

1. Scope

The objective of this DMIS part standard is to define a collection of object interfaces, based on DMIS Part 1, that provide interoperability between DMIS client applications, a DMIS server, and DMIS mathematics and equipment modules. This DMIS part standard also documents the behavior of each object interface in the context of DMIS Part 1. The DMIS server and DMIS modules are essentially black boxes that hide vendor implementation details from the calling application. The documented interfaces provide a window into those black boxes, enabling access to and control of the implementation using the capabilities defined in DMIS, without requiring a knowledge of the implementation details. Therefore, conformance to this DMIS part standard enables a DMIS component to “plug-and-play” with other conforming DMIS components.

An environment using the DMIS object interfaces described in this part standard is depicted in Figure 1 (DMIS Object Interface Environment). This figure shows the relationships among various DMIS Part 2 components. The modules have objects which are exposed by interfaces described by this DMIS part standard. These object interfaces are illustrated as “lollypops”. Other modules or applications can communicate with a module only through these exposed interfaces. The scope of this DMIS part standard is graphically illustrated by the two dashed DMIS Part 2 boxes. For example, a DMIS client application can use functionality found in the DMIS server via exposed interfaces. To service DMIS client application requests, the DMIS server may use functionality of other modules by using their object interfaces. An object request broker (ORB) provides transport services to communicate between components.

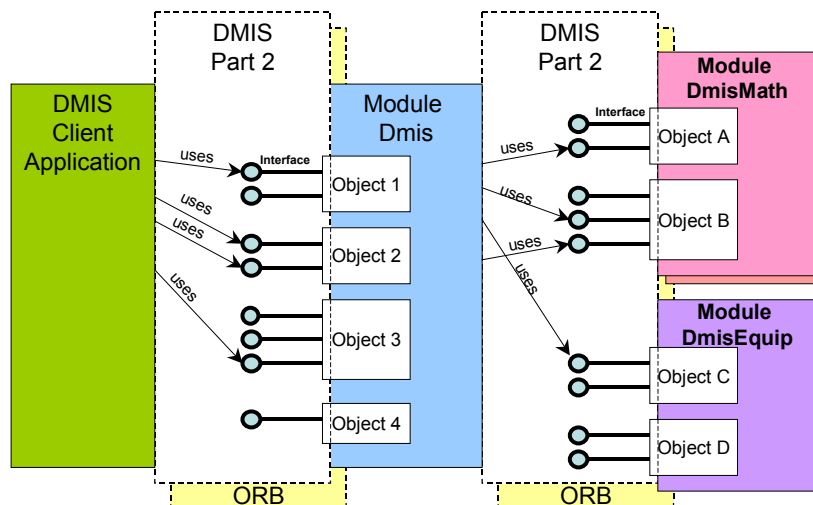


Figure 1 – DMIS Object Interface Environment

The DMIS Object Interface Environment currently describes three DMIS modules.

- Dmis Server Module – contains a mapping of DMIS Part 1 commands to object interfaces and provides additional interfaces to load, modify, and execute DMIS programs, generate DMIS statements from object operations, and notify client application of events and errors.
- Dmis Mathematics Module – contains a set of interfaces for handling all DMIS-related mathematics requests.
- Dmis Equipment Module – contains general equipment-related DMIS object interfaces.

2. Conformance

Most object request brokers (ORBs), which provide the inter-module communications and transport for the operations described within this specification, operate from statically defined and compiled interface definitions. These require a full and complete definition of the structures, interfaces, enumerations, exceptions, and other

6. Interface Reference

The interface definitions herein are described using the CORBA Interface Definition Language. This format was chosen because of its simplicity, its non-proprietary nature, and the existence of well-defined mappings to other interface mechanisms. It was deemed undesirable, given a suitable existing specification, to create a new format solely to define the interfaces described herein.

The IDL contains various types, including the following ones used within this specification. A brief description of each such type is presented below; however the reader is directed to the CORBA specification, which provides a concise and complete description for each.

Fundamental types

boolean	- a bistate type with value of true or false.
short	- a signed 16 bit integral numeric type.
unsigned short	- an unsigned 16 bit integral numeric type.
long	- a signed 32 bit integral numeric type.
unsigned long	- an unsigned 32 bit integral numeric type.
float	- a 32 bit IEEE floating point type, with a signed mantissa and a signed exponent.
double	- a 64 bit IEEE floating point type, with a signed mantissa and a signed exponent.
string	- an unbounded (variable length) sequence of 8 bit characters.

