ZinfiniTree ZT Server--Base Specification

Version 1.0 by Zinfinitem LLC

Overview

Introduction

The purpose of this document is to provide a Base Specification for the ZinfiniTree ZT Server by Zinfinitem LLC. The ZT Server is designed as an extensible platform for building, deploying, running and sharing applications.

The specification describes a system especially suited for storing data related to objects where the rules for storing the data are very exact but where those rules are expected to change significantly during the lifetime of the object. While the rules change over time, there is still need to retain continuity of identification. As an example, a person is often thought of as a single continuous entity that retains an existence from birth until death. A person goes through dramatic changes in size, appearance, abilities, needs, wants, legal status and relationships throughout his/her life. Some people experience a change in their name at various points but are still thought of as the same person. As another example, a company over time can change its customers, employees, suppliers, products or ownership. It can acquire other companies and even change its name but in the minds of most people, it retains a continuous chain of existence.

The specification is also oriented towards handing data that needs to be viewed from multiple perspectives. This includes, for example, maintaining the object in multiple human languages and for multiple roles. Roles in the context of a company could include, for example, customer or vendor.

As the name ZT Server implies, an implementation of a ZT Server would fill the role of a server in a client server relationship. As with a typical client server relationship, the client sends requests to the server. The ZT Server receives the request, processes it and returns a response. Persisted data can be stored by the server in a conventional database. The document describes the requirements of the ZT server and includes a definition of messages that could flow between a client and the server.

Although the server contains features that are intended to be useful to the client, there is no specification for the client in the base specification except that it must be able to pass messages to the server using a protocol that the server supports. The client must be able to receive the response from the server and presumably be able to perform something useful with it. The client would typically handle the interaction with human users (if they exist).

The request to the server will usually be to maintain (create or change) data used to represent objects on the server or to retrieve data that was previously stored. In this sense a ZT Server is conceptually similar to a conventional database server.

Design Goals

The specification for the ZT Server was developed with a set of Design Goals. An implementation of the specification for ZT Server along with certain core client applications (specified separately) should have the following capabilities:

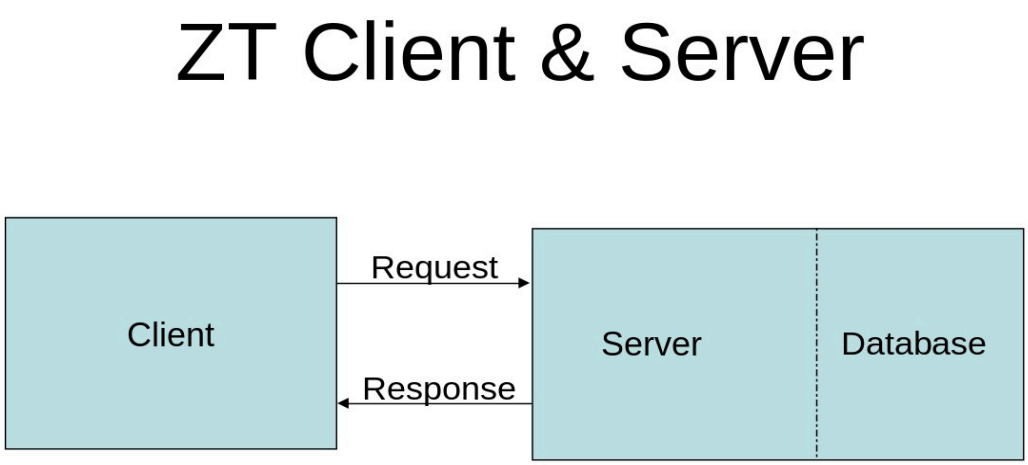
1. It should be possible to build applications with minimal programming using generically defined objects.
2. Applications should be multi-language (human) without having to do any special coding.
3. It should be possible to extend ZT Server applications and combine them with other ZT Server applications with minimal or no coding.
4. Applications should have access to a complete change history without coding. Objects that are linked to older versions of other objects should have access to the older values even if a new version of the object being linked to has changed.
5. It should be easy for users to share data and applications with each other.
6. It should be easy to identify the source of any objects in the system. It should also be easy to remove unwanted objects that were previously imported or created based on source or other criteria.
7. The design should be understandable and accessible. A person should be able to move from being an application user to application configurator to application developer to client or server developer in a natural progression.

In addition, listed below are goals for the specification document itself:

1. The document should be translatable from English to other languages without using characters from the English alphabet.
2. It should be possible to implement the design with a reasonable amount of effort using industry standard protocols and products that are available in open source implementations

ZT System--High Level Diagram

The graphic below depicts the relationship between the ZT Server and a client. The client initiates the relationship by sending a message to the server to maintain data on the server or to query data previously stored on the server. The server parses the message and interacts with data in a conventional database to process the message. The server then prepares a response message to indicate which data was maintained, return the data queried or return an appropriate error message.



Object Attributes

ZT Applications are maintained on a server by defining objects. The attributes of an object fall into four categories, Object Element values, Type values, Links and Maintained Templates. These are described in more detail in the Key Concepts section of this document.

Specification Layers

This Base Specification is described in abstract terms, independent of the details of implementation. Other specifications can be added that reference specific industry standards and protocols or add details not addressed by the base specification. The intention is that the Base Specification will eventually become stable. Other specifications are added to aid implementation and will adapt as surrounding technologies change.

The functionality of a ZT Server is built using multiple layers. Application developers can add functionality to an existing system. One developer can enrich the work of another. The Base Specification can be viewed as the trunk of a tree. Each specification that builds on the Base Specification elongates the trunk of the tree. There may be cases where two developers have competing work. This is like growing branches of the tree. There may be situations where two branches each contain valuable content. An integrator can take elements from each branch to build a third branch, modify it if necessary, test it and then distribute it to other users. Developers are free to add to standard specifications or branch (fork) them to meet their needs.

Database Definition

Any ZT Server implementation that can handle a request message that conforms to the standards and returns an appropriate response would be in compliance with the specification. However, it may be easier to understand the thinking behind the design of the messages if we also look at how data could be maintained on the server to achieve the design goals.

It was mentioned that the ZT Server can be connected to a conventional database. The database in the base specification is also an abstraction. It is considered to be a system that can store and retrieve data in/from tables. A table consists of rows and columns. The database must be able to retrieve data based on values being selected using a particular syntax. Typically databases support a variety of data types. The standard elements, those included in this specification, can be accommodated using only character data and correspond to a database field or column. Objects such as graphics, audio/video clips and web resources etc are supported by referring to locations that are possibly outside of the conventional database.

Key Concepts

Introduction

This section addresses some of the key concepts related to data organization

There are several aspects of the data organization that are used to achieve the design goals:

1. Object data is accessed through a set of Object Elements and the data is stored in Table Elements
2. The Table Elements are common to all objects

3. Objects share a common base key structure using a subset of the Table Elements

4. The common base key structure utilizes a timestamp to indicate the effective time for that row so that multiple versions of an object can be maintained

5. Object Elements can also refer to an extended key to handle values that are, for example, language dependent.

6. Links from one object to another are used to make it possible to extend an Object to include the elements of another Object

7. Objects can refer to other Objects using a Type relationship. The Type relationship allows objects to be categorized based on Type Value.

8. Access to Object Elements, Links and Type Values is controlled via Object Templates

9. Every piece of object data has a namespace that identifies where the data originated from. Inside of each ZT Server environment these namespaces are associated with codes to indicate the namespace in that environment. These are referred to as the ZT Instance Codes or ZTICs.

The Table Elements in the Base Specification are listed in Appendix 5. The purpose of these are explained in later sections of this document. Each Table Element has been assigned a numeric code to identify it.

The use of these Table Elements could vary by implementation. One approach would be to include all of the Table Elements in a single physical database table. In this case table elements not used for a particular row could be left blank. Alternatively, as an example, the values for Server DB--ZT Instance ID or Record Type ZT Instance Code and Record Type Code could be embedded in the table name and thereby spreading the contents into multiple physical database tables and minimizing the number blank elements required. The implementer could also choose other table schemas to i.e. increase efficiency of data access.

Object Kinds

Every object stored on the ZT server is assigned to an Object Kind. Some Object Kinds are part of the Base Specification and are used implement the ZT server. Other Object Kinds can be created by application developers. The Object Kinds defined would depend on the problem domain. For example, a individual may want to use Object Kinds to represent things such as persons, addresses, appointments. A business person may also be interested in Object Kinds that represent customers, sales orders and accounts. A chemist may be interested in an Object Kind that represents elements from the periodic table or an Object Kind to represent units of measure. Object Kinds use Object Kind Code 1. A list of the Object Kinds from the Base Specification can be found in Appendix 1. For objects created using Object Kinds from the Base Specification, the Object Codes shown in Appendix 1 are used in Table Element 2 to indicate the Kind of data being represented in that table row. Other Object Kinds will have their Codes assigned in the source ZT Instance. Table Element 1, which contains the Object Kind ZT Instance Code is used to differentiate the Object Kinds from different sources even though they may share the same Code.

Frequently the terms object kind and table will be used interchangeably. For example it could be said that this group of objects will be stored in the Person table meaning they are all of Object Kind Person. This translates roughly to the use of the term table in a relational database. It is not to imply that all of the objects will be stored in one physical database table. The implementer of the specification can decide how data will be apportioned to database tables.

Codes

The Code is used to identify individual instances of an object within the tables for an object kind. It could take the form of a sequentially assigned series of numbers and/or letters or Codes could be assigned by the client (user). A Code along with the Code ZT Instance Code uniquely identify an Object of a specific Kind. The ZT Instance code identifies the source of an object, meaning the ZT Instance that the object originated from. No maximum code length is specified in the Base Specification, but it will be included in other specifications that are layered on top of Base Specification. Codes used to identify objects referred to in the Base Specification itself will be pure numeric values to make them more easily usable in multiple human languages. Codes assigned by users will not have this limitation although they may choose to follow this convention if they wish to distribute their objects in multiple languages.

Object Elements

Object Elements are defined as components of an Object. An Object Element can be compared to a Column or Field of a Table which assigns meaning to a particular set of data. Object Element Values are typically character data or numeric data stored in fields defined as Object Elements. Object Element Values are associated with an Object Element and a specific instance of an Object. Object Element Values are stored in the Table Element identified by Code 31. Object Elements use Object Kind Code 3. A list of the Object Elements included in the Base Specification can be found in Appendix 2. For Object Elements from the Base Specification, the Code shown in Appendix 2 would be populated in Table Element 8. Table Element 7, Object Element ZTIC, would be populated with the ZT Instance Code that represents the source ZT Instance of the Object Element.

Type Definitions

Object Types are used to categorize instances of an Object Kind. The Type of an object is also used to identify objects that should share a certain behavior, i.e. data content, display characteristics and permitted/required linkages to other objects. The possible types for an Object Kind are always validated against the objects of another Object Kind. The Object Kind used for this validation is referred to as the Type Table. The specific instance of an the object in the Type Table is referred to as the Type Value. For example if Object Kind Country serves as the Type Table for Object Kind Address then Mexico could be the Type Value for a specific Address. More specifically the ZT Instance Code and Code for Mexico would make up the Type Value for the address. When a Type Values are maintained for a specific instance of an Object Kind, the server checks that the Type(s) specified exists in the ZT Instance Code and Code elements of the related Type Table(s).

Object Type relationships are configured using Type Definitions. Type Definition is an Object Kind that also uses two separate Type Definitions in it's own definition. The first Type Definition is used to indicate the Object Kind of the type values. The second Type Definition is used to indicate the Object Set to determine valid Type Values. For example a Type Definition could be set up so that Object Kind, Size will act as a Type Table for Object Kind, Shirt. The Type Definition for Shirt Size for a Shirt would need to specify Object Kind, Shirt Size as a Type Value to indicate that Shirt Size will contain the type values. The Type Definition would also specify an Object Set of Shirt Sizes as the Set of possible type values, small, medium, large, extra-large etc.

Type Definitions are use Object Kind Code 9. A list of the Type Definitions can be found in Appendix 3. Type Definitions become active by creating a link from a Technical Profile to the Type Definition. The link makes type relationships possible/required when an object is maintained using an Object Template that utilizes the Technical Profile. Object Templates and Technical Profiles are explained later in this section of the document.

A given Object Kind, for example Country, can serve as the Type Table for multiple Object Kinds. For example, Object Kind address could use country as a type table and also state/province could use country as a type table. Also an object kind can have multiple type tables. For example an Object Kind for cat could have a type table for sex and another Type Table for breed (Persian, Bengal, Siamese etc). To remove ambiguity when referring to an object type the convention of stating the name of the Type Definition followed by the type value separated by a colon is used. For example, one can refer to a cat of type Cat sex:male, a cat of type Cat breed:Persian or a address of type Country:Mexico.

An Object Kind used as a Type Table can in turn have a Type Table of its own to form a hierarchy. As an example, suppose a developer created an Object Kind for living thing kingdoms. In this table he added plants and animals. Now suppose he added an Object Kind for animal classes and added mammals and reptiles. The developer could then specify that the living kingdom table will be the Type Table for the animal class table. When adding an animal class, the user could then be obliged to categorize the animal class as plant or animal. To do this, the user would, for example, assign the Type Value animal to the mammal and reptile. Now suppose the developer creates another Object Kind, this one for animals and specifies that the animal class table is the Type Table of the animals. The user would then be limited to adding animals of type mammal or type reptile. If cat was added as an animal it could be given the type value, mammal. There is an Object Kind called Type Definition Path (Code 56) which is used to record these relationships. A Type Definition Path has links to Type Definitions. In this example a Type Definition Path for living thing kingdom for animals would be linked to 1. Animal Class for and Animal 2. living thing Kingdom for an Animal Class. This can be used to determine that a cat is part of the animal kingdom. This approach could be extended to model the hierarchy used for classifying animals.

There is a specialized case that allows the automatic creation of links based on a type relationship. To do this a Type Definition can also use Link Type as a Type Table (Type Definition Code 24). The Type Value for this Type Definition indicates that it is a Link Type that will be used to create server generated links from the object being used as a type value to the object being typed. This creates essentially a bi-directional link between an object and the object being used as a type value. As an example, suppose that an application contains financial posting line items and accounting document as Object Kinds. The line items uses document as Type Value to indicate the document that the line item relates to. Suppose there is a desire to see all the line items for a particular document. In this case it is useful to have a link from the document (the Type Value) to the line item. The server is required to automatically create the link if the Type Definition is properly configured, specifying the Link Type as a Type Value.

Links

Typically an object will contain a limited number of elements within its own definition. To make useful applications, it is often advantageous to create links between the objects. A Link is used to establish a relationship between one object and another. Links are attributes of an object, not objects themselves. The from side of the link uses standard object identifiers of the object containing the link. The object being linked to is identified by the Link-To Kind ZT Instance Code, Link-To Kind Code, Link-To ZT Instance Code and Link-To Code. These are stored in Table Elements 19 through 22. The links that are allowed for a particular object are based on the Object Template being used to maintain or query the object. Object Templates are explained later in this section of the document.

The Link Value element in Table Element 30 contains a numeric value that can be used to indicate the strength or priority of a link. If an object is linked to several other objects, the Value can be used to indicate which object has first priority. It can also be used as a percentage or to indicate relative weight. Suppose there are links from a person to her favorite songs to be used by an application that plays music. If one song was most favored the link could be given a value of 100. A less favored song would be given a value of 10. If the application was set to random play, the application could be programmed so that the favored song would be played 10 times as often as the less favored song.

Link Types

Link Types along with the Link Target Type contain the rules which determine which links are allowed/required for a particular Technical Profile invoked by an Object Template. Application development requires defining Object Templates which define the allowed relationships between objects. Link Types provide one of the tools used for this purpose. A list of the Link Types from the Base Specification can be found in Appendix 4.

As a Link Type example, suppose a developer is defining an Object Kind for a person and wants to require that the person is linked to an address. A Link Type would be added that says the person must be linked to exactly 1 (minimum 1, maximum 1) address. The Link Type description could be i.e. Link from person to address. If a client application then tries to create a person that is not linked to a an address, the message would be rejected. As another example, suppose that a Link Type is being created for links from a financial document to the line items of the financial document. The minimum and maximum on the source side specifies how many children the source of the link may/must have. This could be minimum of 1 and maximum of 10,000. This indicates that the financial document must have at least 1 but no more than 10,000 line items.

An object that is linked to can in turn be linked to other objects, creating a hierarchy of links. When an object is queried, the server returns the values of that object plus the values of any objects that are linked to and the the values of any objects that those linked objects are linked to and so on until no more objects are linked. The link types that are relevant for the objects selected is determined from the object templates specified for the objects being queried. This will be explained in more detail.

The Link Type uses an Object Set as a type value. The Object Set determines allowed objects on the from side of a new link using the Link Type. Additionally, the Link Type must specify a Link Target Type as a type value.

Link Target Types

The Link Target Type is used to organize a set of potential targets for the links that use a particular Link Type. The minimum and maximum on the target side specifies how many parents the child can have using this link type. In the financial document example, the minimum could be 1 and the maximum could be 1. The line item must have at least 1 parent meaning it cannot exist in isolation (without a parent financial document). The maximum of 1 indicates that it can have only 1 parent financial document. This is to prevent two financial documents from including the same line item. Recall that the min/max on the source side is contained in the Link Type while the min/max on target side is specified in the Link Target Type. In the Base Specification there is a Link Target Type using a Code that corresponds to the Codes used for the Link Types listed in Appendix 4. The Link Types use Object Kind 10 and the Link Target Types use Object Kind Code 11. Each Link Type specifies the corresponding Link Target Type as a Type Value using Type Definition 2.

The Link Target Type uses a Object Set as a type value. The Object Set determines valid objects to be linked to. The Link Target Type is linked to Object Templates that will be applied to objects being linked to. A validation rule applied to the Link Target Type ensures that the Object Templates linked to have one and only one Object Template per Object Kind. This is to remove any ambiguity as to which Object Template should be applied to an Object of a specific Kind that is a target of a link.

Object Templates

Object Templates control access to the attributes of an object, namely the Object Element values, Type values and links. When an Object Template is maintained, links are created from the Object Template to Technical Profiles using Link Type code 1. The Technical Profiles are in turn linked to Object Elements, Type Definitions and Link Types that are invoked when an object is maintained using the Object Template. When an object is maintained or queried, the Object Template to be used is specified. This determines which attributes can be maintained or queried. A given object can utilize multiple Object Templates. The Object Templates can be used to express alternative usages for an object. For example, if the object kind were company, there could be an Object Template for customer and another Object Template for vendor. Each template would contain the attributes relevant to a particular usage via the linked Technical Profiles. It would be possible for a company to be both a customer and a vendor. This could be handled by maintaining the company twice, once with the Object Template for customer and once with the Object Template for vendor. Object Templates use an Object Kind as a type value to indicate which Object Kind the template will be used for. For each Object Kind listed in Appendix 1 the Base Specification has an Object Template with the same code. The descriptions are the same except that instead of ending with (Object Kind) they end with (Base Template). Object Kinds use Object Kind 1 and Object Templates use Object Kind 2.

When a message is processed to maintain an object using a particular Object Template, the ZTIC (ZT instance code) and Code of that template are recorded with the attributes of that object. Any subsequent query of that object would be done with respect to one of the recorded templates.

An Object Template can refer to another Object Template as a reference using Type Definition with Code 30. In this case the referencing Object Template inherits the attributes of the referenced Object Template. For example, there could be a template for Object Kind, company called specialized customer that references the template for generic customer. The specialized customer would inherit all of the attributes of generic customer and possibly add some of its own. A query of generic customers could also select companies that were only maintained using the Object Template for specialized customer because the specialized customer Object Template references the generic customer Object Template.

Technical Profile

Technical Profiles can linked to Object Elements, Type Definitions and Link Types using Link Types with codes 3, 4, and 5, to make them relevant for Object Templates that are linked to the Technical Profile. The Technical Profile can be linked to other Technical Profiles using Link Type with code 2. A Technical Profile that is linked to another Technical Profile inherits the links to the Object Elements, Type Definitions and Link Types of the Technical Profile it is linked to. A Technical Profile can also be linked, using Link Type with code 25, to another Technical Profile to suppress attributes included in the target Technical Profile from the source Technical Profile. Technical Profiles are designed to encourage re-use of the configuration for Object Templates. Configuration that is likely to be shared by multiple Object Templates can grouped into a single Technical Profile and be utilized by multiple Object Templates. For each Object Kind listed in Appendix 1 the Base Specification has a Technical Profile with the same code. The descriptions are the same except that instead of ending with (Object Kind) they end with (Technical Profile). Technical Profiles use Object Kind 12. In addition to the description attribute, each Technical Profile is linked, using Link Type 3 to an corresponding Object Element that hold the description for each Object Kind. For example, Technical Profile 9 is linked to Objecct Element 9 to hold the description for Type Definitions.

Technical Profiles are invoked during message processing based on the Object Template specified in the message for an object to be maintained or queried. Any attribute definition (Object Element, Type Definitions or Link Type) that has been linked to from a Technical Profile that has been linked to from the Object Template being processed, is determined to be relevant. If an attribute definition is determined to be relevant, then the server allows the related attribute (Object Element values, Type values or links) to be maintained or queried. For Type Definitions and Link Types, the other requirement is that the objects referred to are members of the Object Set used as a type value for the Type Definition or Link Type being evaluated. If it is not, the Type Definition or Link Type is ignored and is determined to be not relevant. If a Link Type is determined to be relevant, the rules specified for the Link Type are applied. For example, minimum and maximum number of links must the satisfied for a Link Type. To determine if the rules are being followed in the case of a Link Type, it is necessary to consider not only links specified in the message, but also links that already exist in the database.

Object Sets

A Object Set is used to specify the members of a group of objects out of a possible universe of all objects of all kinds in a ZT instance. It is used, for example, by Link Types and Link Target Types to determine which objects would be valid as the source or target of a link. Sets can also be used for query selection. For each Object Kind listed in Appendix 1 the Base Specification has a Object Set with the same code. These sets contain all of the objects of the corresponding Object Kind. The descriptions are the same except that they all begin with the words "Set of all" and the words (Object Kind) are removed. Object Sets use Object Kind 13.

The members of an Object Set are determined based on links from the Object Set Definition to other Objects using a variety of Link Types. An Object Set that is not linked to any object is the empty set. The possible Link Types to determine Set membership are listed below. The Link Type Codes refer to the list of Link Types in the Base Specification found in Appendix 4.

1. Links from the Set Definition to an Object Kind to include all objects of that Kind as members. This is Link Type Code 12.
2. Links from the Set Definition to individual objects. Any object linked to using this Link Type, Code 13, is included as a member.
3. Links from the Set Definition to a Type Value Based Object Set to include all objects that have a Type Value in the Type Value Based Object Set as members. This is Link Type Code 14. The link to a Type Value Based Object Set can also refer to Type Values that are further up the type hierarchy. For example if there is an object kind for animals including lions, tigers, cheetahs, wolves, bears, fish and lobsters. A type table for animals could include cats, non-cats and sea creatures. A type table for that could include mammals and non-mammals. The Set Definition could be linked to a Type Value Based Object Set that includes the type values mammals. This would cause the animals lions, tigers, cheetahs, wolves and bears to be included as set members.
4. Links from the Set Definition to a Template Set to include all objects using a Template in that Template Set as members. This is Link Type Code 15.
5. Links from the Set Definition to a Link Target Type to include all objects linked to with a link using that Link Target Type are included as members of the Set. This is Link Type Code 16.
6. Links from the Set Definition to a Generic Object Element Value to include all objects containing Object Elements with that value as members. This is Link Type Code 17.
7. Links from the Set Definition to a Set Definition to include the target set members as members of source set using an OR Condition or (Union of Sets). This is Link Type Code 18.
8. Links from the Set Definition to a Set Definition to include target set members as members if also included in source set. This uses an AND Condition (Intersection of Sets) This is Link Type Code 19.
9. Links from the Set Definition to a Set Definition to remove target set members as members of the source set (NOT Condition). This is Link Type Code 20.
10. Links from the Set Definition to a Custom Function to determine which objects will be included as members. This is Link Type Code 21.
11. Links from the Set Definition to a Namespace so that all objects that have an Object Kind with the ZT Namespace linked to are included as member. This is Link Type Code 23.
12. Links from the Set Definition to a Namespace where all objects that have an Object ZT Namespace linked to are included as members. This is Link Type Code 24.
13. Links from the Set Definition to Generic Objects to include the objects the Generic Object refers to as members. This is Link Type Code 30.

When a server determines the members of a set, it must include with each member the timestamp range when the membership is effective. When returning the members of a set the server needs to base this on a specific point in time. Only members that are effective at that point in time are relevant when determining membership. The point in time used for this purpose will generally be the effective time of the top level object being queried or maintained.

The membership specified is additive. For example if a set is linked to some object to include them as members and it is also linked to a type definition so that all objects of that type are included then the set will include members based on all linkages. Any duplication of the members specified would be ignored and each object would

appear as a member only once.

Type Value Based Template Rule

The Type Value Based Template Rule is used to invoke Object Elements, Type Definitions and Link Types for an Object Template based on the Type Value an object has. Objects with Type Values that have similar characteristics can share a Type Value Based Template Rule. Specialized characteristics can also be added using a Type Value Based Template Rule that is unique to a single Type Value. In many cases, the Object Template will be sufficient. The Type Value Based Template Rule is only needed if specialized behavior is needed based on the Type Value of an object. Type Value Based Template Rules use Object Kind Code 5.

