MasterComponent, 42  
 OnPluginUnloaded  
   dtDIS::MasterComponent, 42  
 OnRemovedFromGM  
   dtDIS::MasterComponent, 42  
 OutgoingMessage  
   dtDIS::OutgoingMessage, 44  
 outgoingmessage.cpp, 93  
 outgoingmessage.h, 94

**- P -**

ParameterTypeDesignator  
   dtDIS::Articulation, 19  
 PART\_PRIMARY\_GUN  
   dtDIS::Articulation, 19  
 PART\_PRIMARY\_TURRET  
   dtDIS::Articulation, 19  
 PART\_SECONDARY\_GUN

- dtDIS::Articulation, 19
- PartType
  - dtDIS::Articulation, 19
- plug\_dir
  - dtDIS::ConnectionData, 28
- pluginmanager.cpp, 95
- pluginmanager.h, 96
- PluginSignal
  - dtDIS::PluginManager, 46
- pluginsymbols.cpp, 97
- pluginsymbols.h, 98
- port
  - dtDIS::ConnectionData, 28
- Process
  - dtDIS::CreateEntityProcessor, 31
  - dtDIS::ESPduProcessor, 37
  - dtDIS::RemoveEntityProcessor, 48
- processingInstruction
  - dtDIS::ConnectionXMLHandler, 30
  - dtDIS::EntityMapXMLHandler, 35
- ProcessMessage
  - dtDIS::MasterComponent, 42
- propertyname.cpp, 99
- propertyname.h, 100
- R -**
- Receive
  - dtDIS::Connection, 27
- RegistryEntry
  - dtDIS::PluginManager, 47
- RemoveActorMapping
  - dtDIS::ActorMapConfig, 25
- RemoveAdaptor
  - dtDIS::OutgoingMessage, 44
- RemoveEntity
  - dtDIS::ActiveEntityControl, 23
- RemoveEntityProcessor
  - dtDIS::RemoveEntityProcessor, 48
- removeentityprocessor.cpp, 101
- removeentityprocessor.h, 102
- RemoveResourceMapping
  - dtDIS::ResourceMapConfig, 49
- RESOURCE\_DAMAGE\_DESTROYED
  - dtDIS::EnginePropertyName, 34
- RESOURCE\_DAMAGE\_OFF
  - dtDIS::EnginePropertyName, 34
- RESOURCE\_DAMAGE\_ON
  - dtDIS::EnginePropertyName, 34
- RESPONSE\_ABLE\_TO\_COMPLY
  - dtDIS, 18
- RESPONSE\_OTHER
  - dtDIS, 18
- RESPONSE\_UNABLE\_TO\_COMPLY
  - dtDIS, 18
- ResponseFlag
  - dtDIS, 17
- S -**
- Send
  - dtDIS::Connection, 27
- SendPartialUpdate
  - dtDIS::ESPduProcessor, 37
- SetApplicationID
  - dtDIS::SharedState, 51
- SetConnectionData
  - dtDIS::SharedState, 51
- SetCoordinateConverter
  - dtDIS::SharedState, 51
- setDocumentLocator
  - dtDIS::ConnectionXMLHandler, 30
  - dtDIS::EntityMapXMLHandler, 35
- SetSiteID
  - dtDIS::SharedState, 51
- SharedState
  - dtDIS::SharedState, 51
- sharedstate.cpp, 103
- sharedstate.h, 104
- site\_id
  - dtDIS::ConnectionData, 28
- skippedEntity
  - dtDIS::ConnectionXMLHandler, 30
  - dtDIS::EntityMapXMLHandler, 35
- src/ Directory Reference, 12
- src/dtDIS/ Directory Reference, 7
- src/dtDIS/plugins/ Directory Reference, 11
- src/dtDIS/plugins/default/ Directory Reference, 6
- Start
  - dtDIS::DefaultPlugin, 32
  - dtDIS::IDISPlugin, 38
- startDocument
  - dtDIS::ConnectionXMLHandler, 30
  - dtDIS::EntityMapXMLHandler, 35
- startElement
  - dtDIS::ConnectionXMLHandler, 30
  - dtDIS::EntityMapXMLHandler, 35
- startPrefixMapping
  - dtDIS::ConnectionXMLHandler, 30
  - dtDIS::EntityMapXMLHandler, 35
- U -**
- UnloadAllPlugins
  - dtDIS::PluginManager, 47
- UnloadPlugin
  - dtDIS::PluginManager, 47
- UnloadPlugins
  - dtDIS::MasterComponent, 42
- V -**
- valuemaps.cpp, 105
- valuemaps.h, 106
- X -**
- XML\_CONNECTION\_SCHEMA\_FILE
  - dtDIS::XMLFiles, 54
- XML\_MAPPING\_SCHEMA\_FILE
  - dtDIS::XMLFiles, 54