Constructed types

typedef	- defines a synonym for a type, which retains all of the characteristics of the original type.
enum	- an enumeration, that is, a grouping of integral constants which take on the ordered value of their zero-based sequential offset within the group (e.g. given enum { A, B, C}; then A=0, B=1, C=2).
struct	- a structure, that is, a grouping of named data values where all values exist simultaneously and are accessed by name.
union	- a grouping of named data values where only one value exists at a time, containing a selection discriminator that indicates the current named value.
interface	- a collection of named attributes and operations; a reference to an interface accesses a (possibly remote) instance of an object.
sequence	- an ordered collection of fundamental and constructed types; only unbounded sequences are used within this specification.
exception	- a structure that is used solely for synchronous reporting of errors.

A consistent format for each of these types is used within this reference.

Description	- used generally for interfaces and their operations, structures and unions with their members, typedefs, enumerations and constant definitions, and exceptions.
Members	- identifies the members of a struct, with their description.
Switch	- identifies the type of the discriminator for a union.
Case	- identifies the name and type of a member of the union, with its description.
Values	- identifies the name and value of enumerations and constants, with their description.
Base Interfaces	- identifies the name of the interface, if any, from which an interface inherits.
Related Interfaces	- identifies the names of any interfaces that have a related or similar function.
Operations	- identifies the actions that may be performed on an interface, along with the types, names, direction, and description of any parameters, and the type of its return value, if any.
Returns	- describes the return value, if any, from an interface operation.
Events	- identifies any asynchronous events which are generated during the execution of an operation.
Exceptions	- identifies any exceptions that may be raised during the execution of an operation; this does not include system exceptions, which may be implicitly raised by any operation.
Sequence Type	- identifies any typedefs for an unbounded sequence of the fundamental or constructed type.
Generates	- identifies any DMIS Part 1 statement which may be generated by invocation of an operation, depending upon the state of DMIS generation per section 5.4.3 DMIS Statement Generation.
DMIS References	
DMIS Refs	- identify any major and/or minor word, and/or any other section, of the DMIS Part 1 specification, which are of relevance to the type or operation.

6.2.2.76. enum Dmis::eWKPLAN

Values	Description
Dmis::eWKPLAN_XY	Indicates the XY workplane.
Dmis::eWKPLAN_YZ	Indicates the YZ workplane.
Dmis::eWKPLAN_ZX	Indicates the ZX workplane.
DMIS References	WKPLAN

6.2.3. Structures and Unions

6.2.3.1. union Dmis::Argument

Description	Defines an argument to a macro, serving as a placeholder identifying the type of data expected as a parameter to a macro.
Switch	Dmis::eARG_TYPE
	string label;
Case	Dmis::eARG_LABEL
Description	Signifies that the expected parameter will be the label (symbol name) portion of a symbol. This will be combined with a specified symbol type during execution of macro statements to look up a symbol. The value of this member is the local name within the macro by which the label is referenced. The value of the runtime parameter is a string containing the label name.
	string value;
Case	Dmis::eARG_VALUE
Description	Signifies that the expected parameter will be a value – either an explicit value, or a fully resolved variable value. The value of the runtime parameter is an operand containing a value.
Sequence Type	typedef sequence<Dmis::Argument> Dmis::ArgSeq;
DMIS References	MACRO

6.2.3.2. struct Dmis::Axis

Description	Defines an axis component for a part alignment (datuming) operation.
Members	
	Dmis::Datum dat;
Description	Identifies a datum that references the feature used for defining a PCS axis.
	Dmis::ePCS_FACE dir;
Description	Identifies the axis and direction specified by the datum.
	ulong orig3;
Description	Identifies zero or more axes for translation, as bitmapped OR'd Dmis::ePCS_ORIG values. A zero value indicates no translation is to be performed.
Sequence Type	typedef sequence<Dmis::Axis> Dmis::AxisSeq;
DMIS References	DATSET

6.2.3.3. struct Dmis::CarriageDef

Description	Identifies parameters defining the range and orientation of an individual carriage.
Members	
	Dmis::Triple_t zero_ref;
Description	Identifies the offset in cartesian machine coordinates to the origin (smallest offset) of the carriage measuring volume.
	Dmis::Triple_t extent;
Description	Identifies the offset in cartesian machine coordinates to the extent (largest offset) of the carriage measuring volume. This is by definition a 3D diagonal to the zero_ref value.
	Dmis::Triple_t ijk_x;
Description	Identifies the carriage X direction. This must be a unit vector.
	Dmis::Triple_t ijk_y;
Description	Identifies the carriage Y direction. This must be a unit vector.