To use a Type Value Based Template Rule a link is created from a Type Definition to the Type Value Base Template Rule using Link Type Code 29. The Type Definition linked to the Type Value Based Template Rule is invoked based on a link from a Technical Profile that is linked to from a Object Template being processed. The Type Value Based Template Rule uses an Object Set as a type value with Type Definition 39. The Object Set members are used to specify all the type values that will have the special attributes determined by the Type Value Based Template Rule such as Object Elements, Type Definitions and Link Types. The Type Based Template Rule also uses Object Template as a type value with Type Definition 38. The Object Template used as a type value is linked to the Object Elements, Type Definitions and Link Types that will be invoked if an object is created with a type value that is included in the Object Set of type values used by the Type Value Based Template Rule. Care should be taken to ensure that there are not over-lapping type values in the Object Sets if a Type Definition is linked to multiple Type Value Based Template Rules.

As an example of how the Type Value Based Template Rule could be used, suppose that an object definition is being created for Address. The Object Template for the Address is linked to a Technical Profile which is linked to the Object Elements that will contain values for the address such as city, street and house number etc. The Technical Profile is also linked to a Type Definition that specifies that Country will be a type table for Address. We would also like to include an Object Element for postal code, but different countries use the postal code differently and some countries use no postal code at all. To solve this, a Type Value Based Template Rule can be used. This Type Value Based Template Rule specifies an Object Template as a type value that is linked to an Object Element that has the correct specification for postal code that is used for a set of countries. The Type Definition, Country for Address is linked to several Type Value Based Template Rules. To specify which countries each Type Value Based Template Rule applies to, each Type Value Based Template Rule uses an Object Set that includes the relevant countries as members as a type value. Using this configuration the ZT Server can determine the appropriate postal code Object Element to use for the address within a country if that country uses postal codes.

Link Based Template Rule

The Link Based Template Rule can be used to specify specialized behavior based on which links exist for a Link Type. It follows he same general pattern as the Type Value Based Template Rule. Link Based Template Rules use Object Kind Code 6.

To use a Link Based Template Rule a link is created from a Link Type to the Link Base Template Rule using Link Type Code 33. The Link Type linked to the Link Based Template Rule is invoked based on a link from a Technical Profile that is linked to from a Object Template being processed. The Link Based Template Rule uses a Object Set as a type value using Type Definition Code 44. Object Set specifies all the objects that can be linked to, to invoke the specialized attributes such as Object Elements, Type Definitions and Link Types. The Link Based Template Rule used Object Template as Type Value, using Type Definition 43, which is linked through Technical Profiles to the Object Elements, Type Definitions and Link Types that will be invoked if an object is created that is linked to an object that is included in the Object Set used by the Link Based Template Rule.

Object Element Value Based Template Rule

The Object Element Value Based Template Rule is used to specify specialized behavior based on how an Object Element has been populated. Object Element Value Based Template Rules use Object Kind Code 7. To use an Object Element Value Based Template Rule a link is created from an Object Element to the Object Element Value Based Template Rule using Link Type Code 34. It follows a similar pattern as the Type Value Based Template Rule and the Link Based Template Rule in that it uses Object Template as a type value, using Type Definition Code 45, which through the Technical Profile is linked to Object Elements, Type Definitions and Link Types to be invoked. The Object Element Value Based Template Rule uses an Object Set as a type value, using Type Definition Code 46 and Object Element as a type value using Type Definition Code 47. These two type value determine which objects and which Object Elements in those objects to look for a match with the Object Element using the Object Element Value Template Rule. If a match is found, the Object Element Value Template Rule is invoked. An example usage would be if an Object Element could be configured with a Object Element Value Template Rule so that if a boolean value of the Object Element is set to true, then additional Object Elements are required to be populated when maintaining an object being created using a Template with an Object Element that includes the rule.

Object Element Metadata

Introduction

This section describes the Object Kinds, Links and Type Definitions used to define how Object Element data can be stored on a ZT Server. These are use to validate data in a Request Message.

Object Element

Object elements are used to declare the data components of an Object Kind. Links are created from a Technical Profiles to the Object Elements to determine if the Object Elements are relevant for the Object Templates that use the Technical Profile. The Object Element has two type tables, Object Element Usage Type and Data Element.

Object Element Usage Type

The Object Element Usage type serves as one of the Type Tables for Object Elements. There are two possible values of Object Element Usage Type, Standard and External.

An Object Element of Object Element Usage Type External gets its value from a related type table. There must be a link from the Object Element (of type external) to the Object Element (of type Standard) where the Object Element value will be pulled from. Also External Object Element would use Type Definition Path as a type value which would have a link to the Type Definition used to find the internal object element value. An example of this could the an address that uses country as a type table. The name of the country is part of the address but it is included with the Country Object Kind as an Standard Object Element. To configure this, country name would be added to the address as an external object element. There would be link from this country name object element to the Standard Object Element for country name used by the Country object kind. The External Object Element would use a Type Definition Path as a type value which would be linked to the type definition, country for address.

This same approach can be used to include values that are higher up the type table hierarchy. For example, suppose there was a wish to include Continent name as part of the address. Also suppose that Continent is used as a type table for country and country is used as a type table for address. In this case there would be a link from the

External object element used for continent in the address to the internal object element used for continent in the continent object kind. To indicate the Type Definition path, the External Object Element would use Type Definition Path as a type value which would have links to two Type Definitions 1. a link to the Type Definition that specifies that country is the type table for address and 2. a link to the Type Definition that specifies that continent is the type table for country. So, if an address was created of type Mexico the name of the continent would be found as North America.

-

Data Element

The Data Element Object Kind is used to define attributes that can be applied to Object Elements and serves as one of the type tables for Object Element. The Data Element contains a description and a length. Data Element uses Data Type as a type table. Object Elements that have the same length and similar purpose can use the same Data Element as a type value. Data Elements of type Calculated String use a Function as a Type Value. The Function is used to determine the value of Object Elements that use the Data Element as a type value. The Function can determine the value using object values in the server database or it can also be linked to one or more Parameter Value Groups. The Parameter Value groups are used to determine which parameters should be passed to the function to arrive at the Calculated string. This is done by checking parameter values included in a message being processed. Parameter values that have the same Parameter Value Groups as the ones linked to from the Data Element can be passed to the Function used to arrive at the Object Element value. The Parameter Values are then passed to the Function if the Function uses Parameters with the same ZTIC and Code.

Data Elements can also be used for reporting. For example each Object Kind has a separate Object Element to hold the description of each Object Kind. However each description Object Element refers to the same Data Element. If a report lists all of the objects in a namespace the report could list values from all the Object Elements that use the Data Element used by descriptions

Data Type

The Data Type serves as a type table for the Data Element table. Its values include Numeric, Character, Boolean, Timestamp and Calculated String. It allows the Data Element to specify which basic data type to use for the Data Element and thus which validations to perform. The Data Type uses Function Group as a type value to specify the validation routine to be used for Data Elements that use that Data Type. The validation i.e. for numerics, it would specify that the numeric details table would give the number of decimals etc. The timestamp data type uses a Function that specifies timestamp validation. The result of the Function execution is that if an insert is attempted that violates the validation, the server will return an error with an appropriate message.

Validation of Character Values

Data Elements can be configured to provide validation for Object Elements that use them. A Data Element with a Data Type value of Character can use Object Set as a type value using Type Definition Code 48. The Data Element with a Data Type value of Character can also use Object Element as a type value using Type Definition Code 49. Type Definition 48 and 49 are used together to find the Object and Object Element containing valid character values for an Object Element using the Data Element being configured. For example an Object Set name boolean could be used as a type value with members representing Generic Object Elements with valid character values including the values + for true, - for false and ? for initial or not determined value in the Object Element for Generic Object Element value. Any Object Element using this Data Element would be limited to the values +, - or ?.

Numeric Data Type Details

The Numeric Data Type Details, Object Kind Code 66, has the detailed specifications for the numeric fixed length data type. It serves as a type table for Data Elements that use a Data Type of type Numeric. It includes elements for minimum value, maximum value, smallest increment (step value) and start value. The smallest increment is used to indicate the level of precision that the Data Element will permit. As an example of how these are used, if the minimum value is 0, then negative values are not allowed. If the start value is 0, the minimum in 0 and the smallest increment is 2 it only allows positive even integers and 0. If the start value is 1 and the smallest increment is 2 it allows, odd numbers. To determine if a numeric value is valid, first a check is done to make sure it is larger or equal to the minimum value and smaller or equal to the maximum value. Then the start value is subtracted from the value to be validated and this result is divided by the by the smallest increment. If the result is an integer, then the number passes the validation test.

Extended Key Definition

Extended Key Definitions are used to configure Object Elements that are accessed using key values beyond the Object Code ZTIC and Object Code. For example language (human) has an Extended Key Definition as part of the Base Specification. The base template for Extended Key Definition for language uses a Type Definition where Language is specified as the Type Table and the Set of all languages are allowed values. Fiscal period is an other example of a possible Extended Key. An Object Element configured to use language or fiscal period as an Extended Key could only be accessed if the appropriate key was specified in the message.

Using of Extended Keys begins with configuring an Extended Key Definition, with Object Kind Code 33, or using an Extended Key Definition from the base standard such as language. The Extended Key Definition uses Object Kind as a type table to indicate the valid keys for the Extended Key Definition. For Example, the Extended Key Definition for language uses Object Kind Language, Object Kind Code 18 as a type table. This means that the Extended Key Definition for language will use keys from the language table as possible key values. The Extended Key Definition is invoked when a Data Element uses the Extended Key Definition as a type value. For example, if one wants an Object Element to be language dependent, the Data Element used by that Object Element must specify the Language Extended Key Definition as a type value.

To populate an Object Element using an extended key, its necessary to specify the relevant keys in the message built by the client. This is done in the header section of the message in the elements under Message Element 231. Message Elements are explained later and are shown in Appendix 6. Each key must specify the ZTIC and Code of the Extended Key Definition (in Message Elements 2311 and 2312) and the ZTIC and Code Extended Key value (in Message Elements 2313 and 2314). In the case of an Extended Key for language, the the ZTIC for Extended Key Definition would be the Code associated with the base specification. The Code would be 2 which is the Code for the Language Extended Key Definition. The Extended Key value ZTIC would be the code associated with ISO units and the Code would be the correct value for the language being used, for example English, Spanish etc.

When Object Element data is passed in a message for Object Elements that use extended keys, either to maintain an object or to query object data, the same Message Elements are used as are used for Object Elements that do not use extended keys. That is Message Element 3044 to maintain an object and Message Element 4144 for a query response. The difference comes when the Object Element values are stored in the database. In this case, Table Elements 9 through 12 are used to store and retrieve the, Extended Key Definition ZTIC, Extended Key Definition Code, Extended Key Value ZTIC and Extended Key Value Code. In this way, data that is stored with a certain extended key, for example language code, will only be retrieved when that extended key is provided in the Header of the message used to retrieve that data.

There may be cases where it is useful to have an Object Element that depends on more than one Extended Key. An example of this could be an Object Element that depends on both language and version. Or perhaps the description for a financial application that depends on both language and fiscal period. In this case the Object

Element Definition still uses Extended Key Definition as a type value. The Extended Key Definition of type Composite must use an Object Kind of Type Composite Key as a type value. This Object Kind must have a template that allows two or more type values so that an object of this kind could have a type value for a language and also a type value for version. If a message is being processed to store Object Element data for an Object Element that is configured to use this composite Extended Key, the server creates a new object of the kind that will hold the composite keys. It adds type values to the new object to specify, for example a certain language and a certain version and puts the new object ZTIC and code in Table ELements 11 and 12. When data is retrieved, the server looks at the configuration of the Object Element being retrieved, finds that it uses a composite key and the related objects that hold the keys. It checks if any of those objects have type values that match the extended keys provided in the query message. If there is a match, then it now has the composite key value to retrieve the data. Note that the composite key values are internal to a server and are not shared in any query response. Two servers could have different composite keys to represent the same combination of type values.

Additional Features

ZT Instances

The ZT Instance represents a distinct occurrence of an environment for storing data in a ZT Server. A ZT Client specifies the target ZT Instance when sending a message to the Server. A user, logging in through a Client, would typically interact with only one ZT Instance per login session. The application data for the two ZT Instances would be completely separated.

The ZT Instance Code (ZTIC) elements are used to identify a ZT Instance and indicate the source of an object within a ZT Instance. They are included so that objects from different sources can have the same Kind Code or Object Code value and still have unique identifiers. Each ZT Instance represents its own environment for assigning ZT Instance codes. Also each message will contain its own environment for assigning codes to ZT instances. The ZT Instance Code used to represent the local instance within a ZT Instance can have any value but by convention will use the value 1 (the first ZT instance to be declared within a new ZT instance). ZT instance code 2 by convention represents objects of the Base Specification. Other ZT Instance Codes can have any other value. Within each environment the ZT instance Codes will be associated with a namespace that must NOT be changed after it is assigned and used to send data to other systems.

ZT Instance Codes (ZTIC) are unique only within the context of the ZT Instance or within a message. Namespaces must be universally unique to avoid naming conflicts when data is shared among ZT instances. When a message is processed, the server must take care of converting from the ZT Instance Codes within the message to the ZT Instance Codes used in the target server by using the namespaces associated with the ZT Instance Codes. A namespace is designed to be a set of characters that is unique from any other namespace. To ensure uniqueness it is generally recommended to include an Internet domain name in the namespace. An example namespace could be my_email_address@a_domain_name.com/my_test_system1.

When two ZT Instances share the same Server database tables, they are differentiated using the Server DB--ZT Instance ID, Table Element 32. Each Server DB--ZT Instance ID is associated with exactly one namespace. The Server DB--ZT Instance ID does not need to be universally unique, only unique within a Server database. For example, two servers could use the Server DB--ZT Instance ID called TEST, but they would be differentiated by using unique namespaces, for example dev1.com/test_env and dev2.com/testing_place. The calling client only needs to specify the namespace of the target ZT Instance in the message and there is no particular need for the client to be aware of the Server DB--Instance ID. The message and message response only refers to namespaces and related ZT Instance Codes. The ZT Instance Codes along with the namespaces make it possible to distribute data from one ZT Instance to another without naming collisions. The following example illustrates the usage of ZT Instance Codes.

This example shows how data from several ZT Instances can be used to enrich an application. For this example, suppose a developer wants to create a ZT Object Kind for storing food data with some useful attributes such as name and calories per kilogram. The ZT Instance that the developer is working in is associated with namespace dev1.com and ZT instance code 1 (the local instance). The developer creates an Object Kind for food and creates the foods Bread and Lettuce. The developer also creates an Object Kind for Food Type and adds an entry for Human Food. Table entries are shown below. Some elements are omitted to simplify the example. Also, language dependent and language independent rows are presented as a composite in the same table row. In actual practice they could be stored in separate tables. In the composite

views, language dependent entries that appear for multiple languages have repeated rows.

Kind table

Code ZTIC	Code	Language	Description
1	11	english	Food
1	111	english	Food Type

Food Type table

Code ZTIC	Code	Language	Description
1	T1	english	Human food

ZT Instance table

Code ZTIC	Code	Language	Namespace	Description
1	1	english	dev1.com	Namespace for developer 1
1	2	english	131131/21	Namespace for ZT Base Specification

Food table

Code ZTIC	Code	Language	Type ZTIC	Type	KCAL	Description
1	1	english	1	T1	60	Bread
1	2	english	1	T1	30	Lettuce

A second developer, developer #2, imports the objects from developer #1 and adds some Food Types, for cat and dog foods. Notice that the ZT Instance code (ZTIC) for the objects imported from developer #1 is different than it was on the first server (101 instead of 1), but the code still represents the same namespace (dev1.com). The new Food Types are created under the local ZT Instance labeled 1. Developer #2 adds foods for humans, cats and dogs. He also translates the object kind to Spanish. The Code and Code ZTIC for the Spanish translation in the Kind table are the same as the English name. They both have the Code ZTIC 101 to represent the namespace of the original developer (dev1.com). The Modifier ZT Instance Code indicates which ZTIC a change to an object originated from. The MZTIC (modifier ZTIC) is 1 to represent

dev2.com because developer #2 modified an object that was created by developer #1.

Kind table

Code ZTIC	Code	Language	Modifier ZTIC	Description
101	11	english		Food
101	11	spanish	1	Comida
101	111	english		Food Type

Food Type table

Code ZTIC	Code	Language	Modifier ZTIC	Description
101	T1	english		Human food
1	T1	english		Cat food
1	T2	english		Dog food

ZT Instance table

Code ZTIC	Code	Language	Modifier ZTIC	Description	Namespace
1	1	english		Namespace for developer 2	dev2.com
1	2	english		Naemspace for ZT Base Specification	131131/21
1	101	english		Namespace for developer 1	dev1.com

Food table

Code ZTIC	Code	Languqge	Type DSIC	Type	Modifier ZTIC	Desccription
101	1	english	101	T1		Bread
101	2	english	101	T1		Lettuce
1	1	english	101	T1		Jelly
1	2	english	101	T1		Peanut butter
1	3	english	101	T1		Apple
1	4	english	101	T1		Orange juice
1	5	english	1	T1		Kitty Chow
1	6	english	1	T1		Kitten Chow
1	7	english	1	T2		Doggie Chow

Note: The food table has duplicate codes in it, but these are differentiated by the ZT Instance Codes (ZTIC). Also the Food Type T1 is used to represent human food (from developer #1) and cat food (from developer #2). These are differentiated by the Type Value ZTIC, 101 for developer #1 and 1 for developer #2.

A third developer, developer #3, receives the objects from developer #2 and imports them into a 3rd ZT Instance and adds a Food Type for bird food, the foods Bird Seed, Bird Feed, Rice, Tofu and Beans under the ZT Instance Code 1. Each of these foods is given its code sequentially starting from 1. The table entries for developer #3 would

appear as follows.

Kind table

Code ZTIC	Code	Language	Modifier ZTIC	Description
101	11	english		Food
101	11	spanish	102	Comida

Food Type table

Code ZTIC	Code	Language	Modifier ZTIC	Description
101	T1	english		Human food
102	T1	english		Cat food
102	T2	english		Dog food
1	T1	english		Bird food

ZT Instance table

Code ZTIC	Code	Language	Modifier ZTIC	Description	Namespace
1	1	english		Namespace for developer 3	dev3.com
1	2	english		Namespace for ZT Base Specification	131131/21
1	101	english		Namespace for developer 1	dev1.com
1	102	english		Namespace for developer 2	dev2.com

Food table

Code ZTIC	Code	Language	Type ZTIC	Type	Modifier ZTIC	Description
101	1	english	101	T1		Bread
101	2	english	101	T1		Lettuce
102	1	english	101	T1		Jelly
102	2	english	101	T1		Peanut butter
102	3	english	101	T1		Apple
102	4	english	101	T1		Orange juice
102	5	english	101	T1		Kitty Chow
102	6	english	101	T1		Kitten Chow
102	7	english	101	T2		Doggie Chow
1	1	english	1	T1		Bird seed
1	2	english	1	T1		Bird feed
1	1	english	101	T1		Rice
1	2	english	101	T1		Tofu
1	3	english	101	T1		Beans

This illustrates how objects from different sources can maintain their uniqueness in the same table. The actual namespaces could be based on a URL to guarantee uniqueness across ZT instances. The ZT Instance table provides cross-reference between the ZT Instance code and the actual namespace. Records in other tables in the database (other than the ZT Instance table) use the ZT Instance Code (ZTIC) instead of the actual namespace. When a message is sent between client and server it includes a cross reference between the namespace and the Code used to represent it in the message. When the server receives a message requesting an insert containing a namespace, it checks the ZT Instance table to see if the namespace exists. If it does not exist, the new namespace can be added along with a cross-reference to its namespace. The server finds the ZT Instance code that represents the namespace and uses that to store the record. Two ZT servers could use the same code internally to represent different namespace. This is not a problem because any time the data is converted to a message to be used externally, the namespace in its appropriate long form is declared in the message.

The pattern described here could be extended to enrich the application further. For example, a fourth developer could decide to build an application for handling recipes that includes the possibility of creating links to the foods depicted here as ingredients. It would also need to include quantities, units of measure and cooking instructions. A fifth developer could then use the recipe application to build a recipe book that allows the possibility for the recipe book to create links to recipes. It was mentioned that

developers can distribute their work to other developers. To achieve this, is is necessary to have a client application that queries a ZT Instance and maps the Response to the Request of a new message. The new message can then be imported by others.

Note: There is significant reuse of constructs within the specification. This tends to blur the distinctions between system definition, application development, configuration, master data and transactional data. The use of separate ZT Instances can help clarify this. For example, development and configuration could occur in a separate ZT Instances (ZTI). When development and configuration is completed it can be transferred to another ZTI for testing and/or transactional postings. The transferred objects retain the namespace of the source ZTI where they were created. This makes them identifiable if they need to be removed.

Timestamp Implementation

The effective timestamp element is used to indicate the time when a record becomes effective. The basis for timestamps is the Modified Julian Day which treats November 17, 1858, Midnight UTC, on the Gregorian calendar as the reference point. This is assigned a timestamp value of 0. The timestamp indicates the number of seconds or decimal fractions of a second elapsed since that reference point. A negative number indicates the number of seconds or decimal fractions of a second before that point. The base specification does not specify the number of decimal places that are permitted. This is included in the General settings specification and could vary by implementation.

Timestamps can generally be compared by subtracting one from another. However, when including time periods after leap seconds were introduced, the result would need to be adjusted to account for this when converting to conventional date/time representations. Leap Seconds are maintained in a Leap Second table that is not included with the Base Specification. The Leap Second object includes an effective timestamp indicating when the leap second should to added or subtracted to aid in timestamp calculation.

Under normal operations records in the database are not changed or deleted. If the value for an object is changed, a new record is inserted with the new values, with a new timestamp to indicate when the record was inserted. The old records would be retained. By keeping the old records, it is possible to provide a change history if requested by the client. The records are created under the code ZTI Code that is local to the data environment. The timestamp--effective would usually be the time stamp (as defined above) of the local server at the time the record is inserted. If the record was originally created on another server, the time stamp would usually be the timestamp time on that other server (as defined above) when the record was inserted on that other server. If records are imported at various times from another server, using the timestamp from the other server as the effective time will cause the records to retain their proper order of creation when they are imported.

There are opportunities for the client to depart from the convention of using the time the records are inserted as the effective time. Suppose the records are for prices that are not yet effective. By inserting the records with a future effective date/time, the new prices will not take effect until that time arrives. The server should reject effective timestamps that are in the past, unless they were created on an external system. Current timestamps are assigned by the server, not the client.

The sequence number, stored in Table Element 27, will usually have a value of 0000. If there is a situation where there already exists a record with the same Kind, Kind ZTI Code, Code, Code ZTI Code, and timestamp--effective and it is necessary to make the new record unique in the database, the sequence number would be incremented by 1. The next record would be 0001, then 0002 etc. Because the timestamp may represent fractions of a second, this will usually be enough to make the new record unique. For this reason it is not likely that the sequence number will need to be incremented.

Another situation that would require use of sequence number other than 0000 relates to effective timestamps that are in the future. Suppose objects are added containing prices that are effective in the future. At a later time, it is decided to change the price, but the effective time remains the same. In this case, the sequence number could be incremented from 0000 to 0001. The price with the 0001 sequence number would take precedence over the price with the 0000 sequence number. Also, to make a record that will not become active (through the passage of time), add a record with the same key as an active record, but increment the sequence number and give the new record an inactive status.

Rationale and Approach to Versioning

As an object is maintained through time, the ZT Server retains the various versions of the object attributes. The standard key includes a time stamp, in Table Element 26, which is used to maintain the multiple versions. To illustrate usage of the timestamp, assume an example sales order application. Typically a sales order application would use master data, for example, the customer master and item master. These are used to build the transactional data, the sales order. The example assumes a simplified set of

master data and a sales order shown below.

Customer Master

Customer Code	Customer Name
1	ABC Co.

Item Master

Item Code	Item Description
10	Bananas

Sales Order

Sales Order Number	Customer Code	Item Code	Quantity	UOM
101	1	10	5	KG

The sales order shown includes only one line-item. The master data objects are referred to in the sales order using their key, 1 for customer (ABC Co.) and 10 for item master (Bananas).

The difficulty arises when there is a change to the master data. Suppose the customer changed their name from ABC Co. to DEF Co. and this is changed in the customer master. If someone displays the sales order, it will show that the Sale was from DEF Co. even though it was originally created for ABC Co. This may be okay in some cases and in others may cause confusion.

One way to remedy this problem is to copy the customer name into the sales order. Since the item master has a similar problem, we could also copy in the item description as below.

Sales Order Number	Customer Code	Customer Name	Item Code	Item Description	Quantity	UOM
101	1	ABC Co.	10	Bananas	5	KG

Of course, There is more in the customer master than the customer name. Any of these elements such as customer telephone number and customer address and customer contact person could change over time. To have a true picture of what these values were at the time the order was created, it would be necessary to copy all of the master data into the sales order. To ensure a consistent truth throughout the system it would also be necessary to protect that master data values in the sales orders from changes. Generally applications take a hybrid approach, copying in some elements from the master data into the transactional data and accepting a level of data inconsistency across the system.

The ZT server uses a key with a timestamp to record when a value became effective. Links are created from objects to other objects so they can relate to the correct version of the object linked to at the time the link was created. For example, when the customer name changes, this is reflected with a second record with the new name

and a timestamp that indicates when the change is effective.

Customer Master

Customer Code	Timestamp	Customer Name
1	2021-05-01-15:00:00	ABC Co.
1	2021-09-15-16:00:00	DEF Co.

Item Master

Item Code	Timestamp	Item Description	
10	2020-03-15-11:00:00	Bananas	

Units of Measure

UOM Code	Timestamp	Item Description	
KG	2019-02-15-11:00:00	Kilogram	
LB	2019-03-15-11:00:00	Pound	

When the sales order is created, it also has a timestamp.

Sales Order Number	Timestamp	Quantity
101	2021-07-25-09:00:00	5

By using timestamps with links and Type Values, the ZT Server can access the version of the master data that existed at the time the sales order was created, even if the master data subsequently changed. This is a simplified example. A Sales Order design would more likely contain a header linked to line items with other objects related by some combination of Links and Type Values.

Code Assignment

When new objects are created using the namespace that is local to ZT Instance where they are being created, it will typically be necessary for the ZT Server to handle generation of new codes to identify the objects. To achieve this, there is some configuration to be performed. The Object Kind refers to an object called Code Range (Object Kind Code 21) as a Type Value. In addition the Code Range must refer to an Object Kind called Code Scheme (Object Kind Code 22) as a Type Value. Code Range table includes elements called Next Code, Numeric Preferred, Minimum Code Value, Maximum Code Value and Maximum Code Length. When an entry is added to the Object Kind table, the Next Code field for its Code Range must be initialized with a value that contains at least one character and cannot be all zeros and spaces. It must be left justified (no leading spaces) and contain no embedded spaces. It also cannot be longer than the Maximum Code Length specified. The Next Code is used to

store the next code value to be allocated if an object an Object Kind using the Code Range is created. After a code is allocated, the code is incremented and stored in the Next Code element.

The Code Scheme can be linked to a Function that specifies how the code is incremented. The default Function for incrementing is as follows: If the next code element in the Code Range contains a pure numeric value and the numeric preferred element is set to true, the next code is found by adding 1 to that numeric value. The pure numeric must include characters from the set 0,1,2,3,4,5,6,7,8 and 9 (with no decimal point no plus sign and no minus sign). Leading zeros, if present, are retained (0003 is incremented to 0004). This continues until the maximum value is reached or until the code is all 9s and its length equals the Code maximum length.

If the numeric preferred is not set to true or if all numeric values have been exhausted, the following Function applies by default. There is a mandatory link from the relevant Code Scheme to a Resource that refers to a text file containing a string of non-repeating characters with no embedded spaces for example 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ$@. When a code needs to be incremented the server takes the following steps: Evaluate the right-most (non-space) position in the next code element and search the text string for this value. If the value is not found change the right-most position of the next code element to the first value in the text string. If the value is found in the text string, change the right-most position of the next code element to the next value in the text string after the one found. If the value found in the text string is the last value in the text string, change the right-most value in the next code element to the first character in the text string and then evaluate the character one position to the left in the next code element and change it using the same rules as the right-most position. If there is no position to the left in the next code element, create it and give it a value from the first character in the text string. Exception: if the first element in the text string is 0 then use the next position in the text string. Continue this process until the next code would exceed the length specified in the code length element. At this point an error is raised: Next code is longer than allowed length. If a new code is requested and the next code element already equals the maximum value then an error is raised

The codes assigned are only relevant for objects created locally (under the local ZTI) when the codes are assigned by the server. Objects from foreign ZTIs would already have codes that were assigned on the foreign system and would be retained when imported. Codes can also be assigned by the client if this is allowed for objects of a given kind and it is specified in the request message. Code from external sources must also follow the rule that they are left-justified (no leading space), have no embedded spaces and cannot contain only zeros or only a ?.

Note: if codes with leading zeros are desired, the next code element should be initialized with leading zeroes, for example 0001 with a maximum length of 4. If pure numeric codes are desired, the maximum code should be set to a numeric value (i.e. all 9s). If the maximum value is longer than the maximum length, it will have no effect. The maximum length will take precedence.

Note: if it is desired to begin with pure numerics then use pure alpha, then specify numeric preferred and in the text file for code sequences specify the codes ABCDEFGHIJKLMNOPQRSTUVWXYZ.

Note: The base specification does not dictate the length of the code field. An implementer could choose to use a short code length even replacing the code values for the standard entries with corresponding values from another coding scheme. For example it could be a code scheme with a large character set. The length of the code can have a large impact on database size. This is because the code length is also becomes the length for the ZTI Code fields. Of course, if the implementer chooses a very small maximum code length, it will not be able to interoperate with other systems that are sending objects with a longer code length. To maintain interoperability, separate specifications (that are yet to be developed) that are targeted to particular usages can include a maximum code length.

Server Definition

A ZT Server is a system that can process messages from a ZT Client. The Server is responsible for handling updates to the back-end database. One aspect of this is maintaining of Sever DB--ZT Instance IDs. Server DB--ZT Instance IDs are used to create a separate data environment designated for maintaining data for a specific ZT Instance. Server DB--ZT Instances are populated in the Table Element identified with Code 32. This can be populated as an element within a table or it can be embedded in the name of the table itself. The Server DB--ZT Instance ID is associated with a unique namespace that identifies a ZT Instance to the outside world. The relationship between the Server DB--ZT Instance ID and its namespace should be maintained by the server in a configuration file or configuration table that is accessed when the ZT Server is started. The relationship between the Server DB--ZT Instance ID and its namespace should not be changed after it is established, especially if data from that ZT Instance has been distributed to other systems. A message from a Client identifies the target ZT Instance in Message Element 223 which has the Receiver ZT Instance Code. The namespace for the Receiver ZT Instance code is found by looking for the namespace in Message Element 2101 that is associated with the ZT Instance code in Message Element 2100. Based on the namespace of the target (receiver) instance, the server can determine the Server DB--ZT Instance ID and use it to populate any data being updated.

Function Groups and Validation

Function Groups are used to organize a set of functionality that can be called in a generic manner. Function Groups can be linked to one or more Functions that are to be executed and one or more System Messages that provide the boilerplate text to be added as log messages. The Function Group calls each Function and possibly replaces portions of boilerplate system messages with variable values. One important use for Function Groups is for validations. The Function Group can be used to analyze an incoming message, do a validation check, add appropriate log messages and set the status of message, for example to indicate that the validation failed.

The Function Group (Object Kind Code 52) is linked to System Messages (Object Kind Code 24) using Link Type Code 31 and Functions (Object Kind Code 36) using Link Type 32. The function is linked to a set of parameters (using Link Type Code 8). The Function Group handles populating parameters that will be used by the functions. The Function Group includes an Object Element that can be populated with a start parameter. This can be used in a variety of ways including being used to determine the choreography of Functions to be called. By default each function is called once based on the link value of the links from the Function Group to the Functions with the links with the high values being called first.

An example of how a Function Group can be used is the Function Group with code 2 in the validation namespace, outside of the base specification. This is used to validate that that if an object is being maintained with links to existing objects that the objects being linked to actually exist in the database. Function with Code 2, in the validation namespace was created to perform the validation. System message with Code 2, in the validation namespace was added to provide a message that can be logged if the validation fails.

Statistical Values

Statistical Values are used by the ZT Server to provide summarized object data to a ZT Client. Suppose, for example, that the ZT Server contains a large number of financial postings and the user would like to query a summarized version of the postings, not retrieve all of the individual postings. Suppose also, the requirement is to display the postings in a spreadsheet workbook. The postings for a each year are to be displayed in separate spreadsheet tabs in the workbook. Each spreadsheet needs to have twelve columns, one for each Gregorian month of the year and a row for each account.

Statistical Values can be implemented using a Statistical Values Object. The Statistical Values Object refers to an Object Element that contains the data to be summarized and Dimensions used to determine how the data is to be summarized. In the example of summarized financial postings, the Statistical Values Object would refer to the Object Element containing the financial posting amount and to several dimensions used to accumulate the postings into appropriate buckets. The first Dimension could be for general ledger account, the second could be for month and a third Dimension for year. Each of the Dimensions would refer to another object called Type Definition Path. The Type Definition Path indicates which Type Definitions in the hierarchy of Type Definitions would be used to arrive at the object used for summarization. A

financial posting could use general ledger account directly as a Type Value. In this case, the Type Definition Path would contain only one Type Definition, general ledger account for a financial posting. The Dimension for Month would use a Type Definition Path that refers to four Type Definitions, posting date (timezone dependent) for a financial posting, day (timezone independent) for a date, month and year for a day and month for a month a year. The third dimension would refer to a Type Definition Path that includes date for a financial posting, day for a date, month and year for a day and year for a month and year. For each financial posting in the specified set of financial postings, the ZT Server finds the value for account, month and year by traversing the type values for the posting through the Type Value hierarchies. Any postings that share the same general account, month and year would be accumulated into the same statistical bucket.

The objects configured are all in the base namespace. The Object Kind for Statistical Values uses Code 58. The Object Element that contains the data to be summarized is referred to as a Type Value of the Statistical Values Object using Type Definition with Code 37. The Statistical Values Object refers to a Set of objects to be summarized using Type Definition with code 33. The Statistical Value Dimensions Object Kind use Code 59. The Statistical Values Object can be linked to one or more Dimension using Link Type with Code 26. Each Dimension refers to a Type Definition Path using Type Definition with code 34. The Type Definition Paths are linked to the Type Definitions included in the Type Definition Path using Link Type with code 27.

When a Statistical Values Object is included in the response to a Query, the server also calculates the summarized values and places them in the Query Response section of the message in Message Elements that are children of Message Element with code 4180. Message Elements 41810 and 41811 contain the Statistical Values Object ZTIC and Statistical Values Object code. Message Elements 41812 and 41813 contain the Statistic Type ZTIC and Code that was configured for the Statistical Values Object. Options include Code 1 for sum, Code 2 for count, Code 3 for average and Code 4 for standard deviation in the base namespace. Message Elements 41814 and 41815 contain the ZTIC and Code of the Set that specifies the objects that contain the values to be summarized. Message Elements 41816 and 41817 contain the ZTIC and Code of the Unit of Measure related to the values being summarized. Message Elements 41820 and 41821 contain the ZTIC and Code of the Object Element that contains the numeric data to be summarized and Message Element 41822 contains the label of the Statistical Values Object. The details for Dimensions, Statistical Values and Groups appear under Message Element 4183, 4185 and 4187 respectively.

The Dimensions relevant to a Statistical Values Object Response are included under Message Element 4183 and values for an individual Dimension are under Message Element 41830. The ZTIC and Code for the Dimension are in Message Elements 41831 and 41832. The Dimension label is in 41833. Message Element 41834 has the index in related Type Definition Path to indicate the Type Definition used for summarization. The Type Definition Path can be viewed as an array of Type Definitions with array indexes that begin with 0 and continue sequentially using positive integers. Index 0 of the Type Definition Path refers to the Type Definition that is used immediately by the object being summarized. For example for a financial posting, the account could be one of its type values. This would make it a Type Definition with index 0 in the Type Definition Path. In this case, the index in Message Element 41834 would be 0 for the Statistical Values Dimension used for account. The Dimension for Month has a larger array of Type Definitions because Month is not directly and type value of financial posing. To find the month in Type Definition Path is is necessary to navigate through the Type Definitions in the path. For example, financial posting uses date as a type value (index 0). Date uses day as a type value (index 1). Day uses month and year as a type value (index 2) and month and year uses month as a type value (index 3). In this case index in Message Element 41834 would be 3 to allow summarization at month level. Message Element 41835 contains an ID used for grouping. The client may also want the summarized data to be grouped. For example it may be important to group all income accounts together and put all expense accounts in a different group. This could be based on a type value for account such as account category. The account category would indicate if the account is an income account or expense account etc.

The actual values for the Statistical Values Object are listed under Message Element 4185. Message Element 41851 holds Statistical Value ID and 41852 has the Statistical Value. In this example the Statistical Value would be the sum of financial postings that are members of the Object Set configured for the Statistical Values Object for a particular account, month and year. The details of the Type Values used to characterize the Statistical Value are listed under Message Element 4186. This includes the ZTIC and Code for the Type Definition and the Type Values that the Statistical Value relates to. For this example there would be three occurrences of Message Element 41860, Statistical Value Type Value, one for account, one for month and one for year. Message Element 41865 through 41868 are used for grouping, sorting and labeling of the Statistical Value.

Message Element 4187 and its children address the definition of groupings for Statistical Values. The grouping ID in message element 41881 identifies a particular grouping and is referred to in message elements 41867, 41868 and 41869. Message elements 41886 and 41887 are used to hold the ZTIC and Code of the type value used for grouping, in the financial postings example it could refer to an account category to indicate if the account is an income statement or an expense account. Message Element 41885 holds the index in the Type Definition Path where the grouping occurs. Statistical Values that share the same grouping ID can be grouped together when it is displayed by the client application.

Response Types

Response Types are used to control the content of the response returned by the server. For example, when an Object Set is queried or an object that is linked to an Object Set is queried or an object that uses an Object Set as a Type Value is queried, by default the server returns the members of that set in the message response. There may be cases where the set is too large to return in one response. By setting Query Response Control Parameters, the client can control how many set members are returned.

The Query Response Control Parameters are under message element 3107. The Response Type ZTIC and Code are in Message Elements 31071 and 31072. The base specification includes Response Type codes: 1 for standard, 2 for Template of a Linkable Object, 3 for permitted type values, 4 for permitted link-to values, 5 for object set members for sets that are linked to and 6 for object set members for sets that are type values of queried objects.

The ZTIC and Code for Response Types specified in the request in Message Element 31051 and 31052 are referred to in the query response in Message Elements 41051 and 41052. Message Element 31053 through 31056 in the request refer to a related object based on the Response Type. For example, if the the Response Type is 6, for object set members for sets that are type values of queried objects then Message Elements 31053 through 31056 would refer to the Object Set. Message Element 41053 through 41056 in the response refer of the related object based on the Response Type. The values in Message Elements 31051 through 31056 are used to determine if the response control parameters are invoked. These message elements can contain the typical ZTIC and Code values to represent the response type and the related object. They can also contain a * to indicate that it represents all values for ZTIC and Code to refer to a more generalized set of query response control parameters.

The response control parameters values are in Message Elements 31060 through 31065. Message Element 31060 has the maximum number of objects returned. Message Element 31061 has the maximum number of hierarchy levels returned. Message Element 31062 has the offset from the beginning object. For example if a set contains 1,000 members and the maximum number of set members to return is 100, if the offset was set to 300 then the server would return members 301 to 400. Message Elements 31063 and 31064 hold the ZTIC and Code of a sort object to determine which order the objects would be returned in. Message Element 31065 has the set member response mode. Mode 1 indicates, return minimal data content for a set member, mode 2 indicates maximum data content.

It was mentioned that using * for ZTIC or Code allows for a generalized application of Query Response Control Parameters. When evaluating a set of Query Response Control Parameters the server looks at the most generalized values first then looks at parameters that specify a more specific selection. If a more specific selection the matches the scenario being process, then the more specific parameters are used to overwrite the generalized parameters. If the specific selection does not match exactly the scenario being processed, it is ignored. As an example, suppose a query selection contains an object that uses an Object Set as a Type Value. This would match Response Type 6, for object set members for sets that are type values of queried objects. If there was a specific Query Response Control Parameter for Response Type 6 and a related object referring to that specific set, then parameter values such as maximum number of members returned would be invoked, if there was not a match the parameters would be ignored. When multiple Query Response Control Parameters have * for key values, these are the rules for determining which parameters are more generalized. The Message Elements that refer to the Response Type are more generalized than the Message Elements that refer to the related object. Message Elements that refer to the Object Kind are more generalized than the Message Elements that refer to the specific object of that kind. Finally, Message Elements that refer to the ZTIC are more generalized that message elements that refer to a Code.

Message Processing

Introduction

The primary task of a ZT Server is to receive messages from a client, validate that the messages are properly formatted, interpret the messages, maintain or query object data and return appropriate responses. This section describes the syntax of a standard message and the tasks performed by a ZT Server to process the message.

Message Definition

Introduction

The Message Definition describes the structure of messages transferred between the client and the server. The complete definition is shown in Appendix 6 which contains the Message Element Codes and Message Element Descriptions referred to in this section of the specification. The hierarchical relationship of the Message Elements in Appendix 6 is shown by the level of indentation for each Message Element. The Message Definition is divided into three major sections, Header, Request and Response which appear under the Message Root. The Message Definition contains numeric Message Element Codes that are referred to by the message. The Root Message Element Code is 1. the Header Message Element Code is 2. The Request Message Element Code is 3 and the Response Message Element Code is 4. The Message Element Codes are constructed to provide a rough navigation aid. The Codes for all of the children Message Elements share the the same beginning codes as the parent Message Element. Further down the hierarchy this pattern is not followed in all cases.

Header

The Message Header contains data needed to control processing of the message. It is populated primarily by the Client as part of the Request but the Server also updates some Message Elements during the Response. The first section of the Header is called MessageDefinition which is used to refer to the Message Definition that is used to validate the syntax of the message. As this implies, it is possible for the Message to refer to a non-standard Message Definition. However, any alternate Message Definition must still contain the standard Message Elements for the Message Definition and the Message Elements in following standard section called ZTISet.

The ZTISet section, Message Element 21, is used to declare all of the ZT Instance Codes that are used in the Message Request or Response and the associated Namespaces. It must contain the namespace 131131/22 and its associated ZT Instance Code to indicate the ZT Instance Code of the Message Definition. The ZT Instance Codes declared here are local to the message. The Server will maintain a different set of ZT Instance Codes and namespaces. The Codes need to be converted during message processing.

The Message Processing Parameters, Message Element 23, is used by the client to transfer instructions for how the message will be processed including default values described in the Message Definition. The ExtendedKeySet, Message Element 231, allows the client to indicate which Extended Key value to be used to access Object Elements that use extended keys, for example language dependent Object Elements. Message Elements 233 is used queue processing if queue processing (processing of messages in a specific order) is supported by the server.

Request MaintainSet

The Request MaintainSet, Message Element Code 30, is populated by the Client as part of a Request Message and it contains data needed to create or update objects on the ZT Server. It contains an ID, Message Element Code 3000, that is returned in the Response in Message Element Code 4000 and a Default Effective Timestamp, Code 3003. The ZTIC and Code of the Update Mode in Message Elements 30080 and 30081.

The ObjectSet, Message Element Code 3030, and its children contain the object data to be created or updated on the server. If the newCode Message Element 3036 is set to + (true), this indicates that the Server should generate a new object code on the server. In this case the code is a temporary code only used in the context of the message.

Request QuerySet

The QuerySet, Message Element 31, section of the request contains information about data that the client wishes to retrieve from the server. The first node under the QuerySet element is Query, Message Element 310. There can be 1 to many Query nodes in the QuerySet. Each Query contains an ID in Message Element 3100. The value in the ID field is assigned by the client. The same value is returned by the server under the QueryResponse in Message Element 4100. Message Elements 3101 and 3102 hold the Template ZTIC and Template Code which determines the object values returned in the response. Message Elements 31030 and 31031 hold the ZTIC and Code of the selection mode. The mode indicates, for example, if only current values are selected (mode 1) or a complete history of values is selected (mode 2). Message Element 31032 has status values that should be excluded from a response, for example objects with an inactive status could be excluded. Message Element 3104, described in the next paragraph, relates to timestamps. Message Element 3105 is the parent query response control parameters. Message Element 3107 has selection parameters that can be used, for example, when an object element has a calculated value. Message Element 3110 is the parent for selection sets.

The TimestampSet, Message Element 3104, is an optional element used to specify the timestamp or the range of timestamps being selected. If no TimeStamp is included the sever uses current time for the selection. Message Element 31040 is a child of 3104 and has two child elements, operator and value, Message Elements 31041 and 31042. The allowed operators are ==, <=, >=, <, or > which correspond to equal, less than or equal, greater than or equal, less than, or greater than. To specify a range of timestamps, two TimeStampEffective elements should be used, one to contain the lower boundary with a > or >= operator and one for the upper boundary with a < or <= operator. The value element can contain a timestamp or -999999999 ... to represent the beginning of time or 9999999999 ... to represent the end of time. The value of ! is used to represent current time. the value of @ is used to represent a day. The plus, minus, * (for multiplication) or / (for division) symbols can also be used in conjunction with the current time. Parenthesis can be used to indicate precedence for calculation. For example, a value of !+3600 represents one hour after the current time. A value of !-3600 represents one hour before the current time. For example !-(@*3)would mean 3 days before the current time. When calculating the seconds for a timestamp using days it would be adjusted for leap seconds if they exist in the time period being calculated.

The Selection section of the Query is used to specify which data should be selected. The first node under Selection node is called groupNumber. This is a 4 digit numeric value. Selections with the same group number are treated as AND relationships. Selections when the group number differs are treated as OR. This means that is if there are two groups that select object that have the same group number, the server will only return objects that the two selections have in common. If the group numbers differ, the server will return the objects if the result from the selection of either group. Under the Object element is the Level element which indicates where is the hierarchy the Object resides. For an object at the top of the hierarchy, this would be given a value of 0. The next level down would have a value of 1. Typically the level in the selection would 0. There may be cases when it may be somewhere else. For example, suppose a group of financial documents is being selected based on line item amount. Object Elements are selected using a selection operator. The allowed operators are ==, <=, >=, <, or >, =*, *= and *=* which correspond to equal, less than or equal, greater than or equal, less than, greater than, begins with, ends with and contains. To select all documents with a value greater than $100.00. For this example, assume that the financial document uses an object kind for header and another object kind for line items which contains the amounts. In this case, the selection that contains the amount being compared is stored one level down from top level (level 1). The server could do a selection on the line items table and find all postings that are greater than 100.00. The selection would specify that selection is based on the link type of financial document header to financial document line item. In this case the links need to be queried in reverse to find all of the header objects that are linked to line item objects that or 100.00 or more

The selections determine what objects are returned by the query but the the data content of the objects. The data content is determined by the object templates used for selection.

Response MaintainResponseSet

The Message Response MaintainResponseSet contains data populated by the server about objects that were maintained on the server. Of particular importance, it provides a mechanism for the server to let the client know what new codes were assigned to newly created objects in the target namespace and the related temporary codes that were assigned by the client until object creation was completed by the server.

Response QueryResponseSet

The Response QueryResponeSet is a section of the message where all the objects that were queried by the client are listed by the server based on the template used in the query. The ID and ParentID Message Elements are used to convey the hierarchy of objects that resulted from links from parent objects to child objects

Message Syntax

The Message Definition described thus far relates to the hierarchy of message data for the request that would be populated by the client and processed by the server and the response to be populated by the server and interpreted by the client. The structure of the message as it is communicated i.e. from client to server differs from this. The message used for communication can be viewed as a series of segments. Each segment is assigned a sequential integer beginning with serial number 1. Each segment contains the serial number of its parent. It also contains a reference to a Message Element ZTIC and Code which indicates the placement of the message data within the hierarchical structure of the Message Definition. The parent of the segment with serial number 1 is 0. Segments are always added (appended) to the end of the message and always a child of a previously added segment.

Messages could be communicated to a server using any protocols the server supports. For example, the Message segments could be put into a JSON document and transferred via a Web Service call. Message segments could be sent as a set of table rows containing message segments, assuming the server has access to that table. They could also be contained in a xml document with a simple structure to contain the segment data. Alternatively segments could be sent in a file with the segments separated by line feeds.

The structure of the communication form of the message is listed below.

Communication Message Structure

Description	Table Element Code	Data Type
Segment number	1	Integer
Segment number of Parent	2	Integer
Priority for child segments	3	Floating decimal numeric
Message Element ZTI Code	4	Character
Message Element Code	5	Character
Message Element data	6	Variable length string of Characters

Note: Segment with serial number 0 is a segment that can be added to pass instructions to the server related to, for example, the structure of the message. The parameter segment takes the form of a string with segment serial number 0 followed by the parameters. Each parameter begins with a [character followed immediately by a character that will be used as a delimiter for the parameter values that follow. For example for comma separated values it the parameter would begin with [, values. These are followed by the parameter ZTIC, the parameter Code and the parameter value. These are separated by the delimiter which is this case is the comma. The parameters could include the namespace and code of the table to indicate the structure of the incoming message. This can be used if there is a desire for a more compact message by using something other than the default structure for the communication message. For example, Table Element 4 in the default structure is the ZTIC of the Message Element. If only standard message elements are used, then this will always have the same value and could be omitted and the value for the ZTIC could be assumed. Table Element 3 in the standard structure is to indicate the priority of child message segments. If this is not important then this could also be omitted in the message. This functionality applies only if it is supported by the server receiving the message.

Message Processing Steps

The following is a list of steps for Message Processing:

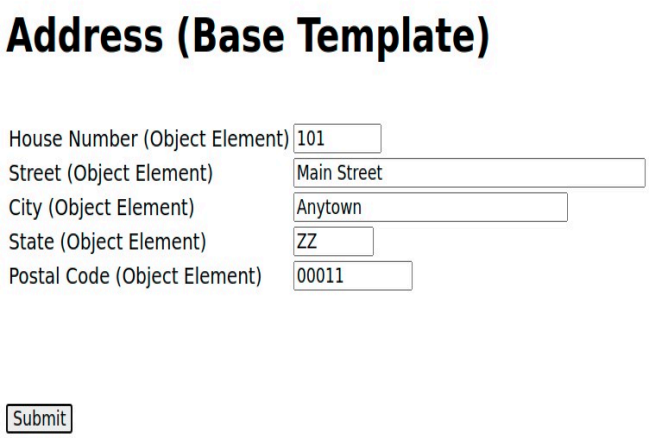
1. A Message is delivered by a client to the Server as payload by some protocol supported by the Server.
2. The Server performs a technical syntax validation to ensure that the message refers to a Message Definition that is recognized by the server, all segments are sequentially numbered and each segment refers to a valid Message Element and the Message Element of the parent segment is the correct parent per the Message Definition.
3. The Server parses the message and transforms it into an internal representation determined by the server developer.