	<pre>Carriage Dmis::ObjectFactory::new_Carriage (in Dmis::Client client_id, in string sym_name, in Dmis::CarriageDef carg_def);</pre>
Description	Constructor. <i>carg_def</i> – identifies the requested measuring volume, which is used by the DME to select a carriage. The measuring volume must be less than or equal to that of at least one of the carriages. The DME will select the carriage whose measuring volume completely encloses the requested measuring volume, and whose machine axes match those of the supplied carriage definition. A list of available carriage definitions may be queried from the Dmis::Equipment object.
Events	Dmis::EventInsertSymbol.
Exceptions	Dmis::XcpRuntime (invalid <i>carg_def</i>).
Generates	CRGDEF.
DMIS Refs	CRGDEF.
	<pre>void get_carriage_data (out Dmis::CarriageDef carg_def);</pre>
Description	Obtains the user-specified carriage definition data (provided in the DMIS program or passed to the constructor). <i>carg_def</i> – obtains the user-specified carriage definition.
Returns	None.
Events	None.
Exceptions	None.
DMIS Refs	CRGDEF.
	<pre>void get_sensor_mount (out Dmis::Triple_t xvec, out Dmis::Triple_t zvec, out Dmis::Triple_t mnt_len);</pre>
Description	Obtains the defined relationship between the sensor coordinate system and the machine coordinate system. <i>xvec</i> – obtains the MCS direction of the sensor X axis. <i>zvec</i> – obtains the MCS direction of the sensor Z axis. <i>mnt_len</i> – obtains the nominal offset from the DME sensor reference point and the origin of the sensor coordinate system.
Returns	None.
Events	None.
Exceptions	None.
DMIS Refs	None.
	<pre>void set_sensor_mount (in Dmis::Client client_id, in Dmis::Triple_t xvec, in Dmis::Triple_t zvec, in Dmis::Triple_t mnt_len);</pre>
Description	Defines the relationship of the sensor coordinate system to the machine coordinate system. <i>xvec</i> – identifies the MCS direction of the sensor X axis. <i>zvec</i> – identifies the MCS direction of the sensor Z axis. <i>mnt_len</i> – identifies the nominal offset from the DME sensor reference point and the origin of the sensor coordinate system.
Returns	None.
Events	Dmis::EventSetCarriageCfg.
Exceptions	Dmis::XcpLocked, Dmis::XcpRuntime.
Generates	SNSMNT.
DMIS Refs	SNSMNT.
	<pre>double get_approach_dist();</pre>
Description	Obtains the approach distance for measurement.
Returns	A double containing the approach distance.
Events	None.
Exceptions	None.
DMIS Refs	None.
	<pre>double get_retract_dist();</pre>
Description	Obtains the retract distance for measurement.
Returns	A double containing the retract distance.

Base Interfaces	Dmis::FeatAct.
Related Interfaces	Dmis::FeatArc4.
Operations	
	<code>Dmis::eFEAT_INNER_OUTER get_inner_outer();</code>
Description	Identifies the direction, inner or outer, in which probe compensation was applied to raw data points when calculating the feature actual.
Returns	An enumerated value indicating the probe compensation direction.
Events	None.
Exceptions	None.
DMIS Refs	FEAT/ARC Output Format 2 INNER or OUTER
	<code>Dmis::Triple_t get_start_point();</code>
Description	Obtains the start point of the actual arc.
Returns	A position; shall be Dmis::eTRIPLE_CART type.
Events	None.
Exceptions	None.
DMIS Refs	FEAT/ARC Output Format 2 e1x,e1y,e1z.
	<code>Dmis::Triple_t get_mid_point();</code>
Description	Obtains the mid-point of the actual arc.
Returns	A position; shall be Dmis::eTRIPLE_CART type.
Events	None.
Exceptions	None.
DMIS Refs	FEAT/ARC Output Format 2 mx,my,mz.
	<code>Dmis::Triple_t get_end_point();</code>
Description	Obtains the end point of the actual arc.
Returns	A position; shall be Dmis::eTRIPLE_CART type.
Events	None.
Exceptions	None.
DMIS Refs	FEAT/ARC Output Format 2 e2x,e2y,e2z.
	<code>Dmis::Triple_t get_center_point();</code>
Description	Obtains the center point of the actual arc.
Returns	A position; shall be Dmis::eTRIPLE_CART type.
Events	None.
Exceptions	None.
DMIS Refs	FEAT/ARC Output Format 2 cx,cy,cz.
DMIS References	FEAT/ARC Format 2.