4. The ZT Instance Codes are converted to the ZT Instance Codes used by the Server
5. The Server validates objects data sent in the MaintainSet section of the Request Message. For example it checks that Object Kinds referred to in the Message exist in the Server. It also checks that all Object Element Values, Type values and Links conform to Object Elements, Type Definitions and Link Types included in the Object Template specified in the message. Any validations from Data Elements that are referred to by Object Elements being used are invoked. If any validations fail, update of the data is stopped and validation error messages are returned in the Message Response under ServerLogSystemMessageSet, Message Element 43.
6. The Server applies Application validations to object in the Message MaintainSet. If any validations fail, update of the data is stopped and validation error messages are returned in the Message Response.
7. The Server updates validated data from the MaintainSet section of the Request to the database.
8. The Server evaluates entries in the QuerySet section of the Request Message and selects the appropriate object data from the database and places is in the internal representation of the Message.
9. The Server creates the Message segments of the Response, converting the ZT Instance Codes used in the Server to the ZT Instance Codes used in the message.
10. The Response Message is returned to the calling client

Message Processing Example

Introduction

To clarify the concepts presented thus far, this section provides examples of message processing, starting with the simplest cases. The first example involves sending a message to the server to create an address and then a subsequent message to query the address. As the first step, suppose a ZT Client application uses the screen below to capture the address data.



Create Address Message Example

To create an address on a server, it is necessary for the client to send a message with address data to the server. For the message to successfully process, there needs to already exist on the server all of the objects needed for address definition. This example assumes that these objects already exist on the server. The table below show what

the initial message with address data would look like.

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
1	0	0001	4	1	
2	1	0001	4	2	
3	2	0001	4	20	
4	3	0001	4	200	
5	3	0001	4	201	
6	3	0001	4	202	
7	3	0001	4	203	
8	2	0001	4	21	
9	2	0001	4	23	
10	9	0001	4	230	
11	9	0001	4	231	
12	1	0001	4	3	
13	12	0001	4	30	
14	12	0001	4	31	
15	12	0001	4	32	
16	12	0001	4	33	
17	8	0001	4	210	
18	17	0001	4	2100	1
19	17	0001	4	2101	zt_abc.com/test1
20	8	0001	4	210	
21	20	0001	4	2100	2
22	20	0001	4	2101	131131/21
23	8	0001	4	210	
24	23	0001	4	2100	3
25	23	0001	4	2101	zinfinitree.com/address
26	8	0001	4	210	
27	26	0001	4	2100	4
28	26	0001	4	2101	131131/22

The graphic shows a request message to create an address. This shows the structure of the message as it is communicated. The first column, SegId is a sequentially assigned identifier for the segment. The Parent column has the Id number of the parent segment. Prio (Priority) is optional and can be used to assign priority when two child segments share the same parent. The ElemZTIC has the ZTIC of the Message Element that this segment refers to. The ElemCode has the Message Element Code that the segment refers to. Value indicates the data value used by the segment for segments that can hold a value. The Message Element ZTIC and Message Element Code shown here refers to a Message Definition. The Message Definition for the base standard is shown in Appendix 6. All of the segments in this example refer to standard Message Elements so they all use the same ElemZTIC, 4, which represents namespace 131131/22, the namespace used for standard Message Elements.

If we look at the first segment of the message, with segment ID 1, we notice that the parent segment is 0 which means that this segment has no parent and is the root segment. The Prio (Priority) column is not relevant to this example. The ElemZTIC is the same for all segments because all come from the base specification. The ElemCode (Message Element Code) is a reference to the Message Definition found in Appendix 6. In this case ElemCode 1 refers to the root element of the Message Definition. The Value column for segment Id 1 is blank because the root Message Element is a node which serves as the parent for other Message Elements and cannot hold data itself.

Continuing to Segment Id 2, we see that the Id of the Parent is 1, which means it is a child of the root Message Element. If we look at the Message Definition in Appendix 6, we see that Message Element 2 is for the message Header. Header is also a node that has child Message Elements and therefore has no value itself. Segment Id 3 is a child of 2 (Header) and has Message Element Code 20 which the Message Definition indicates is the node for declaring the template and code for the Message Definition. Again, this is a node and has no value itself. Segment Ids 4 through 7 refer to Message Elements 200 through 203. These are not nodes but rather are Message Elements

that could contain values for the Message Definition Template ZTIC/Code the Message Definition ZTIC/Code. These are children of Segment Id 3 (Message Element 20). For this example, Message Elements 200 through 203 are blank and the server then assumes that the message definition from the base specification should be used.

Segment Id 8 refers to Message Element 21, which is a child node of Header, because segment Id of the parent is 2. Message Element 21 is a node that is a parent for declarations of ZTI Codes and namespaces used in the context of the message. Segment Ids 9 through 16 are additional nodes that are parents of other Message Elements used further down the message. These include 23 (Segment Id 9) for Message Processing Parameters, 230 (Segment Id 10) for General Processing Parameters, 231 (Segent Id 11) for Extended Key Sets, 3 (Segment Id 12) for the top level of the Request Section, 30 (Segment Id 13) for top of the MaintainSet Section, 31 (Segment Id 14) for top of the QuerySet, 32 (Segment Id 15) for RequestMessageStatusSet and 33 (Segment Id 16) for ClientLogMessageSet.

Message Segment ID 17 begins the declaration of the first ZTIC namespace pair. The ZTIC is in segment Id 18 with Message Element Code 2100 and a value of 1. The namespace is in Segment Id 19 with Message Element 2101 and a value of zt_abc.com/test1. These ZTICs and namespace relationships are only valid within the context of this message. When the message is processed, for example by a server, the ZTICs need to be converted to the values used on the receiving server. Segment Ids 20 through 28 are used to declare other ZTICs and namespaces following that same pattern.

The next graphic is a continuation of the previous which shows an example message to create an address. Segment Ids 29 through 40 were omitted because they are additional ZTIC-namespace pairs that are not needed for this example.

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
41	2	0001	4	22	
42	41	0001	4	223	1
43	10	0001	4	2301	5136301043.6544
44	10	0001	4	2302	id123
45	10	0001	4	2303	1234
46	10	0001	4	2304	1
47	10	0001	4	2305	usr1
48	10	0001	4	2306	7
49	10	0001	4	2307	1
50	11	0001	4	2310	
51	12	0001	4	2311	2
52	12	0001	4	2312	2
53	12	0001	4	2313	2
54	12	0001	4	2314	1
55	13	0001	4	300	
56	55	0001	4	3000	MaintainSetId#1
57	55	0001	4	3001	
58	55	0001	4	3002	
59	55	0001	4	3003	5136301043.6544
60	55	0001	4	3008	
61	55	0001	4	3030	

Segment Id 41 is a child of Segment Id 2, which contains the Header. It has message element 22 which is used for message partners. It is a node with child elements and holds no value itself. Segment Id 42 is a child of Segment Id 41 and refers to Message Element 223 which is for the Receiver ZTIC. The server uses the Receiver ZTIC to determine which Server DB--ZT Instance ID to use when storing message data. The Receiver ZTIC in this case is 1. In the ZTIC-Namespace declarations, ZTIC is associated with namespace zt_abc.com/test1 in Segment Id 17. The server keeps track of the namespace for each Server DB--ZT Instance ID that it handles. If the zt_abc.com/test1 namespace is associated with a Server DB--ZT Instance ID that the server handles, it will use that Server DB--ZT Instance ID to store the data. The Server DB--ZT Instance ID is stored in Table Element 32.

Segment Id 43 through 49 are children of Segment Id 10 for Message Element 230, General Message Processing Parameters. Segment Id 43 is for Message Element 2301 which holds the default timestamp. Segment Id 44 has Message Element 2302 which hold the client message Id. Message Id 45 has Message Element 2303 which is the last segment Id of the request message. This is used to indicate which was the last segment Id populated by the client. subsequent segments are populated by the server. Last segment Id in the request is an unimplemented feature in this example and 1234 is used as a placeholder. The value should be 92. Segment Id 46 which has Message Element 2304 has the default update mode which is applied if not specified in a MaintainSet. Segment Id 47 which has Message Element 2305 has the user name a value. The user must be authenticated by the client before passing the message to the server. Segment Ids 48 and 49 have the ZTIC and code of the ServerSoftwareRuntimeProfile which is used to determine which server software patch level to use when processing the message.

Segment Id 50 for Message Element 2310 is a child of Segment Id 11 and is the parent segment for an Extended Key and key value. Segment Id 51 for Message Element 2311 contains the Extended Key Definition Code ZTIC. This contains a value of 2 which is the code of the base namespace. Segment Id 52 for Message Element 2312, Extended Key Definition Code has a value of 2 indicating that it is the extended key for language. Segment Id 53 for Message Element 2313, Extended Key Value ZTIC has a value of 2, the code for the base namespace. Segment Id 54 for Element 2314, Extended Key Value Code, has a value of 1, indicating language English. This means that any Object Elements that use the Extended Key for language will be recorded with the key for the English language.

Segment Id 55 for Message Element 300 is a child of Segment Id 13, Message Element 30 for Maintain Set which contains Maintain nodes. Segment Id 56 for Message Element 3000 is a child of Segment Id 55 and contains the identifier of the Maintain node. In this case the Identifier is MaintainSetId#1. Message Id 59 for Message Element 3003, contains the Default Effective TimeStamp for the message update. Message Id 61 for Message Element 3030, named Object Set is the parent Message Element for object data that will be created or updated by the message.

The next graphic shows the remaining segments from the Request portion of the message.

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
62	61	0001	4	3031	
63	62	0001	4	3032	3
64	62	0001	4	3033	1
65	62	0001	4	3034	1
66	62	0001	4	3035	1
67	62	0001	4	3036	+
68	62	0001	4	3037	
69	62	0001	4	3038	
70	62	0001	4	30390	3
71	62	0001	4	30391	1
72	62	0001	4	3040	
73	72	0001	4	3041	
74	73	0001	4	3042	3
75	73	0001	4	3043	1
76	73	0001	4	3044	101
77	72	0001	4	3041	
78	77	0001	4	3042	3
79	77	0001	4	3043	2
80	77	0001	4	3044	Main Street
81	72	0001	4	3041	
82	81	0001	4	3042	3
83	81	0001	4	3043	3
84	81	0001	4	3044	Anytown
85	72	0001	4	3041	
86	85	0001	4	3042	3
87	85	0001	4	3043	4
88	85	0001	4	3044	ZZ
89	72	0001	4	3041	
90	89	0001	4	3042	3
91	89	0001	4	3043	5
92	89	0001	4	3044	00011

Segment Id 62 for Message Element 3031 is a child of Segment Id 61 and is the parent for the object identifiers of an object being maintained. Segment Id 63 for Message Element 3032 for Object Kind ZTIC contains a value of 3 which corresponds to namespace ds.com/address. The server converts the namespace ds.com/address to the ZT Instance Code that corresponds to the ds.com/address on the server and then updates the converted code into table element 1. Segment Id 64 for Message Element 3033 for Object Kind Code contains a value of 1. This the code of the Object Kind for Address within the Address application which was defined within namespace ds.com/address. Segment Id 65 Message Element 3034 for Object ZTIC has a value of 1 which corresponds to namespace abc.com/test1, the target namespace of the message where the new address is to be created. Segment Id 66 for Message Element 3035 for Object Code Temp. This has a value of 1 which a temporary code used only within the context of the message. The actual code is assigned by the server as the message is processed. Segment 67 for Message Element 3036 has a value of + which is

a boolean that indicates that this is a newly created object that will be assigned a code by the server. Segments 68 and 69 for Message Elements 3037 and 3038 can be used to indicate hierarchy for multi-level objects. Segments Id 70 for Message Element 30390 for Template ZTIC. This has a value of 3 which corresponds to namespace ds.com/address. Segment Id 71 for Message Element 30391 for Template Code with a value of 1, a Template defined within the Address application. Segment Id 72 for Message Element 3040 for Element Set which is the parent for Object Element values being maintained.

Segment Id 73 for Message Element 3041 for Element is the parent for a single Object Element value. Segment Id 74 for Message Element 3042 for Object Element ZT Instance Code has a value 3. This corresponds to namespace ds.com/address, the namespace for the address application. Segment Id 75 for Message Element 3043 for Object Element Code has a value of 1. This is code assigned to the Object Element, House Number within the address application. One of the validations the server must perform is to ensure that if data is being maintained for an Object Element, the Object Element must already be exist in the server and be a valid Object Element for the Object Template being used to maintain the object. Segment Id 76 for Message Element 3044 for Object Element value. This has a value of 101, the house number for the address being created. Segment Ids 77 to 80 follow the same pattern as Segment Ids 73 to 76. In this case it is for Object Element Code 2 which is used for Street which has a value of Main Street. Segments Ids 81 to 84 continue the same pattern, in this case for Object Element Code 3 for City which contains a value of Anytown. Segment Ids 85 to 88 use the same pattern, in this case for Object Element Code 4 for State/Province with a value of ZZ. Segment Ids 89 to 92 continue the pattern, in this case for Object Element Code 5 for Postal Code with a value of 00011.

The graphic below shows the response from the server.

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
93	1	5000	4	4	
94	93	5000	4	40	
95	94	5000	4	400	
96	95	5000	4	4030	
97	96	5000	4	4031	
98	97	5000	4	4032	3
99	97	5000	4	4033	1
100	97	5000	4	4034	1
101	97	5000	4	4035	1
102	97	5000	4	4036	57
103	97	5000	4	4037	+

All of the previous message segments were populated by the client to form the Request portion of the message. All of the remaining segments shown above were populatd by the Server to form the message Response. Segment Id 93 for Message Element 4 is a child of Segment Id 1 for Message Element 1, the root of the entire message. Message Element 4 is the parent for the message Response. Segment Id 94 for Message Element 40 for Maintain Response Set is a child of Segment Id 93. Segment Id 95 for Message Element 400 for Maintain Response is a child of Segment Id 94. Segment Id 96 for Message Element 4030 for Object Set is the parent for objects in the Maintain Response. Segment Id 97 for Message Element 4031 is the parent for an individual object of the Object Set. Segment Id 98 for Message Element 4032 for Object Kind ZT Instance Code has a value of 3 for which corresponds to ds.com/address. This ZT Instance Code is based on the ZT Instance Codes defined within the message. The actual ZT Instance code used to save the address on the server will be based on the ZT Instance Codes used on the server. Segment Id 99 for Message Element 4033 for Object Kind Code. This is the Kind Code set up for addresses in the address application defined in the ds.com/address namespace. Segment Id 100 for Message Element 4034 for Object ZT Instance Code which has a value of 1 which corresponds to ds.com/test1 the target of the message which is the ZT Instance where the new address is created. Segment Id 101 for Message Element 4035 for Object Code Temporary. This is the temporary code that was sent by the client that is used to identify the object until an object code is assigned by the server. Segment Id 102 for Message Element 4036 for Object Code Assigned has a value of 57, the code assigned by the server.

Updated Database

The graphic below shows the updated database table on the server

Svr DB ZTI Id TE-32	Kind ZTIC TE-1	Kind Code TE-2	Object ZTIC TE-3	Object Code TE-4	OE ZTIC TE-7	OE Code TE-8	EK Def ZTIC TE-9	EK Def Code TE-10	EK ZTIC TE-11	EK Code TE-12	Eff TS TE-26	OE Value TE-31
TST1	4	1	1	57	4	1					5148916905.1968	101
TST1	4	1	1	57	4	2					5148916905.1968	Main Street
TST1	4	1	1	57	4	3	2	2	2	1	5148916905.1968	Anytown
TST1	4	1	1	57	4	4					5148916905.1968	ZZ
TST1	4	1	1	57	4	5					5148916905.1968	00011

The column headings in this example contain TE- followed by a number. TE stands for Table Element. The number after TE- is the code of the Table Element as shown in the Key Concepts--Introduction section of this document. The value in the first column for TE-32, Server DB-ZT Instance Id, is derived by the server based on the namespace that corresponds to the code in Message Element 223, found in Segment Id 42. In this case the code is 1. The namespace that corresponds to code 1 is found in Message Element 2101, Segment Id 19 is abc.com/test1. The server finds the Server DB--ZT Instance Id based a configuration file that relates the namespaces with Server DB--ZT Instance Id. In this case the Server DB--ZT Instance Id is TST1. The server applies the TST1 value to all updates related to the message.

The second column headed TE-1 hold the ZT Instance Code for the Object Kind of the Object. The source of the data for this column is based on the value of message Segment Id 63 for Message Element 3032 which has a Object Kind ZTIC value of 3. A conversion is performed by the server based on the namespace that this code represents. In the message, Segment Ids 24 and 25 we see that code 3 represents namespace ds.com/address, the namespace used to develop the Address application. The graphic below shows the ZT Instance codes on the server for ZT Instance TST1. Here we can see that the namespace ds.com/address is associated with ZT Instance Code 4. When the server updates the table with the address data, it uses Object Kind ZTIC with the value, 4, after converting the ZTIC values from the value relevant in the message to the value relevant in the server.

TST1--ZT Instance Code	Namespace
1	zt_abc.com/test1
2	131131/21
3	zinfinitree.com/server_config
4	zinfinitree.com/address
5	131131/22

The third column headed TE-2 holds the Object Kind Code. This is the code assigned to the Address Object Kind in the ds.com/address application which has a value of 1. The value was taken from Segment Id 64 for Message Element 3033.

The fourth column headed TE-3 contains the ZT Instance Code (ZTIC) of the address being created. Because this is an object being created locally in target ZT Instance, the namespace associated with the object is the same as the namespace used to identify the target instance, abc.com/test1. In this case, the ZTIC associated with abc.com/test1 in the message, in Segment Id 18 and 19, is the same as the ZTIC associated with abc.com/test1 in the server (code 1) so there is no need for the server to convert the ZTIC in this case. As a result, the server populates this column with the value 1, the value from Segment Id 65 for Message Element 3034 for Object ZTIC.

The fifth column headed TE-4, contains the Code of the newly created address. Although a Code was provided by the client in the request message in Segment Id 66 for Message Element 3035 with a value of 1, this code was temporary. The actual Code for the address, 57, was generated by the server, placed in TE-4 column. The generated Code was provide to the client in the message response in Segment Id 102 for Message Element 4036.

The sixth column, headed TE-7, contains the Object Element ZT Instance Code. These are Object Elements that were defined as part of the Address Application under the namespace ds.com/address, the same namespace used to define the Object Kinds. The source of the ZTIC in the message is Segment Id 74, 78, 82, 86 and 90 for Message Element 3042 with a value of 3. A conversion is performed by the server based on the namespace that this code represents. In the message, Segment Ids 24 and 25 code 3 represents namespace ds.com/address, the namespace used to develop the Address application. The ZTIC associated with ds.com/address on the server is 4, so this column is populated with the 4, the same ZTIC used in the second column headed TE-1.

The seventh column, headed TE-8, contains Object Element Codes. The meaning of the code was defined as part of the address application. The source of the data in the message are Segment Ids 75, 79, 83, 87 and 91 for Message Element 3043 with values 1, 2, 3, 4 and 5. While there could be a variety of Object Elements defined for an Object Kind, the server should only accept maintenance of Object Elements that are are included for the Object Template being used by the message being processed.

The eighth column, headed TE-9 contains the Extended Key Definition ZT Instance Code. This column is only used if the Object Element uses an extended key. In this example, only the Object Element for City uses Extended Key to accommodate the names of cities in multiple human languages. The source of this ZTIC is Segment Id 51 for Message Element 2311 with a value of 2 which corresponds to the namespace 131131/21, the base namespace as shown is Segment Ids 21 and 22. The server also uses ZTIC 2 to represent the base namespace so there is no need for the server to convert the ZTIC from the value used in the message to the value used in the server. As a result, the value 2 is used in the table when storing the city for the address being maintained.

The ninth column headed TE-10 contains the Extended Key Definition Code. The source of this data is Segment Id 52 for Message Element 2312. It is used to identify which Extended Key Definition the Extended Key relates to. In this case the Extended Key Definition is for language. As with the eighth column, it is only populated for Object Elements that use an extended key and for this example only the Object Element for Address City can be maintained in multiple languages.

the tenth column headed TE-11 contains the Extended Key ZT Instance Code. As with the column headed TE-9, this column is only used if the Object Element uses an extended key. In this example, only the Object Element for City uses Extended Key. The source of this ZTIC is Segment Id 53 for Message Element 2313 with a value of 2 which corresponds to the namespace 131131/21, the base namespace, as shown is Segment Ids 21 and 22. And as with TE-9, the server also uses ZTIC 2 to represent the base namespace so there is no need for the server to convert the ZTIC from the value used in the message to the value used in the server. As a result, the value 2 is used in the table when storing the city for the address being maintained.

The eleventh column headed TE-12 contains the Extended Key Code. The source of this data is Segment Id 54 for Message Element 2314. The Code 1 for the purpose of this example represents the English language, the language used for maintaining the City for this address. This is because the Object Element for City was configured to use an Extended Key to allow storing the city name in multiple languages.

The twelfth column headed TE-26 contains the effective timestamp of the record which is the number of seconds or decimal fractions of a second elapsed since November 17, 1858, Midnight UTC, on the Gregorian calendar. For data that was created in another namespace the timestamp would be assigned on that server managing that namespace. Because the address is being created locally the timestamp is assigned by the local server processing the message.

The thirteenth and final column headed TE-31 contains the Object Element value that is being updated by the message. The source of the data is Segment Id 76, 80, 84, 88 and 92 for Message Element 3044 for Object Element Value.

Address Query

This section shows in detail an example message used by a client to query the address created in the previous example. When an object is queried, it is frequently useful to also query the template of the object being queried. For this example, both the Address Template and the Address Object are queried. The graphic below shows the first part of the message. The explanation for this section of the message can be taken from the Address Create example because the beginning part of the two messages are

very much the same.

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
1	0	0001	4	1	
2	1	0001	4	2	
3	2	0001	4	20	
4	3	0001	4	200	
5	3	0001	4	201	
6	3	0001	4	202	
7	3	0001	4	203	
8	2	0001	4	21	
9	2	0001	4	23	
10	9	0001	4	230	
11	9	0001	4	231	
12	1	0001	4	3	
13	12	0001	4	30	
14	12	0001	4	31	
15	12	0001	4	32	
16	12	0001	4	33	
17	8	0001	4	210	
18	17	0001	4	2100	1
19	17	0001	4	2101	zt_abc.com/doc1
20	8	0001	4	210	
21	20	0001	4	2100	2
22	20	0001	4	2101	131131/21
23	8	0001	4	210	
24	23	0001	4	2100	3
25	23	0001	4	2101	zinfinitree.com/address
26	8	0001	4	210	
27	26	0001	4	2100	4
28	26	0001	4	2101	131131/22

The next graphic completes the header of the message and begins filling in segments for the Query and Selection of the Address template.

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
50	8	0001	4	210	
51	50	0001	4	2100	12
52	50	0001	4	2101	zt_abc.com/test1
53	2	0001	4	22	
54	53	0001	4	223	12
55	10	0001	4	2301	5136387443.6544
56	10	0001	4	2302	id123
57	10	0001	4	2303	1234
58	10	0001	4	2304	1
59	10	0001	4	2305	usr1
60	10	0001	4	2306	9
61	10	0001	4	2307	2
62	11	0001	4	2310	
63	62	0001	4	2311	2
64	62	0001	4	2312	2
65	62	0001	4	2313	2
66	62	0001	4	2314	1
67	14	0001	4	310	
68	67	0001	4	3100	ObjTmplQset
69	67	0001	4	3101	2
70	67	0001	4	3102	2
71	67	0001	4	3103	1
72	67	0001	4	3104	+
73	67	0001	4	3110	
74	73	0001	4	3111	
75	74	0001	4	3112	
76	74	0001	4	3113	
77	74	0001	4	3114	

Segment Id 50 through 52 declares the namespace zt_abc.com/test1 associates it with code 12. Segment Id 54 for Message Element 223 hold the Receiver ZT Instance Code. It holds a value of 12 which means the the namespace of the target ZT Instance is zt_abc.com/test1. Segment Id 55 is a child of Segment Id 10 for Message Element 230 which is the parent for General Message Processing Parameters. Segment Id 55 uses Message Element 2301 which contains a timestamp, the number of seconds or decimal fractions of a second elapsed since November 17, 1858, Midnight UTC. In this case, the timestamp is used to determine which data will be selected for the query. Records are selected that have the most recent timestamp that is less than or equal to the selection timestamp in Message Element 2301. Segment Id 56 for Message Element 2302 has a Message Id that is assigned by the client. Segment Id 57 for Message Element 2303 has the Segment Id of the last segment sent by the client. Last segment Id in the request is an unimplemented feature in this example and 1234 is used as a placeholder. The value should be 112. Segment Id 58 for Message Element 2304 has the Default Update Mode which is not used in this case because the message is only for query. Segment 59 for Message Element 2305 contain the user id. Segment Ids 60 and 61 contain the ZTIC and Code of the Server Software Runtime Profile used to process the message.

Segment Id 62 for Message Element 2310 is a child of Segment Id 11 and it is the parent for an Extended Key. Segment Ids 63 to 66 for Message Elements 2311, 2312, 2313 and 2314 are children of Segment Id 62 and contain the values for an Extended Key. Message Element 2311 for Extended Key Definition ZTIC has a value of 2. The matching namespace declared in Segment Ids 21 and 22 is for namespace 131131/21, the base namespace. Message Element 2312 for Extended Key Definition Code has a value of 2 which is used to represent the Extended Key Definition for language. Message Element 2313 for Extended Key ZTIC has a value of 2 which again refers to namespace 131131/21, the base namespace. Message Element 2314 for Extended Key Code has a value of 1 which in the context of this example refers to the English language.

Segment Id 67 for Message Element 310, for Query, is the parent for the values of a Query. This message has two Query nodes. The first one is used to query the values of the Address Template. The second one is used to retrieve the values of the Address. Including the Query of the Address Template is useful because it retrieves the Meta Data of the address which provides meaningful descriptions for the Address data. Segment Id 68 for Message Element 3100 contains a Query Id that is assigned by the client. When the server returns a response, this same Id will be placed in a corresponding Message Element in the Query Response in Message Element 4100. Segment Id 69 for Message Element 3101 is for the Object Template ZTIC for object being queried. Because the object being queried is an Object Template this is the ZTIC for the Object Template of an Object Template. This has a value of 2 representing namespace 131131/21, the base namespace. Segment Id 70 for Message Element 3102 for Object Template Code has a value of 2. This is the Code for the Object Template of an Object Template. Segment Id 71 for Message Element 3103 for the Selection Mode has a value of 1 to indicate standard selection. Segment Id 72 for Message Element 3104 has a value of + which can be ignored for this example.

Segment Id 73 for Message Element 3110 for Selection Set and is the parent for Selection. Segment Id 74 for Message Element 3111 is the parent for the first Selection. Segments Ids 75, 76 and 77 for Message Elements 3112, 3113 and 3114 contain the Selection Group Number, Selection Group Number of parent and set operator. This would be used for a more complex selection involving a hierarchy of selection groups and are not used for this example.

The next graphic completes the segments of the Request side of the message

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
78	74	0001	4	3120	
79	78	0001	4	3121	
80	79	0001	4	3122	1lev
81	79	0001	4	3123	1usg
82	79	0001	4	3130	
83	79	0001	4	3140	
84	79	0001	4	3150	
85	79	0001	4	3160	
86	79	0001	4	3170	
87	82	0001	4	3131	
88	87	0001	4	3134	3
89	87	0001	4	3135	1
90	14	0001	4	310	
91	90	0001	4	3100	ObjectQset
92	90	0001	4	3101	3
93	90	0001	4	3102	1
94	90	0001	4	3103	1
95	90	0001	4	3104	+
96	90	0001	4	3110	
97	96	0001	4	3111	
98	97	0001	4	3112	
99	97	0001	4	3113	
100	97	0001	4	3114	
101	97	0001	4	3120-	
102	101	0001	4	3121	
103	102	0001	4	3122	1lev
104	102	0001	4	3123	1usg
105	102	0001	4	3130	
106	102	0001	4	3140	
107	102	0001	4	3150	
108	102	0001	4	3160	
109	102	0001	4	3170	
110	105	0001	4	3131	
111	110	0001	4	3134	12
112	110	0001	4	3135	57

Segment Id 78 for Message Element 3120 contains Object Selection Sets. Segment Id 79 for Message Element 3121 is a child of Segment Id 78 and contains an Object Selection. Segment Ids 80 and 81 for Message Elements 3122 and 3123 are for hierarchy level and selection mode. This selection does not have a multi-level hierarchy and the selection mode is 1, for standard. Segment Id 82 for Message Element 3130 is for an object set used for selection. This set will have only one member, the address template. Segment Ids 83 through 86 for Message Elements 3140, 3150, 3160 and 3170 are parents elements for selection attributes that are not used in this message. Segment Id 87 for Message Element 3131 is the parent for one object used in the selection. Segment Ids 88 and 89 for Message Elements 3134 and 3135 contain the

Object ZTIC and Object Code of the object being selected. In this case the Object ZTIC has a value of 3 which corresponds to the namespace zinfinitree.com/address the namespace where the address application was developed. The Object Code 1 is the code for the Address Template Object in the Address Application.

Segment Id 90 for Message Element 310 is the parent for the second Selection. This Selection will be used to select the Address object. It follows the same pattern that was used to select the Address Object Template. Segment Id 91 for Message Element 3100 contains the Id of the Selection. This Id will be included by the server it the response in Message Element 4100. Segment Id 92 for Message Element 3101 contains the ZT Instance Code for the template for the selection of the object. This has a value of 3 which corresponds to namespace zinfinitree.com/address, the namespace used for the address application. Segment Id 93 for Message Element 3102 contains the code of the template used for the selection of the object. This has a value of 1 for the Address Template defined in the Address Application. Segment Id 94 for Message Element 3103 has the selection mode. This has a value of 1 for standard selection. Segment Id 95 for Message Element 3104 has a boolean value + for respond in summary only. This attribute is not used in this message. Segment Ids 96 through 112 for selection the Address object use the same pattern as Segment Ids 73 through 89 used to select the Address template object. The only difference is in Segment Ids 111 and 112 for Message Elements 3134 and 3135 for the Object ZTIC and Code. In this case the ZTIC is 12 which corresponds to namespace zt_abc.com/test1, the namespace where the address was created and the code 57, the code of the newly created address.

Query Address Response

The table below shows the beginning of the Address query response. These segments are added by the server.

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
113	1	5001	4	4	
114	113	5001	4	43	
115	113	5001	4	41	
116	115	5001	4	410	
117	116	5001	4	4100	ObjTmplQset
118	116	5001	4	4104	1
119	116	5001	4	4105	
120	116	5001	4	4106	
121	116	5001	4	4130	
122	121	5001	4	4131	
123	122	5001	4	4132	2
124	122	5001	4	4133	2
125	122	5001	4	4134	3
126	122	5001	4	4135	1
127	122	5001	4	4137	1
128	122	5001	4	4138	0
129	122	5001	4	4139	0
130	122	5001	4	4140	
131	130	5001	4	4141	
132	131	5001	4	4142	2
133	131	5001	4	4143	2
134	131	5001	4	4144	Address (Base Template)
135	122	5001	4	4150	
136	135	5001	4	4151	
137	136	5001	4	4152	2
138	136	5001	4	4153	3
139	136	5001	4	4154	3
140	136	5001	4	4155	1
141	122	5001	4	4160	
142	141	5001	4	4161	
143	142	5001	4	4162	2
144	142	5001	4	4163	1
145	142	5001	4	4164	2
146	142	5001	4	4165	12
147	142	5001	4	4166	3
148	142	5001	4	4167	1
149	142	5001	4	41691	1
150	142	5001	4	41692	4919147515.000
151	142	5001	4	41693	1000

Segment Id 113 for Message Element 4 is a child of the Root Message Element and is the parent for the Message Response. Segment Id 114 for Message Element 43 is a child of Segment Id 113 and is a parent for Server Log System Messages. Segment Id 115 for Message Element 41 is also a child of Segment Id 113 and is a parent for the

Query Response. Segment Id 116 for Message Element 410 is a child of Message Segment 115 and is the parent for Message Elements for the first Response node used to query the Address Template. Segment Id 117 for Message Element 4100 is a child of Segment Id 116. This segment holds the Id of the of Response node. The value, ObjTmplQSet was assigned by the client and sent in Message Element 3100 in Segment Id 68. The same value is returned by the server so that the client knows which Query node the Query Response Set is responding to. Segment Id 118 for Message Element 4104 is a child of Segment Id 116 and hold the Response Type with a value of 1 for standard. Segment Id 119 and 120 are not used for this message.

Segment Id 121 for Message Element 4130 is the parent for the objects that will be included in the Response node for the Address Template. Segment Id 122 for Message Element 4131 is a child of Segment Id 121. This is the parent for the Message Elements of the an individual object, in this case the object for The Address Template. Segment Id 123 for Message Element 4132 is a child of Segment Id 122. It holds the Object Kind ZT Instance Code with a value of 2. This refers to namespace 131131/21 the base namespace in Segment Ids 21 and 22. Segment Id 124 for Message Element 4133 for the Object Kind Code has a value of 2, the Kind Code for Object Templates. Segment id 125 for Message Element 4134 has the ZT Instance Code of the Object with a value of 3. This corresponds to namespace zinfinitree.com/address in Segment Ids 24 and 25, the namespace used by the Address application. Segment Id 126 for Message Element 4135 hold the Object Code with a value of 1, the Code of the Address Template within the Address application. Segment Id 127 for Message Element 4137 with a value of 1 holds a numeric Id, assigned by the server to identify the object in a hierarchy of objects returned by the server. Segment Id 128 for Message Element 4138 holds the Id of the parent object. Because the Address Template is at the top of the hierarchy, the parent has a value of 0. Segment Id 129 for Message Element 4139 has the number of levels down in the hierarchy. Because the Address Template object is at the top of the hierarchy the levels down is 0.

Segment Id 130 for Message Element 4140 is the parent for Message Elements containing the Object Elements of an object. Segment Id 131 for Message Element 4141 is a child of Segment Id 130 and is the parent for the Message Elements holding an individual Object Element. Segment Id 132 for Message Element 4142 holds the ZT Instance Code of the Object Element with a value of 2. This corresponds to namespace 131131/21 for the base namespace. Segment Id 133 for Message Element 4143 hold the Code of the Object Element with a value of 2. This the is the code for Object Template Description. Segment Id 134 for Message Element 4144 hold the value of the Object Element, in the case the value is: Address (Base Template)

Segment Id 135 for Message Element 4150 is the parent for Message Elements for Type Values of the Address Template. Segment Id 136 for Message Element 4151 is the parent for an individual Type Value. Segment Id 137 for Message Element 4152 is a child of Segment Id 136. It contains the ZT Instance Code for the Type Definition of the Type Value and has a value of 2. This corresponds to namespace 131131/21, the base namespace. Segment 138 for Message Element 4153 is also a child of Segment Id 136. It contains the Code for the Type Definition of the Type Value and has a value of 3. Type Definition Code 3 is used to specify the Object Kind for a Object Template. Segment Id 139 for Message Element 4154 is a child of Segment Id 136. It contains the ZT Instance Code for the Type Value with a value of 3 which corresponds to namespace zinfinitree.com/address. Segment Id 140 for Message Element 4155 contains the Type Value Code and has a value of 1. Object Kind 1 within the zinfinitree.com/address namespace represents the Address Object Kind.

Segment Id 141 for Message Element 4160 is the parent for links from the Address Template. Segment Id 142 for Message Element 4161 is a child of Segment Id 141 and is the parent for an individual link. Segment Id 143 for Message Element 4162 is a child of Segment Id 142. This contains the ZT Instance Code of the Link Type and has a value of 2, the base namespace, 131131/21. Segment Id 144 for Message Element 4163 for the Link Type Code and has a value of 1. Link Type Code 1 represents to Link Type to link an Object Template to a Technical Profile. Segment Id 145 for Message Element 4164 is for the ZT Instance Code of the Object Kind of the object being linked to. This has a value of 2 which corresponds to the base namespace. Segment Id 146 for Message Element 4165 contains the Kind Code for the object being linked to. This has a value of 12 which is the Kind Code for Technical Profiles. Segment Id 147 for Message Element 4166 for the ZT Instance Code of the object being linked to. This has a value of 3 which corresponds to namespace zinfinitree.com/address. Segment Id 148 for Message Element 4167 contains the code of the object being linked to. This has a value of 1 which is the code for the Technical Profile used by the Address Template in the Address application. Segment Id 149 for Message Element 41691 contains the status of the link. This has a value of 1 for standard/active. Segment Id 150 for Message Element 41692 contains the effective timestamp for the link. Segment Id 151 for Message Element 41693 contains the value or strength of the link. This has a value of 1000.

The next table shows values returned by the server for the Address Technical Profile. When an object is queried, in this case the Address Template, the server returns not only the object queried but also any objects that the queried object is linked to. In this case the Address Template is linked to a Technical Profile. In turn, any object that the child object is linked to is also returned. This continues down the hierarchy until any object that was linked to is returned. Each object that is returned follows the same pattern as the Address Template to include some combination of Object Element, Type Values and links. Message Element 4137, 4138 and 4139 indicate relationships of the objects. Message Element 4137 has a numeric id for the object and 4138 has the id of the parent. In the graphic below, Message Element 4137 has a value of 2, the Id value of the Technical Profile. Message Element 4138 has a value of 1, the Id of the parent of the Technical Profile which is the Address Template. Message Element 4139

for levels down has a value of 1 because the Technical Profile is 1 level down the root level.

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
152	121	5001	4	4131	
153	152	5001	4	4132	2
154	152	5001	4	4133	12
155	152	5001	4	4134	3
156	152	5001	4	4135	1
157	152	5001	4	4137	2
158	152	5001	4	4138	1
159	152	5001	4	4139	1
160	152	5001	4	4140	
161	160	5001	4	4141	
162	161	5001	4	4142	2
163	161	5001	4	4143	12
164	161	5001	4	4144	Address (Technical Profile)
165	152	5001	4	4160	
166	165	5001	4	4161	
167	166	5001	4	4162	2
168	166	5001	4	4163	3
169	166	5001	4	4164	2
170	166	5001	4	4165	3
171	166	5001	4	4166	3
172	166	5001	4	4167	1
173	166	5001	4	41691	1
174	166	5001	4	41692	4919147515.000
175	166	5001	4	41693	1000
176	165	5001	4	4161	
177	176	5001	4	4162	2
178	176	5001	4	4163	3
179	176	5001	4	4164	2
180	176	5001	4	4165	3
181	176	5001	4	4166	3
182	176	5001	4	4167	2
183	176	5001	4	41691	1
184	176	5001	4	41692	4919147515.000
185	176	5001	4	41693	1000

The next table shows links from the Technical Profile to other Objects used by the Object Template

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
186	165	5001	4	4161	
187	186	5001	4	4162	2
188	186	5001	4	4163	3
189	186	5001	4	4164	2
190	186	5001	4	4165	3
191	186	5001	4	4166	3
192	186	5001	4	4167	3
193	186	5001	4	41691	1
194	186	5001	4	41692	4919147515.000
195	186	5001	4	41693	1000
196	165	5001	4	4161	
197	196	5001	4	4162	2
198	196	5001	4	4163	3
199	196	5001	4	4164	2
200	196	5001	4	4165	3
201	196	5001	4	4166	3
202	196	5001	4	4167	4
203	196	5001	4	41691	1
204	196	5001	4	41692	4919147515.000
205	196	5001	4	41693	1000
206	165	5001	4	4161	
207	206	5001	4	4162	2
208	206	5001	4	4163	3
209	206	5001	4	4164	2
210	206	5001	4	4165	3
211	206	5001	4	4166	3
212	206	5001	4	4167	5
213	206	5001	4	41691	1
214	206	5001	4	41692	4919147515.000
215	206	5001	4	41693	1000

The table below continues the links from the Technical Profile

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
216	165	5001	4	4161	
217	216	5001	4	4162	2
218	216	5001	4	4163	22
219	216	5001	4	4164	2
220	216	5001	4	4165	31
221	216	5001	4	4166	3
222	216	5001	4	4167	1
223	216	5001	4	41691	1
224	216	5001	4	41692	4919147515.000
225	216	5001	4	41693	1000
226	165	5001	4	4161	
227	226	5001	4	4162	2
228	226	5001	4	4163	22
229	226	5001	4	4164	2
230	226	5001	4	4165	31
231	226	5001	4	4166	3
232	226	5001	4	4167	2
233	226	5001	4	41691	1
234	226	5001	4	41692	4919147515.000
235	226	5001	4	41693	1000
236	165	5001	4	4161	
237	236	5001	4	4162	2
238	236	5001	4	4163	22
239	236	5001	4	4164	2
240	236	5001	4	4165	31
241	236	5001	4	4166	3
242	236	5001	4	4167	3
243	236	5001	4	41691	1
244	236	5001	4	41692	4919147515.000
245	236	5001	4	41693	1000

The table below continues links for the Technical Profile and includes an Object Element for house number.

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
246	165	5001	4	4161	
247	246	5001	4	4162	2
248	246	5001	4	4163	22
249	246	5001	4	4164	2
250	246	5001	4	4165	31
251	246	5001	4	4166	3
252	246	5001	4	4167	4
253	246	5001	4	41691	1
254	246	5001	4	41692	4919147515.000
255	246	5001	4	41693	1000
256	165	5001	4	4161	
257	256	5001	4	4162	2
258	256	5001	4	4163	22
259	256	5001	4	4164	2
260	256	5001	4	4165	31
261	256	5001	4	4166	3
262	256	5001	4	4167	5
263	256	5001	4	41691	1
264	256	5001	4	41692	4919147515.000
265	256	5001	4	41693	1000
266	121	5001	4	4131	
267	266	5001	4	4132	2
268	266	5001	4	4133	3
269	266	5001	4	4134	3
270	266	5001	4	4135	1
271	266	5001	4	4137	3
272	266	5001	4	4138	2
273	266	5001	4	4139	2
274	266	5001	4	4140	
275	274	5001	4	4141	
276	275	5001	4	4142	2
277	275	5001	4	4143	3
278	275	5001	4	4144	House Number (Object Element)

The table below shows the Object Elements for the Street and the City

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
279	266	5001	4	4150	
280	279	5001	4	4151	
281	280	5001	4	4152	2
282	280	5001	4	4153	9
283	280	5001	4	4154	3
284	280	5001	4	4155	1
285	121	5001	4	4131	
286	285	5001	4	4132	2
287	285	5001	4	4133	3
288	285	5001	4	4134	3
289	285	5001	4	4135	2
290	285	5001	4	4137	4
291	285	5001	4	4138	2
292	285	5001	4	4139	2
293	285	5001	4	4140	
294	293	5001	4	4141	
295	294	5001	4	4142	2
296	294	5001	4	4143	3
297	294	5001	4	4144	Street (Object Element)
298	285	5001	4	4150	
299	298	5001	4	4151	
300	299	5001	4	4152	2
301	299	5001	4	4153	9
302	299	5001	4	4154	3
303	299	5001	4	4155	2
304	121	5001	4	4131	
305	304	5001	4	4132	2
306	304	5001	4	4133	3
307	304	5001	4	4134	3
308	304	5001	4	4135	3
309	304	5001	4	4137	5
310	304	5001	4	4138	2
311	304	5001	4	4139	2
312	304	5001	4	4140	
313	312	5001	4	4141	
314	313	5001	4	4142	2
315	313	5001	4	4143	3
316	313	5001	4	4144	City (Object Element)

The table below shows the Object Elements for State/Province and Postal Code

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
317	304	5001	4	4150	
318	317	5001	4	4151	
319	318	5001	4	4152	2
320	318	5001	4	4153	9
321	318	5001	4	4154	3
322	318	5001	4	4155	3
323	121	5001	4	4131	
324	323	5001	4	4132	2
325	323	5001	4	4133	3
326	323	5001	4	4134	3
327	323	5001	4	4135	4
328	323	5001	4	4137	6
329	323	5001	4	4138	2
330	323	5001	4	4139	2
331	323	5001	4	4140	
332	331	5001	4	4141	
333	332	5001	4	4142	2
334	332	5001	4	4143	3
335	332	5001	4	4144	State (Object Element)
336	323	5001	4	4150	
337	336	5001	4	4151	
338	337	5001	4	4152	2
339	337	5001	4	4153	9
340	337	5001	4	4154	3
341	337	5001	4	4155	4
342	121	5001	4	4131	
343	342	5001	4	4132	2
344	342	5001	4	4133	3
345	342	5001	4	4134	3
346	342	5001	4	4135	5
347	342	5001	4	4137	7
348	342	5001	4	4138	2
349	342	5001	4	4139	2
350	342	5001	4	4140	
351	350	5001	4	4141	
352	351	5001	4	4142	2
353	351	5001	4	4143	3
354	351	5001	4	4144	Postal Code (Object Element)

The table below shows Data Elements for House Number and Street

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
355	342	5001	4	4150	
356	355	5001	4	4151	
357	356	5001	4	4152	2
358	356	5001	4	4153	9
359	356	5001	4	4154	3
360	356	5001	4	4155	5
361	121	5001	4	4131	
362	361	5001	4	4132	2
363	361	5001	4	4133	31
364	361	5001	4	4134	3
365	361	5001	4	4135	1
366	361	5001	4	4137	8
367	361	5001	4	4138	2
368	361	5001	4	4139	2
369	361	5001	4	4140	
370	369	5001	4	4141	
371	370	5001	4	4142	2
372	370	5001	4	4143	31
373	370	5001	4	4144	House Number DE (Data Element)
374	369	5001	4	4141	
375	374	5001	4	4142	2
376	374	5001	4	4143	214
377	374	5001	4	4144	6
378	121	5001	4	4131	
379	378	5001	4	4132	2
380	378	5001	4	4133	31
381	378	5001	4	4134	3
382	378	5001	4	4135	2
383	378	5001	4	4137	9
384	378	5001	4	4138	2
385	378	5001	4	4139	2
386	378	5001	4	4140	
387	386	5001	4	4141	
388	387	5001	4	4142	2
389	387	5001	4	4143	31
390	387	5001	4	4144	Street DE (Data Element)

The table below show data element for city

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
391	86	5001	4	4141	
392	391	5001	4	4142	2
393	391	5001	4	4143	214
394	391	5001	4	4144	40
395	121	5001	4	4131	
396	395	5001	4	4132	2
397	395	5001	4	4133	31
398	395	5001	4	4134	3
399	395	5001	4	4135	3
400	395	5001	4	4137	10
401	395	5001	4	4138	2
402	395	5001	4	4139	2
403	395	5001	4	4140	
404	403	5001	4	4141	
405	404	5001	4	4142	2
406	404	5001	4	4143	31
407	404	5001	4	4144	City DE (Data Element)
408	403	5001	4	4141	
409	408	5001	4	4142	2
410	408	5001	4	4143	214
411	408	5001	4	4144	30
412	395	5001	4	4150	
413	412	5001	4	4151	
414	413	5001	4	4152	2
415	413	5001	4	4153	10
416	413	5001	4	4154	3
417	413	5001	4	4155	3
418	412	5001	4	4151	
419	418	5001	4	4152	2
420	418	5001	4	4153	11
421	418	5001	4	4154	2
422	418	5001	4	4155	2

The table below shows the Data Elements for State/Province and Postal Code

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
423	121	5001	4	4131	
424	423	5001	4	4132	2
425	423	5001	4	4133	31
426	423	5001	4	4134	3
427	423	5001	4	4135	4
428	423	5001	4	4137	11
429	423	5001	4	4138	2
430	423	5001	4	4139	2
431	423	5001	4	4140	
432	431	5001	4	4141	
433	432	5001	4	4142	2
434	432	5001	4	4143	31
435	432	5001	4	4144	State DE (Data Element)
436	431	5001	4	4141	
437	436	5001	4	4142	2
438	436	5001	4	4143	214
439	436	5001	4	4144	5
440	121	5001	4	4131	
441	440	5001	4	4132	2
442	440	5001	4	4133	31
443	440	5001	4	4134	3
444	440	5001	4	4135	5
445	440	5001	4	4137	12
446	440	5001	4	4138	2
447	440	5001	4	4139	2
448	440	5001	4	4140	
449	448	5001	4	4141	
450	449	5001	4	4142	2
451	449	5001	4	4143	31
452	449	5001	4	4144	Postal Code (Data Element)
453	448	5001	4	4141	
454	453	5001	4	4142	2
455	453	5001	4	4143	214
456	453	5001	4	4144	10

The table below shows the values from the Address object being queried.