6.2.4.29. interface Dmis::FeatCircle

Description	Provides operations to create nominal circle features and obtain data about them.
Base Interfaces	Dmis::FeatNom.
Related Interfaces	Dmis::FeatACircle.
Operations	
	<code>Dmis::FeatCircle Dmis::ObjectFactory::new_FeatCircle (in Dmis::Client client_id, in string sym_name, in Dmis::eFEAT_INNER_OUTER inner_outer, in Dmis::Triple_t pos, in Dmis::Triple_t dir, in double diam);</code>
Description	Constructor. <i>inner_outer</i> – specifies whether the feature is an inner or outer feature. <i>pos</i> – the location of the center of the circle; must not be Dmis::eTRIPLE_IJK type. <i>dir</i> – a vector normal to the plane in which the circle lies; must be Dmis::eTRIPLE_IJK type. <i>diam</i> – the diameter of the circle; must be positive.
Events	Dmis::EventInsertSymbol.
Exceptions	Dmis::XcpLocked. Dmis::XcpRuntime: spec if (i) an incorrect enumeration type is used with any Dmis::Triple_t, (ii) dir has length zero, (iii) diam is not positive, (iv) a symbol of the same name is already defined.

Generates	FEAT.
DMIS Refs	FEAT/CIRCLE Input Format.
	<code>Dmis::eFEAT_INNER_OUTER get_inner_outer();</code>
Description	Identifies the feature as inner (material encloses the defined feature) or outer (material is enclosed by the defined feature).
Returns	An enumerated value indicating whether the feature was defined inner or outer.
Events	None.
Exceptions	None.
DMIS Refs	FEAT/CIRCLE Output Format INNER or OUTER
	<code>Dmis::Triple_t get_pos();</code>
Description	Obtains the center point of the nominal circle.
Returns	A position; shall not be Dmis::eTRIPLE_IJK type.
Events	None.
Exceptions	None.
DMIS Refs	FEAT/CIRCLE Output Format x,y,z or r,a,h.
	<code>Dmis::Triple_t get_dir();</code>
Description	Obtains the orientation vector of the nominal circle that is, the direction vector of the plane in which the circle lies.
Returns	A unit vector, type shall be Dmis::eTRIPLE_IJK.
Events	None.
Exceptions	None.
DMIS Refs	FEAT/CIRCLE Output Format i,j,k.
	<code>double get_diam();</code>
Description	Obtains the diameter of the nominal circle.
Returns	The diameter; shall be positive.
Events	None.
Exceptions	None.
DMIS Refs	FEAT/CIRCLE Output Format diam.
DMIS References	FEAT/CIRCLE.

6.2.4.30. interface Dmis::FeatACircle

Description	Provides operations to obtain data about actual circle features.
Base Interfaces	Dmis::FeatAct.
Related Interfaces	Dmis::FeatCircle.
Operations	
	<code>Dmis::eFEAT_INNER_OUTER get_inner_outer();</code>
Description	Identifies the direction, inner or outer, in which probe compensation was applied to raw data points when calculating the feature actual.
Returns	An enumerated value indicating the probe compensation direction.
Events	None.
Exceptions	None.
DMIS Refs	FEAT/CIRCLE Output Format INNER or OUTER
	<code>Dmis::Triple_t get_pos();</code>
Description	Obtains the center point of the actual circle.
Returns	A position; shall not be Dmis::eTRIPLE_IJK type.
Events	None.
Exceptions	None.
DMIS Refs	FEAT/CIRCLE Output Format x,y,z or r,a,h.
	<code>Dmis::Triple_t get_dir();</code>
Description	Obtains the orientation vector of the actual circle that is, the direction vector of the plane in which the circle lies.
Returns	A unit vector, type shall be Dmis::eTRIPLE_IJK.
Events	None.
Exceptions	None.