SegId	Parent	Prio	ElemZTIC	ElemCode	Value
457	115	5001	4	410	
458	457	5001	4	4100	ObjectQset
459	457	5001	4	4104	1
460	457	5001	4	4105	
461	457	5001	4	4106	
462	457	5001	4	4130	
463	462	5001	4	4131	
464	463	5001	4	4132	3
465	463	5001	4	4133	1
466	463	5001	4	4134	12
467	463	5001	4	4135	57
468	463	5001	4	4137	1
469	463	5001	4	4138	0
470	463	5001	4	4139	0
471	463	5001	4	4140	
472	471	5001	4	4141	
473	472	5001	4	4142	3
474	472	5001	4	4143	1
475	472	5001	4	4144	101
476	471	5001	4	4141	
477	476	5001	4	4142	3
478	476	5001	4	4143	2
479	476	5001	4	4144	Main Street
480	471	5001	4	4141	
481	480	5001	4	4142	3
482	480	5001	4	4143	3
483	480	5001	4	4144	Anytown
484	471	5001	4	4141	
485	484	5001	4	4142	3
486	484	5001	4	4143	4
487	484	5001	4	4144	ZZ
488	471	5001	4	4141	
489	488	5001	4	4142	3
490	488	5001	4	4143	5
491	488	5001	4	4144	00011

Appendix

Appendix 1--Object Kinds

Listed below are the Object Kinds that are included in the Base Specification

Base Object Kinds	
Code	**Description**
1	Object Kind (Object Kind)

2	Object Template (Object Kind)
3	Object Element (Object Kind)
4	Table Element (Object Kind)
5	Type Value Based Template Rule (Object Kind)
6	Link Based Template Rule (Object Kind)
7	Object Element Value Based Template Rule (Object Kind)
8	Object Kind Type (Object Kind)
9	Object Type Definition (Object Kind)
10	Link Type (Object Kind)
11	Link Target Type (Object Kind)
12	Technical Profile (Object Kind)
13	Object Set Definition (Object Kind)
14	ZT Instance (Object Kind)
15	Namespace (Object Kind)
16	Object Element Usage Type (Object Kind)
17	Object Template Type (Object Kind)
18	Language (Object Kind)
19	User (Object Kind)
20	User Type (Object Kind)
21	Code Range (Object Kind)
22	Code Scheme (Object Kind)
23	Code Scheme Type (Object Kind)
24	System Message Definition (Object Kind)
25	System Message Definition Type (Object Kind)
26	Parameter (Object Kind)
27	Parameter Type (Object Kind)
28	System Message Definition Category (Object Kind)
29	System Message Parameter Category (Object Kind)
30	Table (Object Kind)
31	Data Element (Object Kind)
32	Data Type (Object Kind)
33	Extended Key Definition (Object Kind)
34	Extended Key Definition Type (Object Kind)
35	Object Element Usage Type (Object Kind)
36	Function (Object Kind)
37	Class/Program (Object Kind)
38	Class/Program Type (Object Kind)
39	Resource (Object Kind)
40	Resource Protcol (Object Kind)
41	Resource Name Extension (Object Kind)
42	Software Runtime Profile (Object Kind)
43	Software Runtime Component (Object Kind)
44	Patch Level (Object Kind)
45	Language & Patch Level Composite Extended Key (Object Kind)
46	Object Set Definition Type (Object Kind)
47	Link Type Usage Type (Object Kind) i.e. standard, object sequencing
48	Type Value Based Object Set (Object Kind)
49	Template Set (Object Kind)
50	Generic Object (Object Kind)
51	Generic Object Element Value (Object Kind)
52	Function Group (Object Kind)
53	Function Group Type (Object Kind)
54	Parameter Value Group (Object Kind)
55	Object Template Path (Object Kind)
56	Type Definition Path (Object Kind)
57	Parameter Value (Object Kind)
58	Statistical Values Object (Object Kind)
59	Statistical Values Dimension (Object Kind)
60	Statistic Type (Object Kind) sum, count, mean, mode, median, std dev
61	Unit of Measure (Object Kind)
62	Unit of Measure Dimension (Object Kind) length, weight, time, currency
63	Type Definition Usage Type (Object Kind)
64	System Message Element Definition (Object Kind)
65	System Message Element Definition Type (Object Kind)
66	Numeric Data Type Details (Object Kind)

Appendix 2--Object Elements

Listed below are the Object Elements in the base specification.

Object Elements in Base Namespace	
Code	**Description**
1	Object Kind Description
2	Object Template Description
3	Object Element Description
4	Table Element Description
5	Type Value Based Template Rule Description
6	Link Based Template Rule Description
7	Object Element Value Based Template Rule Description
8	Object Kind Type Description
9	Object Type Definition Description
10	Link Type Description
11	Link Target Type Description
12	Technical Profile Description
13	Object Set Definition Description
14	ZT Instance Description
15	Namespace Description
16	Object Element Usage Type Description
17	Object Template Type Description
18	Language Description
19	User Description
20	User Type Description
21	Code Range Description
22	Code Scheme Description
23	Code Scheme Type Description
24	System Message Definition Description
25	System Message Definition Type Description
26	Parameter Description
27	Parameter Type Description
28	System Message Definition Category Description
29	System Message Parameter Category Description
30	Table Description
31	Data Element Description
32	Data Type Description
33	Extended Key Definition Description
34	Extended Key Definition Type Description
35	Object Element Usage Type Description
36	Funtion Description
37	Class/Program Description
38	Class/Program Type Description
39	Resource Description
40	Resource Protocol Description
41	Resource Name Extension Description
42	Software Runtime Profile Description
43	Software Runtime Component Description
44	Patch Level Description
45	Language & Patch Level Composite Extended Key Description
46	Object Set Definition Type Description
47	Link Type Usage Type Description
48	Type Value Based Object Set Description
49	Template Set Description
50	Generic Object Description
51	Generic Object Element Value Description
52	Function Group Description
53	Function Group Type Description
54	Parameter Value Group Description
55	Object Template Path Description
56	Type Definition Path Description
57	Parameter Value Description
58	Statistical Values Object Description
59	Statistical Values Dimension Description
60	Statistic Type Description sum, count, mean, mode, median, std dev
61	Unit of Measure Description
62	Unit of Measure Dimension Description length, weight, time, currency
63	Type Definition Usage Type Description
64	System Message Element Definition Description

65	System Message Element Definition Type Description
66	Numeric Data Type Details Description
200	System Message Short Text
201	System Message Long Text
202	Minimum value for Code Range
203	Maximum value for Code Range
204	Next Code for Code Range
205	Code string for Code Scheme
206	Numeric Preferred for Code Range
207	Code Prefix
208	Leading Character for Code Range
209	Leading Character for Numeric Codes
210	Resource Path/Name
211	Resource Path/Name--Patch Level Dependent
212	Resource Path/Name--Language Dependent
213	Resource Path/Name--Patch Level and Language Dependent
214	Data Element--Maximum Length
215	Namespace
216	Dimension Label
217	Index for Type Definition for Summarization in Type Definition Path used by Dimension
218	Index for Type Definition for Grouping in Type Defiition Path used for Dimension
219	Index for Type Definition for Groupin2 in Type Definition Path used for Dimension
220	Namespace of Object Kind of Object Represented by a Generic Object
221	Code of Object Kind of Object Represented by a Generic Object
222	Namespace of Object Represented by a Generic Object
223	Code of Object Represented by a Generic Object
224	Start Index in a Type Definition Path for a Type Value Based Object Set
225	Stop Index in a Type Definition Path for a Type Value Based Object Set
226	System Message--Short Boilerplate Text
227	System Message--Long Boilerplate Text

Appendix 3--Type Definitions

Listed below are Type Definitions in the base specification

Type Definitions in Base Namespace	
Code	**Description**
1	Type Definition: Object Element Usage Type for a Object Element
2	Type Definition: Link Target Type for Link Type
3	Type Definition: Object Kind for Object Template
4	Type Definition: Object Kind for Type Definition
5	Type Definition: Code Range for Object Kind
6	Type Definition: Code Scheme for Code Range
7	Type Definition: Set of Functions for Software Runtime Component
8	Type Definition: Patch Level for Software Runtime Component
9	Type Definition: Data Element for Object Element
10	Type Definition: Data Type for Data Element
11	Type Definition: Extended Key Definition for Data Element
12	Type Definition: Extended Key Definition Type for Extended Key Definition
13	Type Definition: Type Definition for Extended Key Definition
14	Type Definition: Object Set for a Type Definition for allowed Type Values
15	Type Definition: Increment Function for Code Scheme
16	Type Definition: Class/Program for Function
17	Type Definition: Patch Level for a Runtime Profile Component
18	Type Definition: Set of Classses/Programs for a Runtime Profile Component
19	Type Definition: Set Type for a Set
20	Type Definition: Object Template for Extended Key Definition using Composite Key
21	Type Definition: Object Kind Type for Object Kind (standard, composite key etc)
22	Type Definition: Default Object Template for a Object Kind--Minimal Data (to return values)
23	Type Definition: Default Object Template for a Object Kind--Full Data (to return values)
24	Type Definition: Link Type for a Type Definition to create a bi-directional link (1 server generated + 1 type def)
25	Type Definition: Object Template for a Type Definition (to select values of a type value)
26	Type Definition: Link Type Usage Type for a Link Type
27	Type Definition: Function Group Type for a Function Group
28	Type Definition: Template Type for a Template
29	Type Definition: Type Definition Usage Type for a Type Definition

30	Type Definition: Reference Template for a Template
31	Type Definition: Parameter for a Parameter Value
32	Type Definition: Statistic Type for a Statistical Values Object (sum, count, mode, mean, std deviation)
33	Type Definition: Object Set for a Statistical Values Object
34	Type Definition: Type Definition Path for a Statistical Values Dimension
35	Type Definition: Unit of Measure for a Statistical Values Object
36	Type Definition: Unit of Measure Dimension for a Unit of Measure
37	Type Definition: Object Element for a Statistical Values Object for source of data
38	Type Definition: Object Template for a Type Based Template Rule
39	Type Definition: Object Set for a Type Value Based Template Rule
40	Type Definition: Object Set for Input Objects for a Type Values Based Object Set
41	Type Definition: Object Set for Type Values for a Type Value Based Object Set
42	Type Definition: Type Defininition Path for a Type Value Based Object Set
43	Type Definition: Object Template for a Link Based Template Rule
44	Type Definition: Object Set for a Link Based Template Rule
45	Type Definition: Object Template for an Object Element Value Based Template Rule
46	Type Definition: Object Set for an Object Element Value Based Template Rule
47	Type Definition: Object Element for an Object Element Value Based Template Rule
48	Type Definition: Object Set for a Data Element for validation of character values
49	Type Definition: Object Element for a Data Element for validation of character values

Appendix 4--Link Types

Listed below are Link Types in the base specification

List of Link Types in Base Namespace	
Code	**Description**
1	Link Type: Object Template to Technical Profile
2	Link Type: Technical Profile to Technical Profile
3	Link Type: Technical Profile to Object Elements
4	Link Type: Technical Profile to Type Definitions
5	Link Type: Technical Profile to LinkTypes
6	Link Type: Link Target Type to Object Template
7	Link Type: System Message Definition to Parameter
8	Link Type: Function to Parameter
9	Link Type: Class to Resource
10	Link Type: Server Software Runtime Profile to Server Software Runtime Component
11	Link Type: Server Software Runtime Component to Server Software Runtime Component
12	Link Type: Set Definition to an Object Kind to include all objects of that Kind as members
13	Link Type: Set Definition to individual objects to include as members
14	Link Type: Set Definition to Type Value Based Object Set, to include all objects with corresponding type values as members
15	Link Type: Set Definition to Template Set to include all objects using a Template in the Template Set as members
16	Link Type: Set Definition to Set Definition of Type, Link Target to include all objects linked to the targets as members
17	Link Type: Set Definition to Set Def. of Type Object Element Value to include objects with the object element value as members
18	Link Type: Set Definition to Set Definition to include target set members as members of source set (OR Condition)
19	Link Type: Set Definition to Set Def. to include target set members as members if also included in source set (AND Condition)
20	Link Type: Set Definition to Set Def. to remove target set members as members of the source set (NOT Condition)
21	Link Type: Set Definition to Custom Function to determine which objects will be members
22	Link Type: Technical Profile to Data Elements
23	Link Type: Set Definition to Namespace where all objects that have an Object Kind with the ZTNS linked to are included as members
24	Link Type: Set Definition to Namespace where all objects that have an Object ZTNS linked to are included as members
25	Link Type: Technical Profile to Technical Profile to remove the attributes of the target from the Tech Profile of the source
26	Link Type: Statistical Values Object to Statistical Values Dimension
27	Link Type: Type Definition Path to Type Definition
28	Link Type: Data Element to a Function to determine the value of Object Elements that use the Data Element
29	Link Type: Type Definition to a Type Value Based Template Rule
30	Link Type: Set Definition to Generic Objects to include the object referred to by the generic object as a member
31	Link Type: Function Group to a System Message Definition
32	Link Type: Function Group to a Function
33	Link Type: Link Type to Link Based Template Rule
34	Link Type: Object Element to Object Element Value Based Template Rule

Appendix 5--Table Elements

Listed below are Table Elements in the base specification

List of Base Table Elements for Objects	
Code	**Description**
1	Object Kind ZTIC
2	Object Kind Code
3	Object Code ZTIC
4	Object Code
5	Record Type ZTIC
6	Record Type Code
7	Object Element ZTIC
8	Object Element Code
9	Extended Key Definition ZTIC
10	Extended Key Definition Code
11	Extended Key Value ZTIC
12	Extended Key Value Code
13	Type Definition ZTIC
14	Type Definition Code
15	Type Value ZTIC
16	Type Value Code
17	Link Type ZTIC
18	Link Type Code
19	Link To Kind ZTIC
20	Link To Kind Code
21	Link To Code ZTIC
22	Link To Code
23	Object Template ZTIC
24	Object Template Code
25	Status
26	Timestamp--Effective
27	Sequence Number
28	Message ID
29	Message Segment Index
30	Link Value
31	Object Element Value
32	Server DB--ZT Instance ID

Appendix 6--Standard Message Definition

The standard Message Definition is shown below. Each Message Element is identified by a numeric code. The level of indentation on the left side of the page shows the hierarchical position of each Message Element. The description for the Message Elements is on the right side of the page.undefined

1: Root	1: Root is at the top of the Message Definition Heirarchy and it the direct or indirect parent of all Message Elements in the Message
2: Header	2: Header is at the top of the Header section of the Message Defintion. It is populated primarily by the Client but can also be updated by the Server, for example, to add to ZT Instance Codes and related Namespaces when the Response refers to Namespaces that are not included in the Request
---20: Message Definiton	20: Message Defintion includes the Message Elements used to declare the Message Definition version and Template
---|---200: Message Definition version ZT Instance Code	200: Message Definition version ZT Instance Code identifies the source Instance of the Message Definition. The ZT Instance Code used here needs to be declared in Message Element 2100 along with the related Namespace in Message Element 2101. Typically this would be the base instance and Namespace of ZT Specification.
---|---201: Message Definition version Code	201: Message Definition version Code is used to declare the version of the Message Definition used by the Message
---|---202: Message Definition version Template ZT Instance Code	202: Message Definition version Template ZT Instance Code identifies the source Instance of Template use for the Message Definition. The ZT Instance Code use here needs to be declare int Messge Element 2100 along with the related Namespace in Message Element 2101.
---|---203: Message Definition version Template Code	203: Message Definition version Template Code is used to identify the template for the Message Definition version used to for this message
---21: ZTISet	21: ZTISet contains all of the ZT Instances declared for the Message and their related Namespaces
---|---210: ZT Instance	210: ZT Instance includes the values for one occurance of ZT Instance Code and its Namespace
---|---|---2100: Code	2100: Code is used to declare one DS Instance Code within a ZT Instance Set
---|---|---2101: Namespace	2101: Namespace is used to delare the Namespace associated with the ZT Instance Code in Message Element 2100
---22: Message Partners	22: Message Partners contains identifiers for the sender and receiver of a message
---|---220: SenderServerZTICode	
---|---221: SenderZTICode	

---|---222: ReceiverServerZTICode	
---|---223: ReceiverZTICode	223: Receiver ZT Instance Code is a reference to Message Element 2100 and the related Namespace in Message Element 2101. The server uses this to find the target Table Name Prefix, Table Element 32, for storing or retrieving data
---23: MessageProcessingParameters	23: Message Processing Parameters contains values that tell the server how to process the message
---|---230: General	230: General is a section of the Message that contains General Processing Parameters
---|---|---2301: DefaultEffectiveTimeStamp	2301: DefaultEffectiveTimeStamp is the timestamp used for processing if the timestamp is not specified at a more detailed level of the message
---|---|---2302: MessageIdClient	2302: MessageIdClient is a message identifier that is assigned by the client and retained by the server. Each message sent by the client should have a unique id assigned by the client
---|---|---2303: IndexOfLastRequestSegment	2303: IndexOfLastRequestSegment contains the last index used by a client after building a request message. Subsequent updates to the message are by the server, even updates to the header which is primarily built by the client. An example is an update of the header status by the server
---|---|---2304: DefaultUpdateMode--TO BE DEPRECATED	2304: DefaultUpdateMode--TO BE DEPRECATED is used for update mode if it is not assigned at a more detailed level of the message
---|---|---23040: DefaultUpdateModeZTIC	23040: DefaultUpdateModeZTIC is the ZTIC of update mode used if it is not assigned at a more detailed level of the message
---|---|---23041: DefaultUpdateModeCode	23041: DefaultUpdateModeCode is the Code of the update mode used if it is not assigned at a more detailed level
---|---|---23050: User ZTIC	23050: ZT Instance Code of User making request
---|---|---23051: User Code	23051: User Code for User making request
---|---|---2306: ServerSoftwareRuntimeProfileZTICode	2306: ServerSoftwareRuntimeProfileZTICode is the DS Instance Code of the preferred runtime software profile for processing of the message
---|---|---2307: ServerSoftwareRuntimeProfileCode	2307: ServerSoftwareRuntimeProfileCode is the Code of the preferred runtime software profile for processing of the message
---|---231: ExtendedKeySet	231: ExtendedKeySet is where extended keys are declared for the message
---|---|---2310: ExtendedKey	2310: ExtendedKey is the parent for an individual instance of an Extended Key
---|---|---|---2311: ExtendedKeyDefinitionCodeZTIC	2311: ExtendedKeyDefinitionCodeZTIC is the Code ZTIC for an Extentended Key
---|---|---|---2312: ExtendedKeyDefinitionCode	2312: ExtendedKeyDefinitionCode is the Code for an Extended Key
---|---|---|---2313: ExtendedKeyValueDSIC	2313: ExtendedKeyValueDSIC is the ZTIC for an Extended Key value
---|---|---|---2314: ExtendedKeyValueCode	2314: ExtendedKeyValueCode is the Code for an Extended Key value
---|---233: QueueProcessing	233: QueueProcessing contains the parameters required for queue processing of messages if this is supported by the server
---|---|---2331: SerializationActive	2331: SerializationActive is a boolean value that activates serialization of messages if supported by the server
---|---|---2332: QueueName	2332: QueueName identifies a unique queue, assigned by the client for queue sequential processing of messages
---|---|---2333: SerializationKey	2333: SerializationKey determines the sequence of this message for queue processing, it is a positve integer
3: Request	3: Request is the parent to all Message Elements in the Request section of the message
---30: MaintainSet	30: MaintainSet is a child of Request and contains Message Elements use for creating or changing objects
---|---300: Maintain	300: Maintain holds the values used to create or change object, the Message Elements contained here would share the same parameters
---|---|---3000: ID	3000: ID Identifies a Maintain. It is populated by the client and returned by the server in the response using Message Element 4000
---|---|---3003: DefaultEffectiveTimestamp	3003: DefaultEffectiveTimestamp is the default timestamp for database updates, used if not specified at a more detailed level of the message
---|---|---3008: UpdateMode -- to be DEPRECATED	3008: UpdateMode -- to be DEPRECATED determines how data is processed. mode 1 is standard, mode 99 has no validation and is used from server initialization
---|---|---30080: UpdateModeZTIC	30080: UpdateMode ZTIC
---|---|---30081: UpdateModeCode	30081: UpdateMode determines how data is processed. mode 1 is standard, mode 99 has no validation and is used from server initialization
---|---|---3030: ObjectSet	3030: ObjectSet contains a group of objects to be maintained (created or changed)
---|---|---|---3031: Object	3031: Object contains the data needed to maintain one instance of an object
---|---|---|---|---3032: ObjectKindZTICode	3032: ObjectKindZTICode is the ZT Instance Kind Code for an object being maintained
---|---|---|---|---3033: ObjectKindCode	3033: ObjectKindCode is the Kind Code for an object being maintained
---|---|---|---|---3034: ObjectDSICode	3034: ObjectDSICode is the ZT Instance Code for the object being maintained
---|---|---|---|---3035: ObjectCodeTemp	3035: ObjectCodeTemp is code assigned in a message before it has been assigned by the server
---|---|---|---|---3036: NewCode (boolean)	3036: NewCode (boolean) is used to indicate if the server should assign a new code for the object being maintained
---|---|---|---|---|---3037: ID	3037: ID is used to identify an object, assigned by the client and provided back in the response
---|---|---|---|---|---3038: ParentID	
---|---|---|---|---|---30390: TemplateZTIC	30390: TemplateZTIC has the template DS Instance Code of the object being maintained
---|---|---|---|---|---30391: TemplateCode	30391: TemplateCode has the template Code of the object being maintained
---|---|---|---|---|---30392: ObjectStatus	30392: ObjectStatus is status of complete object
---|---|---|---|---|---3040: ElementSet	3040: ElementSet contains the element values for an object being maintained
---|---|---|---|---|---|---3041: Element	3041: Element contains the values for one element
---|---|---|---|---|---|---|---3042: ObjectElementZTICode	3042: ObjectElementZTICode is the Element ZT Instance Code for the object element being maintained
---|---|---|---|---|---|---|---3043: ObjectElementCode	3043: ObjectElementCode is the Element Code for the object being maintained
---|---|---|---|---|---|---|---3044: ObjectElementValue	3044: ObjectElementValue is the value of the element being maintained
---|---|---|---|---|---|---|---3046: ObjectElementStatus	3046: ObjectElementStatus is the status for ObjectElementValue
---|---|---|---|---|---|---|---3047: EffectiveTimeStamp	3047: EffectiveTimeStamp is the effective timestamp for Object ElementValue
---|---|---|---|---|---3050: TypeSet	3050: TypeSet contains the type values for the object being maintained

---|---|---|---|---|---|---3051: Type	3051: Type contains an individual type value for the object being maintained
---|---|---|---|---|---|---3052: TypeDefinitionZTICode	3052: TypeDefinitionZTICode is type definition DS Instance Code for the object being maintained
---|---|---|---|---|---|---3053: TypeDefinitionCode	3053: TypeDefinitionCode is the type definition Code for the object being maintained
---|---|---|---|---|---|---3054: TypeValueZTICode is the ZT Instance Code of a type value for the object being maintained	
---|---|---|---|---|---|---3055: TypeValueCode is the Code of a type value for the object being maintained	
---|---|---|---|---|---|---3056: TypeValueStatus	3056: TypeValueStatus is the status for the Type Value
---|---|---|---|---|---|---3057: EffectiveTimeStamp	3057: EffectiveTimeStamp is the effective timestamp for the Type Value
---|---|---|---|---3060: LinkSet	3060: LinkSet contains the link values for the object being maintained
---|---|---|---|---|---3061: Link	3061: Link contains an individual link for the object being maintained
---|---|---|---|---|---|---3062: LinkTypeZTIC	3062: LinkTypeZTIC has the DS Instance Code of the link type for the link of the object being maintained
---|---|---|---|---|---|---3063: LinkTypeCode	3063: LinkTypeCode has the Code of the link type for the link of the object being maintained
---|---|---|---|---|---|---3064: LinkToKindZTIC	3064: LinkToKindZTIC has the ZT Instance Code of the kind of the object being linked to for the object being maintained
---|---|---|---|---|---|---3065: LinkToKindCode	3065: LinkToKindCode has the Kind Code of the object being linked to for the object being maintained
---|---|---|---|---|---|---3066: LinkToZTIC	3066: LinkToZTIC has the ZT Instance Code of the object being linked to for the object being maintained
---|---|---|---|---|---|---3067: LinkToCode	3067: LinkToCode has the ZT Code for the object being linked to for the object being maintained
---|---|---|---|---|---|---3068: NewLinkTargetCode (boolean)	3068: NewLinkTargetCode is a boolean that indicates that this is a link to an object that will be assigned a new code on the server
---|---|---|---|---|---|---30691: LinkStatus	30691: Status Code for the link
---|---|---|---|---|---|---30692: EffectiveTimeStamp	30692: EffectiveTimeStamp is the effective timestamp for the link
---|---|---|---|---|---|---30693: LinkValue	30693: LinkValue indicates the strength or priority of the link
---31: QuerySet	31: QuerySet contains the values needed to query objects from a ZT Server
---|---310: Query	310: Query contains the values to query objects for a specific template
---|---|---3100: ID	3100: ID contains and identifier assigned by the client and returned by the server in the response
---|---|---3101: ObjectTemplateZTIcode	3101: ObjectTemplateZTIcode is the ZT Instance Code the template being used in a query
---|---|---3102: ObjectTemplateCode	3102: ObjectTemplateCode is the Code of the template being used in a query
---|---|---31030: SelectionModeZTIC	31030: SelectionModeZTIC contains ZTIC of the Selection Mode used to indicate how selection will be done
---|---|---31031: SelectionModeCode	31031: SelectionModeCode contains Code of the Selection Mode used to indicate how selection will be done
---|---|---31032: StatusExclusionSet	31032: StatusExclusionSet is the parent for a set of excluded status codes for the response message
---|---|---|---31033: ExcludedStatus	31033: ExcludedStatus is a object status code that is to be excluded from the response message
---|---|---3104: EffectiveTimestampSet	3104: Effective Timestamp Set holds a set of timestamps the used for selection of data from the server
---|---|---|---31040: EffectiveTimestamp	31040: Effective Timestamp is used to control which data is select from a server
---|---|---|---|---31041: Timestamp Operator	31041: Timestamp Operator is used to indicate ranges for a timestamp using for example < or >
---|---|---|---|---31042: Timestamp value	31042: Timestamp value, the value of an individual timestamp or the low or high value for a range of timestamps
---|---|---3105: QueryResponseControlParameterSet	3105: QueryResponseControlParameterSet holds parameters that control how many objects and the content of objects returned in a response
---|---|---|---31050: QueryResponseControlParameter	31050: QueryResponseControlParameter contains the reponse control parameters for a specific response type
---|---|---|---|---31051: ResponseTypeZTIC	31051: ResponseTypeZTIC is the ZT Instance Code for the response type
---|---|---|---|---31052: ResponseTypeCode	31052: ResponseTypeCode indicates type of response, 1=standard, 2=Template of a Linkable Object, 3=permitted type values, 4=permitted link-to values, 5=object set members for sets that are linked to and 6=object set members for sets that are type values of queried objects
---|---|---|---|---31053: RelatedObject_ObjectKindZTIC	31053: RelatedObject_ObjectKindZTIC is the ZT Instance Code of the object kind of a related object used for response control
---|---|---|---|---31054: RelatedObject_ObjectKindCode	31054: RelatedObject_ObjectKindCode is the Code of the object kind of a related object used for response control
---|---|---|---|---31055: RelatedObject_ObjectZTIC	31055: RelatedObject_ObjectZTIC is the ZT Instance Code of a related object used for response control
---|---|---|---|---31056: RelatedObject_ObjectCode	31056: RelatedObject_ObjectCode is the Code of a related object used for response control, usually a set, a type definition or a link type
---|---|---|---|---31060: MaximumNumberOfObectsReturned	31060: MaximumNumberOfObjectReturned is the maximum number of objects returned for a response type
---|---|---|---|---31061: MaximumHierarchyLevelsReturned	31061: MaximumHierarchyLevelsReturned indicates how deep in the hierarchy to return linked objects
---|---|---|---|---31062: OffsetFromBeginningObject	31062: OffsetFromBeginningObject is the number of objects after the starting object to return to the client
---|---|---|---|---31063: SortObjectZTIC	31063: SortObjectZTIC is the ZT Instance Code for the sort object used to sort objects returned to the client
---|---|---|---|---31064: SortObjectCode	31064: SortObjectCode is the Code of the sort object used to sort objects returned to the client
---|---|---|---|---31065: SetMemberResponseMode	31065: SetMemberResponseMode indicates how much data is returned for set members 1=minimal data, 2=maximal data
---|---|---3107: SelectionParameterValueSet	3107: SelectionParameterValueSet hold parameters that can be passed to the server, can be used by Object Elements that have calculated values
---|---|---|---31070: SelectionParameterValue	31070: SelectionParameterValue contains the identifiers and value for a parameter value
---|---|---|---|---31071: ParameterValueGroupZTIC	31071: ParameterValueGroupZTIC is the ZT Instance Code used to identify a group of parameters that could for example be passed to calculate and object element value
---|---|---|---|---31072: ParameterValueGroupCode	31072: ParameterValueGroupCode is the Code to identify a group of parameters that could be passed to calculate an object element value
---|---|---|---|---31073: ParameterZTIC	31073: ParameterZTIC is the ZT Instance Code for a parameter
---|---|---|---|---31074: ParameterCode	31074: ParameterCode is the Code for a parameter
---|---|---|---|---31075: ParameterValueRowCounter	31075: ParameterValueRowCounter is a numeric counter used whith parameter values with multiple

	RecIdKdsicKcodeOdsicOcodeEdsicEcodeTdsicTcodeNewCdOEValue
---|---|---|---|---31076: ParameterValue	31076: ParameterValue is the value for a parameter passed to the server for a query
---|---|---3110: SelectionSet	3110: SelectionSet contains the values needed to select objects from the server
---|---|---|---3111: Selection	3111: Selection contains the values needed to select a specific set of objects from the server, possibly combined with other sets
---|---|---|---|---3112: SelectionGroupNumber	3112: SelectionGroupNumber identifies a group used in combining selection sets
---|---|---|---|---3113: NumberOfSelectionGroupParent	3113: NumberOfSelectionGroupParent identifies the parent group, used to combine selection sets
---|---|---|---|---3114: SetOperator	3114: SetOperator is a set operator used to combiine selection sets
---|---|---|---|---3120: ObjectSelectionSet	3120: ObjectSelectionSet is used to select object based on object identifiers
---|---|---|---|---|---3121: ObjectSelection	3121: ObjectSelection is an individual instance of an object set used for selection based on object identifiers
---|---|---|---|---|---|---3122: Level	3122: Level is the hierarchy level of an object being used for selection
---|---|---|---|---|---|---3123: Usage	3123: Usage indicate the usage type for object selection
---|---|---|---|---|---|---3130: ObjectSet	3130: ObjectSet is used as a parent for a group of objects used for selection
---|---|---|---|---|---|---|---3131: Object	3131: Object is an individual instance of an object used for selection
---|---|---|---|---|---|---|---|---3134: ObjectZTIC	3134: ObjectZTIC is the ZTIC of an individual object used for selection
---|---|---|---|---|---|---|---|---3135: ObjectCode	3135: ObjectCode is the Code of an individual object used for selection
---|---|---|---|---|---|---3140: ElementSet	3140: ElementSet is a set of elements used to select objects based on element values
---|---|---|---|---|---|---|---3141: Element	3141: Element is a an individual element used to select objects based on element values
---|---|---|---|---|---|---|---|---3142: ObjectElementZTIC	3142: ObjectElementZTIC is the Element ZTIC for an element used to select objects based on element values
---|---|---|---|---|---|---|---|---3143: ObjectElementCode	3143: ObjectElementCode is the Element Code of an element used to select objects based on element values
---|---|---|---|---|---|---|---|---3144: SelectionGroupNumber	3144: SelectionGroupNumber is an identifier used to combine selections by element values
---|---|---|---|---|---|---|---|---3145: SetOperator	3145: SetOperator contains a set of operators to combine selections based on element values
---|---|---|---|---|---|---|---|---3146: Operator	3146: Operator used to compare element values for selection
---|---|---|---|---|---|---|---|---3147: Qualifier	3147: Qualifier used to specify how selection based on element values is performed
---|---|---|---|---|---|---|---|---3148: Value	3148: Value is the element value used for selection
---|---|---|---|---|---|---3150: TypeSet	3150: TypeSet contains values for selecting objects based on type values
---|---|---|---|---|---|---|---3151: Type	3151: Type contains an individual type value for selection objects based on type value
---|---|---|---|---|---|---|---|---3152: TypeDefinitionZTICode	3152: TypeDefinitionZTICode is the ZT Instance Code for the type definition used for selection based on type value
---|---|---|---|---|---|---|---|---3153: TypeDefinitionCode	3153: TypeDefinitoionCode is the Code for the type definition used for selection based on type value
---|---|---|---|---|---|---|---|---3154: TypeValueZTICode	3154: TypeValueZTICode is the DS Instance Code for the type value used for selection based on type value
---|---|---|---|---|---|---|---|---3155: TypeValueCode	3155: TypeValueCode is the Code for the type value used for selection based on type value
---|---|---|---|---|---|---|---|---3156: SelectionGroupNumber	3156: SelectionGroupNumber is an identifier used when combining selections based on type value
---|---|---|---|---|---|---|---|---3157: SetOperator	3157: SetOperator is used to determine how selections based on type value are combined
---|---|---|---|---|---|---3160: LinkSet	3160: LinkSet contains values for selection objects based on links
---|---|---|---|---|---|---|---3161: Link	3161: Link contains values for an individual link used to select objects based on links
---|---|---|---|---|---|---|---|---3162: LinkTypeZTIC	3162: LinkTypeZTIC is the ZT Instance code for the link type used for selection based on links
---|---|---|---|---|---|---|---|---3163: LinkTypeCode	3163: LinkTypeCode is the Code used for the link type used for selection base on links
---|---|---|---|---|---|---|---|---3164: LinkToKindZTIC	3164: LinkToKindZTIC is the ZT Instance Code to the object kind being linked to for selection based on links
---|---|---|---|---|---|---|---|---3165: LinkToKindCode	3165: LinkToKindCode is the code for the object kind being linked to for selection based on links
---|---|---|---|---|---|---|---|---3166: LinkToZTIC	3166: LinkToZTIC is the ZT Instance Code of the object being linked to for selection based on links
---|---|---|---|---|---|---|---|---3167: LinkToCode	3167: LinkToCode is the Code of the object being linked to for selection based on links
---|---|---|---|---|---|---|---|---3168: SelectionGroupNumber	3168: SelectionGroupNumber is an identifier used to combine selections based on links
---|---|---|---|---|---|---|---|---3169: SetOperator	3169: SetOperator is an operator used for combining selections based on links
---|---|---|---|---|---|---|---3170: AdditionalSelectionSet	3170: AdditionalSelectionSet contains values for selection based on additional criterion
---|---|---|---|---|---|---|---|---3171: AdditionalSelection	3171: AdditionalSelection is an individual instance of an object selecting based on additional criterion
---|---|---|---|---|---|---|---|---3172: UsageType	3172: UsageType indicates how selection will be done based on additional criterion
---|---|---|---|---|---|---|---|---3173: ObjectZTIC	3173: ObjectZTIC is the ZT Instance Code of an object used for selection based on additional criterion
---|---|---|---|---|---|---|---|---3174: ObjectCode	3174: ObjectCode is the code of an object used for selection based on additional criterion
---|---|---|---|---|---|---|---|---3175: SelectionGroupNumber	3175: SelectionGroupNumber is an identifier used for combining selections based on additional criterion
---|---|---|---|---|---|---|---|---3176: SetOperator	3176: SetOperator is an operator used for combining selections based on group number
---32: RequestMessageStatusSet	32: RequestMessageStatusSet contains message status data added by the client in the request message
---|---320: MessageStatus	320: MessageStatus is an individual instance of a message status
---|---|---3201: MessageStatusZTIC	3201: MessageStatusZTIC is the ZTIC for a message status update
---|---|---3202: MessageStatusCode	3202: MessageStatusCode is the Code for a message status update
---|---|---3203: MessageStatusTimestamp	3203: MessageStatusTimestamp has the time a message status was updated
---|---|---3204: MessageStatusText	3204: MessageStatusText has the text for a message status update
---33: ClientLogMessageSet	33: ClientLogMessageSet contains log message entries
---|---330: ClientLogMessage	330: ClientLogMessage is an indiviual occurance of a log message
---|---|---3301: LogMessageZTIC	3301: LogMessageZTIC is the ZTIC for a log message
---|---|---3302: LogMessageCode	3302: LogMessageCode is the Code for a log message
---|---|---3303: LogMessageTimestamp	3303: LogMessageTimestamp is the time a log entry was written
---|---|---3304: LogMessageRelatedIndex	3304: LogMessageRelatedIndex tells the message index that the log message relates to
---|---|---3305: LogMessageText	3305: LogMessageText has the text of a log message
4: Response	4: Response contains all of the messge elements of the response and is populated by the server
---40: MaintainResponseSet	40: MaintainResponseSet corresponds to the MaintainSet in the Request and contains data pertaining to newly created or changed objects
---|---400: MaintainResponse	400: MaintainResponse corresponds to Maintain in the Request

---|---|---4000: ID	4000: ID corresponds to ID in in the Maintain in the Request and hold the same value as was sent in the Request
---|---|---4030: ObjectSet	4030: ObjectSet contains data related to objects that have been maintained
---|---|---|---4031: Object	4031: Object contains data for an individual object that has been maintained
---|---|---|---|---4032: ObjectKindZTICode	4032: ObjectKindZTICode is the Kind DSI Code for an object that was maintained
---|---|---|---|---4033: ObjectKindCode	4033: ObjectKindCode is the Kind Code for an object that was maintained
---|---|---|---|---4034: ObjectDSICode	4034: ObjectDSICode is the code of an object that was maintained
---|---|---|---|---4035: ObjectCodeTemp	4035: ObjectCodeTemp is the temporary code assigned by the client before the actual code was assigned by the server for a created object
---|---|---|---|---4036: ObjectCodeAssigned	4036: ObjectCodeAssigned is the code assigned by the server for an object that was created
---|---|---|---|---4037: ID	4037: ID is the identifier in the message for an object maintained, used to show hierarchy
---|---|---|---|---4038: ParentID	4038: ParentID is the identifier in the message for the parent of the object maintained, used to show hierarchy
---41: QueryResponseSet	41: QueryResponseSet contains the response by the server to a query sent by the client
---|---410: QueryResponse	410: QueryResponse contains the response to a query set from the client
---|---|---4100: ID	4100: ID identifies a query response and corresponds/holds the same value sent in 3100
---|---|---4180: StatisticalValueObjectSet	4180: StatisticalValueObjectSet contains all statistical values for a query response
---|---|---|---4181: StatisticalValuesObject	4181: StatisticalValuesObject contains object with a set of statistical values for a query response
---|---|---|---|---41810: StatisticalValuesObjectZTIC	41810: StatisticalValuesObjectZTIC is the ZT instance code for a statistical values object
---|---|---|---|---4188: GroupSet	4188: GroupSet contains a set of groups to define grouping for statistical values
---|---|---|---|---|---|---41881: GroupID	41881: GroupID is the ID used identify grouping values for statistical values
---|---|---|---|---|---|---41882: GroupLabel	41882: GroupLabel is the label for a grouping code
---|---|---|---|---|---|---41883: DimensionZTIC	41883: DimensionZTIC is the ZT instance code group
---|---|---|---|---|---|---41884: DimensionCode	41884: DimensionCode is the code for a Dimension for a group
---|---|---|---|---|---|---41885: IndexInTypeDefinitionPath	41885: IndexInTypeDefinitionPath is the index of a Type Definition in a Type Definition Path
---|---|---|---|---|---|---41886: TypeValueZTIC	41886: TypeValueZTIC is the ZT instance code for the Type Value in a group
---|---|---|---|---|---|---41887: TypeValueCode	41887: TypeValueCode is the code for a Type Value in a group
---|---|---|---|---41811: StatisticalValuesObjectCode	41811: StatisticalValuesObjectCode is the code for a statistical values object
---|---|---|---|---41812: StatisticTypeZTIC	41812: StatisticTypeZTIC is the ZT instance code for the statistic type
---|---|---|---|---41813: StatisticTypeCode	41813: StatisticTypeCode is the code for the statistic type
---|---|---|---|---4185: StatisticalValueSet	4185: StatisticalValueSet contains a set of statistical values
---|---|---|---|---|---41850: StatisticalValue	41850: StatisticalValue constains a statistical value
---|---|---|---|---|---|---41851: StatisticalValueID	41851: StatisticalValueID is the ID for a statistical value
---|---|---|---|---|---|---41852: StatisticalValue	41852: StatisticalValue is the value for a statistical value
---|---|---|---|---|---|---4186: StatisticalValueTypeValueSet	4186: StatisticalValueTypeValueSet contains the type values for a statistical value
---|---|---|---|---|---|---|---41860: StatisticalValueTypeValue	41860: StatisticalValueTypeValue contains the Type Value for a Statistical Value
---|---|---|---|---|---|---|---|---41861: TypeDefinitionZTIC	41861: TypeDefinitionZTIC is the ZT instance code for a Type Definition for a statistical value
---|---|---|---|---|---|---|---|---41862: TypeDefinitionCode	41862: TypeDefinitionCode is the code for a Type Definition for a statistical value
---|---|---|---|---|---|---|---|---41863: TypeValueZTIC	41863: TypeValueZTIC is the ZT instance code for a Type Value for a statistical value
---|---|---|---|---|---|---|---|---41864: TypeValueCode	41864: TypeValueCode is the code for a Type Value for a statistical value
---|---|---|---|---|---|---|---|---41865: SortKey	41865: SortKey is the sort key for the Type Value for a statistical value
---|---|---|---|---|---|---|---|---41866: LabelForTypeValue	41866: LabelForSummarizationForTypeValue is the label for a Type Value for a statistical value
---|---|---|---|---|---|---|---|---41867: GroupID1	41867: GroupID1 is the ID for the primary grouping for the statistical value
---|---|---|---|---|---|---|---|---41868: GroupID2	41868: GroupID2 is the ID of the secondary grouping for the statistical value
---|---|---|---|---|---|---|---|---41869: GroupID3	41869: GroupID3 is the ID for the tertiary grouping of the statistical value
---|---|---|---|---41814: SourceSetZTIC	41814: SourceSetZTIC is the ZT instance code for the set of objects to be summarized
---|---|---|---|---41815: SourceSetCode	41815: SourceSetCode is the code of set of objects to be summarized
---|---|---|---|---41816: UomZTIC	41816: UomZTIC is the ZT instance code of the unit of measure being used
---|---|---|---|---41817: UomCode	41817: UomCode is the code of the unit of measure being used
---|---|---|---|---41820: SourceObjectElementZTIC	41820: SourceObjectElementZTIC is the ZT instance code of the object element the summarized data comes from
---|---|---|---|---41821: SourceObjectElementCode	41821: SourceObjectElementCode is the code of the object element the summarized data comes from
---|---|---|---|---41822: StatisticalValuesObjectLabel	41822: StatisticalValuesObjectLabel is the label of the statistical values object
---|---|---|---|---4183: DimensionSet	4183: DimensionSet contains a set of dimensions
---|---|---|---|---|---41830: Dimension	41830: Dimension contains the data for one dimension
---|---|---|---|---|---|---41831: DimensionZTIC	41831: DimensionZTIC is the ZT instance code for the dimension
---|---|---|---|---|---|---41832: DimensionCode	41832: DimensionCode is the code for the dimension
---|---|---|---|---|---|---41833: DimensionLabel	41833: DimensionLabel is the label for the dimension
---|---|---|---|---|---|---41834: IndexInTypeDefinitionPathForDataSummarization	41834: IndexInTypeDefinitionPathForDataSummarization is the index position in the type definition path used for summarization
---|---|---|---|---|---|---41835: IndexInTypeDefinitionPathForGrouping	41835: IndexInTypeDefinitionPathForGrouping is the index position in the type definition path for grouping
---|---|---|---|---|---|---41836: IndexInTypeDefinitionPathForGrouping2	41836: IndexInTypeDefinitionPathForGrouping2 is the index position in the type definition path for second grouping
---|---|---|---|---|---|---4184: TypeDefinitionPath	4184: TypeDefinitionPath contains a set of Type Definitions used for the Statistical Values Object
---|---|---|---|---|---|---|---41840: PathZTIC	41840: PathZTIC is the ZT instance code for a Type Definition Path
---|---|---|---|---|---|---|---41841: PathCode	41841: PathCode is the Code for a Type Definition Path
---|---|---|---|---|---|---|---41842: TypeDefinitionSet	41842: TypeDefinitionSet contains a set of Type Definitions
---|---|---|---|---|---|---|---|---41843: TypeDefinition	41843: TypeDefinition contains values for a Type Definition
---|---|---|---|---|---|---|---|---41844: TypeDefinitionZTIC	41844: TypeDefinitionZTIC is the ZT instance code for the Type Definition

---|---|---|---|---|---|---|---|---|---|---41845: TypeDefinitionCode	41845: TypeDefinitionCode is the Code for the Type Definition
---|---|---|---|---|---|---|---|---|---|---41846: IndexInPath	41846: IndexInPath is the index of the Type Definition in the Type Definition Path
---|---|---4101: ObjectTemplateZTIcode	4101: ObjectTemplateZTIcode has the ZTIC of the template used for the query
---|---|---4102: ObjectTemplateCode	4102: ObjectTemplateCode is the Code of the template used for the query
---|---|---4103: DefaultEffectiveTimeStamp	4103: DefaultEffictiveTimeStamp is the timestamp that is used for selections from this query set if not specified at a more detailed level
---|---|---41051: ResponseTypeZTIC	41051: ResponseTypeZTIC is the ZT Instance Code of the response type for the parent Query Response
---|---|---41052: ResponseTypeCode	41052: ResponseTypeCode is the Code of the response type for the parent Query Response
---|---|---41053: RelatedObjectKindZTIC	41053: RelatedObjectKindZTIC is the ZT Instance Code of the related object kind
---|---|---41054: RelatedObjectKindCode	41054: RelatedObjectKindCode is the Code of the related object kind
---|---|---41055: RelatedObjectZTIC	41055: RelatedObjectZTIC is the ZT Instance Code of the related object
---|---|---41056: RelatedObjectCode	41056: RelatedObjectCode is the Code of the related object for the response type
---|---|---4130: ObjectSet	4130: ObjectSet contains the objects from a query response
---|---|---|---4131: Object	4131: Object is an individual instance of an object from a query response
---|---|---|---|---4132: ObjectKindZTICode	4132: ObjectKindZTICode is the Object Kind DSI Code for an object in a query response
---|---|---|---|---4133: ObjectKindCode	4133: ObjectKindCode is the Kind Code for an object in a query response
---|---|---|---|---4134: ObjectZTICode	4134: ObjectZTICode is the ZTIC for an object in a query response
---|---|---|---|---4135: ObjectCode	4135: ObjectCode is the Code for an object in a query response
---|---|---|---|---4137: ID	4137: ID is an indentifier assigned by the server for an object in a query response, used to show the hierarchy of objects
---|---|---|---4138: ParentId	4138: ParentId is the id of the parent object in the hierarchy of objects in a query response
---|---|---|---|---4139: LevelsDown	4139: LevelsDown indicates how far down in the hierarchy the object is in the set of objects in a query response
---|---|---|---|---41390: TemplateZTIC	41390: TemplateZTIC is the ZTIC of the template used by a object in a query response
---|---|---|---|---41391: TemplateCode	41391: TemplateCode is the Code of the template used by a object in a query response
---|---|---|---|---41392: ObjectStatus	41392: ObjectStatus is the status of the complete object i.e. active, inactive, flag for deletion
---|---|---|---|---4140: ElementSet	4140: ElementSet has the elements for an object in a query response
---|---|---|---|---|---4141: Element	4141: Element is an individual element for an object in a query response
---|---|---|---|---|---|---4142: ObjectElementZTICode	
---|---|---|---|---|---|---4143: ObjectElementCode	4143: ObjectElementCode is the Code for an object element for an object in a query response
---|---|---|---|---|---|---4144: ObjectElementValue	4144: ObjectElementValue is the value for an object element for an object in a query response
---|---|---|---|---|---|---4146: ObjectElementStatus	4146: ObjectElementStatus is the status for the object element
---|---|---|---|---|---|---4147: EffectiveTimeStamp	4147: EffectiveTimeStamp is the time stamp for the ObjectElement
---|---|---|---|---4150: TypeSet	4150: TypeSet has the type values for an object in a query response
---|---|---|---|---|---4151: Type	4151: Type is an individual type value for an object in a query response
---|---|---|---|---|---|---4152: TypeDefinitionZTICode	4152: TypeDefinitionZTICode is the the ZTIC for the type defintion for a type value for an object in a query response
---|---|---|---|---|---|---4153: TypeDefinitionCode	4153: TypeDefinitionCode is the Code for the type definition for a type value for an object in a query response
---|---|---|---|---|---|---4154: TypeValueZTICode	4154: TypeValueZTICode is the ZTIC for the type value for an object in a query response
---|---|---|---|---|---|---4155: TypeValueCode	4155: TypeValueCode is the Code for the type value for an object in a query response
---|---|---|---|---|---|---4156: TypeValueStatus	4156: TypeValueStatus is the status code for the TypeValueCode
---|---|---|---|---|---|---4157: EffectiveTimeStamp	4157: EffectiveTimeStamp is the time stamp for the Type Value
---|---|---|---|---4160: LinkSet	4160: LinkSet has the links for an obect in a query response
---|---|---|---|---|---4161: Link	4161: Link is an individual link for an object in a query response
---|---|---|---|---|---|---4162: LinkTypeZTIC	4162: LinkTypeZTIC is the ZTIC for the link type for a link for an object in a query response
---|---|---|---|---|---|---4163: LinkTypeCode	4163: LinkTypeCode is the Code for the link type for a link for an object in a query response
---|---|---|---|---|---|---4164: LinkToKindZTIC	4164: LinkToKindZTIC is the ZTIC for the link-to kind for a link for an object in a query response
---|---|---|---|---|---|---4165: LinkToKindCode	4165: LinkToKindCode is the Code for the link-to kind for a link for an object in a query response
---|---|---|---|---|---|---4166: LinkToZTIC	4166: LinkToZTIC is the ZTIC for the link-to for a link for an object in a query response
---|---|---|---|---|---|---4167: LinkToCode	4167: LinkToCode is the Code for the link-to for a link for an object in a query response
---|---|---|---|---|---|---41691: LinkStatus	41691: LinkStatus is the status for the link
---|---|---|---|---|---|---41692: EffectiveTimeStamp	41692: EffectiveTimeStamp is the effective time stamp for the link
---|---|---|---|---|---|---41693: LinkValue	41693: LinkValue indicates the strength or priority of the link
---42: ResponseMessageStatusSet	42: ResponseMessageStatusSet contains the Response Message Status values
---|---420: MessageStatus	420: MessageStatus is an indidual instance of a MessageStatus
---|---|---4201: MessageStatusZTIC	4201: MessageStatusZTIC is the DS Instance a ResponseMessage Status
---|---|---4202: MessageStatusCode	4202: MessageStatusCode is the Code for a Response Message Status
---|---|---4203: MessageStatusTimestamp	4203: MessageStatusTimestamp is the timestamp for a Message Status
---|---|---4204: MessageStatusText	4204: MessageStatusText is the text for a message status
---43: ServerLogSystemMessageSet	43: ServerLogSystemMessageSet contains server log messages
---|---430: ServerLogSystemMessage	430: ServerLogSystemMessage is an individual instance of a server log message
---|---|---4300: ID	4300: ID is the identifier for a server log message
---|---|---4301: SystemMessageDefZTIC	4301: SystemMessageDefZTIC is the DS Instance for a system message definition
---|---|---4302: SystemMessageDefCode	4302: SystemMessageDefCode is the Code the a system message definition
---|---|---4303: MessageCategory	4303: MessageCategory is the category for a system message
---|---|---4304: Timestamp	4304: Timestamp is the timestamp for a system message
---|---|---4311: RelatedSection	4311: RelatedSection has the section of the message related to the system message
---|---|---4312: RelatedSectionID	4312: RelatedSectionID has the section ID of the message related to the system message
---|---|---4313: SerialNumberOfRelatedMessageElement	4313: SerialNumberOfRelatedMessageElement contains the serial number of the message element related to

	the system message
---|---|---4321: ShortSystemMessage	4321: ShortSystemMessage is the short text version of a system message
---|---|---4322: LongSystemMessage	4322: LongSystemMessage is the long text version of a system message
---|---|---4330: ParameterSet	4330: ParameterSet contains the parameters for a system message
---|---|---|---4331: Parameter	4331: Parameter contains an indiidual parameter for a system message
---|---|---|---|---4332: ParameterZTIC	4332: ParameterZTIC is the ZT Instance Code for a parameter for a system message
---|---|---|---|---4333: ParameterCode	4333: ParameterCode is the code for a parameter for a system message
---|---|---|---|---4334: ParameterValue	4334: ParameterValue is the value for a parameter for a system message

Appendix 7--Base Object Element Values

Listed below are values for the Base Object Elements Values. The column headings correspond to the Table Elements found in Appendix 5.

1-OKIC	2-OKC	3-OIC	4-OC	7-OEIC	8-OEC	31-OE-Value
2	1	2	1	2	1	Object Kind (Object Kind)
2	1	2	2	2	1	Object Template (Object Kind)
2	1	2	3	2	1	Object Element (Object Kind)
2	1	2	4	2	1	Table Element (Object Kind)
2	1	2	5	2	1	Type Value Based Template Rule (Object Kind)
2	1	2	6	2	1	Link Based Template Rule (Object Kind)
2	1	2	7	2	1	Object Element Value Based Template Rule (Object Kind)
2	1	2	8	2	1	Object Kind Type (Object Kind)
2	1	2	9	2	1	Object Type Definition (Object Kind)
2	1	2	10	2	1	Link Type (Object Kind)
2	1	2	11	2	1	Link Target Type (Object Kind)
2	1	2	12	2	1	Technical Profile (Object Kind)
2	1	2	13	2	1	Object Set Definition (Object Kind)
2	1	2	14	2	1	ZT Instance (Object Kind)
2	1	2	15	2	1	Namespace (Object Kind)
2	1	2	16	2	1	Object Element Usage Type (Object Kind)
2	1	2	17	2	1	Object Template Type (Object Kind)
2	1	2	18	2	1	Language (Object Kind)
2	1	2	19	2	1	User (Object Kind)
2	1	2	20	2	1	User Type (Object Kind)
2	1	2	21	2	1	Code Range (Object Kind)
2	1	2	22	2	1	Code Scheme (Object Kind)
2	1	2	23	2	1	Code Scheme Type (Object Kind)
2	1	2	24	2	1	System Message Definition (Object Kind)
2	1	2	25	2	1	System Message Definition Type (Object Kind)
2	1	2	26	2	1	Parameter (Object Kind)
2	1	2	27	2	1	Parameter Type (Object Kind)
2	1	2	28	2	1	System Message Definition Category (Object Kind)
2	1	2	29	2	1	System Message Parameter Category (Object Kind)
2	1	2	30	2	1	Table (Object Kind)
2	1	2	31	2	1	Data Element (Object Kind)
2	1	2	32	2	1	Data Type (Object Kind)
2	1	2	33	2	1	Extended Key Definition (Object Kind)
2	1	2	34	2	1	Extended Key Definition Type (Object Kind)
2	1	2	35	2	1	Object Element Usage Type (Object Kind)
2	1	2	36	2	1	Function (Object Kind)
2	1	2	37	2	1	Class/Program (Object Kind)
2	1	2	38	2	1	Class/Program Type (Object Kind)
2	1	2	39	2	1	Resource (Object Kind)
2	1	2	40	2	1	Resource Protcol (Object Kind)
2	1	2	41	2	1	Resource Name Extension (Object Kind)
2	1	2	42	2	1	Software Runtime Profile (Object Kind)
2	1	2	43	2	1	Software Runtime Component (Object Kind)
2	1	2	44	2	1	Patch Level (Object Kind)
2	1	2	45	2	1	Language & Patch Level Composite Extended Key (Object Kind)
2	1	2	46	2	1	Object Set Definition Type (Object Kind)
2	1	2	47	2	1	Link Type Usage Type (Object Kind) i.e. standard, object sequencing
2	1	2	48	2	1	Type Value Based Object Set (Object Kind)
2	1	2	49	2	1	Template Set (Object Kind)
2	1	2	50	2	1	Generic Object (Object Kind)
2	1	2	51	2	1	Generic Object Element Value (Object Kind)
2	1	2	52	2	1	Function Group (Object Kind)
2	1	2	53	2	1	Function Group Type (Object Kind)
2	1	2	54	2	1	Parameter Value Group (Object Kind)
2	1	2	55	2	1	Object Template Path (Object Kind)

2	1	2	56	2	1	Type Definition Path (Object Kind)
2	1	2	57	2	1	Parameter Value (Object Kind)
2	1	2	58	2	1	Statistical Values Object (Object Kind)
2	1	2	59	2	1	Statistical Values Dimension (Object Kind)
2	1	2	60	2	1	Statistic Type (Object Kind) sum, count, mean, mode, median, std dev
2	1	2	61	2	1	Unit of Measure (Object Kind)
2	1	2	62	2	1	Unit of Measure Dimension (Object Kind) length, weight, time, currency
2	1	2	63	2	1	Type Definition Usage Type (Object Kind)
2	1	2	64	2	1	System Message Element Definition (Object Kind)
2	1	2	65	2	1	System Message Element Definition Type (Object Kind)
2	1	2	66	2	1	Numeric Data Type Details (Object Kind)
2	2	2	1	2	2	Object Kind (Base Template)
2	2	2	2	2	2	Object Template (Base Template)
2	2	2	3	2	2	Object Element (Base Template)
2	2	2	4	2	2	Table Element (Base Template)
2	2	2	5	2	2	Type Value Based Template Rule (Base Template)
2	2	2	6	2	2	Link Based Template Rule (Base Template)
2	2	2	7	2	2	Object Element Value Based Template Rule (Base Template)
2	2	2	8	2	2	Object Kind Type (Base Template)
2	2	2	9	2	2	Object Type Definition (Base Template)
2	2	2	10	2	2	Link Type (Base Template)
2	2	2	11	2	2	Link Target Type (Base Template)
2	2	2	12	2	2	Technical Profile (Base Template)
2	2	2	13	2	2	Object Set Definition (Base Template)
2	2	2	14	2	2	ZT Instance (Base Template)
2	2	2	15	2	2	Namespace (Base Template)
2	2	2	16	2	2	Object Element Usage Type (Base Template)
2	2	2	17	2	2	Object Template Type (Base Template)
2	2	2	18	2	2	Language (Base Template)
2	2	2	19	2	2	User (Base Template)
2	2	2	20	2	2	User Type (Base Template)
2	2	2	21	2	2	Code Range (Base Template)
2	2	2	22	2	2	Code Scheme (Base Template)
2	2	2	23	2	2	Code Scheme Type (Base Template)
2	2	2	24	2	2	System Message Definition (Base Template)
2	2	2	25	2	2	System Message Definition Type (Base Template)
2	2	2	26	2	2	Parameter (Base Template)
2	2	2	27	2	2	Parameter Type (Base Template)
2	2	2	28	2	2	System Message Definition Category (Base Template)
2	2	2	29	2	2	System Message Parameter Category (Base Template)
2	2	2	30	2	2	Table (Base Template)
2	2	2	31	2	2	Data Element (Base Template)
2	2	2	32	2	2	Data Type (Base Template)
2	2	2	33	2	2	Extended Key Definition (Base Template)
2	2	2	34	2	2	Extended Key Definition Category (Base Template)
2	2	2	35	2	2	Object Element Usage Type (Base Template)
2	2	2	36	2	2	Function (Base Template)
2	2	2	37	2	2	Class/Program (Base Template)
2	2	2	38	2	2	Class/Program Type (Base Template)
2	2	2	39	2	2	Resource (Base Template)
2	2	2	40	2	2	Resource Protcol (Base Template)
2	2	2	41	2	2	Resource Name Extension (Base Template)
2	2	2	42	2	2	Software Runtime Profile (Base Template)
2	2	2	43	2	2	Software Runtime Componenst (Base Template)
2	2	2	44	2	2	Patch Level (Base Template)
2	2	2	45	2	2	Language & Patch Level Composite Extended Key (Base Template)
2	2	2	46	2	2	Object Set Defintion Type (Base Template)
2	2	2	47	2	2	Link Type Usage Type (Base Template)
2	2	2	48	2	2	Type Value Based Object Set (Base Template)
2	2	2	49	2	2	Template Set (Base Template)
2	2	2	50	2	2	Generic Object (Base Template)
2	2	2	51	2	2	Generic Object Element Value (Base Template)
2	2	2	52	2	2	Function Group (Base Template)
2	2	2	53	2	2	Function Group Type (Base Template)
2	2	2	54	2	2	Parameter Value Group (Base Template)
2	2	2	55	2	2	Object Template Path (Base Template)
2	2	2	56	2	2	Type Definition Path (Base Template)
2	2	2	57	2	2	Parameter Value (Base Template)
2	2	2	58	2	2	Statistical Values Object (Base Template)
2	2	2	59	2	2	Statistical Values Dimension (Base Template)

2	2	2	60	2	2	Statistic Type (Base Template) sum, count, mean, mode, median, std dev
2	2	2	61	2	2	Unit of Measure (Base Template)
2	2	2	62	2	2	Unit of Measure Dimension (Base Template) length, weight, time, currency
2	2	2	63	2	2	Type Definition Usage Type (Base Template)
2	2	2	64	2	2	System Message Element Definition (Base Template)
2	2	2	65	2	2	System Message Element Definition Type (Base Template)
2	2	2	66	2	2	Numeric Data Type Details (Base Template)
2	3	2	1	2	3	Object Kind Description
2	3	2	2	2	3	Object Template Description
2	3	2	3	2	3	Object Element Description
2	3	2	4	2	3	Table Element Description
2	3	2	5	2	3	Type Value Based Template Rule Description
2	3	2	6	2	3	Link Based Template Rule Description
2	3	2	7	2	3	Object Element Value Based Template Rule Description
2	3	2	8	2	3	Object Kind Type Description
2	3	2	9	2	3	Object Type Definition Description
2	3	2	10	2	3	Link Type Description
2	3	2	11	2	3	Link Target Type Description
2	3	2	12	2	3	Technical Profile Description
2	3	2	13	2	3	Object Set Definition Description
2	3	2	14	2	3	ZT Instance Description
2	3	2	15	2	3	Namespace Description
2	3	2	16	2	3	Object Element Usage Type Description
2	3	2	17	2	3	Object Template Type Description
2	3	2	18	2	3	Language Description
2	3	2	18	2	3	Application Action Object Code (Object Element)
2	3	2	19	2	3	User Description
2	3	2	20	2	3	User Type Description
2	3	2	21	2	3	Code Range Description
2	3	2	22	2	3	Code Scheme Description
2	3	2	23	2	3	Code Scheme Type Description
2	3	2	24	2	3	System Message Definition Description
2	3	2	25	2	3	System Message Definition Type Description
2	3	2	26	2	3	Parameter Description
2	3	2	27	2	3	Parameter Type Description
2	3	2	28	2	3	System Message Definition Category Description
2	3	2	29	2	3	System Message Parameter Category Description
2	3	2	30	2	3	Table Description
2	3	2	31	2	3	Data Element Description
2	3	2	32	2	3	Data Type Description
2	3	2	33	2	3	Extended Key Definition Description
2	3	2	34	2	3	Extended Key Definition Type Description
2	3	2	35	2	3	Object Element Usage Type Description
2	3	2	36	2	3	Funtion Description
2	3	2	37	2	3	Class/Program Description
2	3	2	38	2	3	Class/Program Type Description
2	3	2	39	2	3	Resource Description
2	3	2	40	2	3	Resource Protocol Description
2	3	2	41	2	3	Resource Name Extension Description
2	3	2	42	2	3	Software Runtime Profile Description
2	3	2	43	2	3	Software Runtime Component Description
2	3	2	44	2	3	Patch Level Description
2	3	2	45	2	3	Language & Patch Level Composite Extended Key Description
2	3	2	46	2	3	Object Set Definition Type Description
2	3	2	47	2	3	Link Type Usage Type Description
2	3	2	48	2	3	Type Value Based Object Set Description
2	3	2	49	2	3	Template Set Description
2	3	2	50	2	3	Generic Object Description
2	3	2	51	2	3	Generic Object Element Value Description
2	3	2	52	2	3	Function Group Description
2	3	2	53	2	3	Function Group Type Description
2	3	2	54	2	3	Parameter Value Group Description
2	3	2	55	2	3	Object Template Path Description
2	3	2	56	2	3	Type Definition Path Description
2	3	2	57	2	3	Parameter Value Description
2	3	2	58	2	3	Statistical Values Object Description
2	3	2	59	2	3	Statistical Values Dimension Description
2	3	2	60	2	3	Statistic Type Description sum, count, mean, mode, median, std dev
2	3	2	61	2	3	Unit of Measure Description
2	3	2	62	2	3	Unit of Measure Dimension Description length, weight, time, currency

2	3	2	63	2	3	Type Definition Usage Type Description
2	3	2	64	2	3	System Message Element Definition Description
2	3	2	65	2	3	System Message Element Definition Type Description
2	3	2	66	2	3	Numeric Data Type Details Description
2	3	2	200	2	3	System Message Short Text
2	3	2	201	2	3	System Message Long Text
2	3	2	202	2	3	Minimum value for Code Range
2	3	2	203	2	3	Maximum value for Code Range
2	3	2	204	2	3	Next Code for Code Range
2	3	2	205	2	3	Code string for Code Scheme
2	3	2	206	2	3	Numeric Preferred for Code Range
2	3	2	207	2	3	Code Prefix
2	3	2	208	2	3	Leading Character for Code Range
2	3	2	209	2	3	Leading Character for Numeric Codes
2	3	2	210	2	3	Resource Path/Name
2	3	2	211	2	3	Resource Path/Name--Patch Level Dependent
2	3	2	212	2	3	Resource Path/Name--Language Dependent
2	3	2	213	2	3	Resource Path/Name--Patch Level and Language Dependent
2	3	2	214	2	3	Data Element--Maximum Length
2	3	2	215	2	3	Namespace
2	3	2	216	2	3	Dimension Label
2	3	2	217	2	3	Index for Type Definition for Summarization in Type Definition Path used by Dimension
2	3	2	218	2	3	Index for Type Definition for Grouping in Type Defiition Path used for Dimension
2	3	2	219	2	3	Index for Type Definition for Groupin2 in Type Definition Path used for Dimension
2	3	2	220	2	3	Namespace of Object Kind of Object Represented by a Generic Object
2	3	2	221	2	3	Code of Object Kind of Object Represented by a Generic Object
2	3	2	222	2	3	Namespace of Object Represented by a Generic Object
2	3	2	223	2	3	Code of Object Represented by a Generic Object
2	3	2	224	2	3	Start Index in a Type Definition Path for a Type Value Based Object Set
2	3	2	225	2	3	Stop Index in a Type Definition Path for a Type Value Based Object Set
2	3	2	226	2	3	System Message--Short Boilerplate Text
2	3	2	227	2	3	System Message--Long Boilerplate Text
2	4	2	1	2	4	Object Kind ZTIC
2	4	2	2	2	4	Object Kind Code
2	4	2	3	2	4	Object Code ZTIC
2	4	2	4	2	4	Object Code
2	4	2	5	2	4	Record Type ZTIC
2	4	2	6	2	4	Record Type Code
2	4	2	7	2	4	Object Element ZTIC
2	4	2	8	2	4	Object Element Code
2	4	2	9	2	4	Extended Key Definition ZTIC
2	4	2	10	2	4	Extended Key Definition Code
2	4	2	11	2	4	Extended Key Value ZTIC
2	4	2	12	2	4	Extended Key Value Code
2	4	2	13	2	4	Type Definition ZTIC
2	4	2	14	2	4	Type Definition Code
2	4	2	15	2	4	Type Value ZTIC
2	4	2	16	2	4	Type Value Code
2	4	2	17	2	4	Link Type ZTIC
2	4	2	18	2	4	Link Type Code
2	4	2	19	2	4	Link To Kind ZTIC
2	4	2	20	2	4	Link To Kind Code
2	4	2	21	2	4	Link To Code ZTIC
2	4	2	22	2	4	Link To Code
2	4	2	23	2	4	Object Template ZTIC
2	4	2	24	2	4	Object Template Code
2	4	2	25	2	4	Status
2	4	2	26	2	4	Timestamp--Effective
2	4	2	27	2	4	Sequence Number
2	4	2	28	2	4	Message ID
2	4	2	29	2	4	Message Segment Index
2	4	2	30	2	4	Link Value
2	4	2	31	2	4	Object Element Value
2	4	2	32	2	4	Server DB--ZT Instance ID
2	9	2	1	2	9	Type Definition: Object Element Usage Type for a Object Element
2	9	2	2	2	9	Type Definition: Link Target Type for Link Type
2	9	2	3	2	9	Type Definition: Object Kind for Object Template
2	9	2	4	2	9	Type Definition: Object Kind for Type Definition
2	9	2	5	2	9	Type Definition: Code Range for Object Kind
2	9	2	6	2	9	Type Definition: Code Scheme for Code Range

2	9	2	7	2	9	Type Definition: Set of Functions for Software Runtime Component
2	9	2	8	2	9	Type Definition: Patch Level for Software Runtime Component
2	9	2	9	2	9	Type Definition: Data Element for Object Element
2	9	2	10	2	9	Type Definition: Data Type for Data Element
2	9	2	11	2	9	Type Definition: Extended Key Definition for Data Element
2	9	2	12	2	9	Type Definition: Extended Key Definition Type for Extended Key Definition
2	9	2	13	2	9	Type Definition: Type Definition for Extended Key Definition
2	9	2	14	2	9	Type Definition: Object Set for a Type Definition for allowed Type Values
2	9	2	15	2	9	Type Definition: Increment Function for Code Scheme
2	9	2	16	2	9	Type Definition: Class/Program for Function
2	9	2	17	2	9	Type Definition: Patch Level for a Runtime Profile Component
2	9	2	18	2	9	Type Definition: Set of Classses/Programs for a Runtime Profile Component
2	9	2	19	2	9	Type Definition: Set Type for a Set
2	9	2	20	2	9	Type Definition: Object Template for Extended Key Definition using Composite Key
2	9	2	21	2	9	Type Definition: Object Kind Type for Object Kind (standard, composite key etc)
2	9	2	22	2	9	Type Definition: Default Object Template for a Object Kind--Minimal Data (to return values)
2	9	2	23	2	9	Type Definition: Default Object Template for a Object Kind--Full Data (to return values)
2	9	2	24	2	9	Type Definition: Link Type for a Type Definition to create a bi-directional link (1 server generated + 1 type def)
2	9	2	25	2	9	Type Definition: Object Template for a Type Definition (to select values of a type value)
2	9	2	26	2	9	Type Definition: Link Type Usage Type for a Link Type
2	9	2	27	2	9	Type Definition: Function Group Type for a Function Group
2	9	2	28	2	9	Type Definition: Template Type for a Template
2	9	2	29	2	9	Type Definition: Type Definition Usage Type for a Type Definition
2	9	2	30	2	9	Type Definition: Reference Template for a Template
2	9	2	31	2	9	Type Definition: Parameter for a Parameter Value
2	9	2	32	2	9	Type Definition: Statistic Type for a Statistical Values Object (sum, count, mode, mean, std deviation)
2	9	2	33	2	9	Type Definition: Object Set for a Statistical Values Object
2	9	2	34	2	9	Type Definition: Type Definition Path for a Statistical Values Dimension
2	9	2	35	2	9	Type Definition: Unit of Measure for a Statistical Values Object
2	9	2	36	2	9	Type Definition: Unit of Measure Dimension for a Unit of Measure
2	9	2	37	2	9	Type Definition: Object Element for a Statistical Values Object for source of data
2	9	2	38	2	9	Type Definition: Object Template for a Type Based Template Rule
2	9	2	39	2	9	Type Definition: Object Set for a Type Value Based Template Rule
2	9	2	40	2	9	Type Definition: Object Set for Input Objects for a Type Values Based Object Set
2	9	2	41	2	9	Type Definition: Object Set for Type Values for a Type Value Based Object Set
2	9	2	42	2	9	Type Definition: Type Defininition Path for a Type Value Based Object Set
2	9	2	43	2	9	Type Definition: Object Template for a Link Based Template Rule
2	9	2	44	2	9	Type Definition: Object Set for a Link Based Template Rule
2	9	2	45	2	9	Type Definition: Object Template for an Object Element Value Based Template Rule
2	9	2	46	2	9	Type Definition: Object Set for an Object Element Value Based Template Rule
2	9	2	47	2	9	Type Definition: Object Element for an Object Element Value Based Template Rule
2	9	2	48	2	9	Type Definition: Object Set for a Data Element for validation of character values
2	9	2	49	2	9	Type Definition: Object Element for a Data Element for validation of character values
2	10	2	1	2	10	Link Type: Object Template to Technical Profile
2	10	2	2	2	10	Link Type: Technical Profile to Technical Profile
2	10	2	3	2	10	Link Type: Technical Profile to Object Elements
2	10	2	4	2	10	Link Type: Technical Profile to Type Definitions
2	10	2	5	2	10	Link Type: Technical Profile to LinkTypes
2	10	2	6	2	10	Link Type: Link Target Type to Object Template
2	10	2	7	2	10	Link Type: System Message Definition to Parameter
2	10	2	8	2	10	Link Type: Function to Parameter
2	10	2	9	2	10	Link Type: Class to Resource
2	10	2	10	2	10	Link Type: Server Software Runtime Profile to Server Software Runtime Component
2	10	2	11	2	10	Link Type: Server Software Runtime Component to Server Software Runtime Component
2	10	2	12	2	10	Link Type: Set Definition to an Object Kind to include all objects of that Kind as members
2	10	2	13	2	10	Link Type: Set Definition to individual objects to include as members
2	10	2	14	2	10	Link Type: Set Definition to Type Value Based Object Set, to include all objects with corresponding type values as members
2	10	2	15	2	10	Link Type: Set Definition to Template Set to include all objects using a Template in the Template Set as members
2	10	2	16	2	10	Link Type: Set Definition to Set Definition of Type, Link Target to include all objects linked to the targets as members
2	10	2	17	2	10	Link Type: Set Definition to Set Def. of Type Object Element Value to include objects with the object element value as members
2	10	2	18	2	10	Link Type: Set Definition to Set Definition to include target set members as members of source set (OR Condition)
2	10	2	19	2	10	Link Type: Set Definition to Set Def. to include target set members as members if also included in source set (AND Condition)
2	10	2	20	2	10	Link Type: Set Definition to Set Def. to remove target set members as members of the source set (NOT Condition)
2	10	2	21	2	10	Link Type: Set Definition to Custom Function to determine which objects will be members
2	10	2	22	2	10	Link Type: Technical Profile to Data Elements
2	10	2	23	2	10	Link Type: Set Definition to Namespace where all objects that have an Object Kind with the ZTNS linked to are included as members
2	10	2	24	2	10	Link Type: Set Definition to Namespace where all objects that have an Object ZTNS linked to are included as members

2	10	2	25	2	10	Link Type: Technical Profile to Technical Profile to remove the attributes of the target from the Tech Profile of the source
2	10	2	26	2	10	Link Type: Statistical Values Object to Statistical Values Dimension
2	10	2	27	2	10	Link Type: Type Definition Path to Type Definition
2	10	2	28	2	10	Link Type: Data Element to a Function to determine the value of Object Elements that use the Data Element
2	10	2	29	2	10	Link Type: Type Definition to a Type Value Based Template Rule
2	10	2	30	2	10	Link Type: Set Definition to Generic Objects to include the object referred to by the generic object as a member
2	10	2	31	2	10	Link Type: Function Group to a System Message Definition
2	10	2	32	2	10	Link Type: Function Group to a Function
2	10	2	33	2	10	Link Type: Link Type to Link Based Template Rule
2	10	2	34	2	10	Link Type: Object Element to Object Element Value Based Template Rule
2	11	2	1	2	11	Link Target Type: Object Template to Technical Profile
2	11	2	2	2	11	Link Target Type: Technical Profile to Technical Profile
2	11	2	3	2	11	Link Target Type: Technical Profile to Object Elements
2	11	2	4	2	11	Link Target Type: Technical Profile to Type Definitions
2	11	2	5	2	11	Link Target Type: Technical Profile to LinkTypes
2	11	2	6	2	11	Link Target Type: Link Target Type to Object Template
2	11	2	7	2	11	Link Target Type: System Message to Parameter
2	11	2	8	2	11	Link Target Type: Function to Parameter
2	11	2	9	2	11	Link Target Type: Function to Resource
2	11	2	10	2	11	Link Target Type: Server Software Runtime Profile to Server Software Runtime Component
2	11	2	11	2	11	Link Target Type: Server Software Runtime Component to Server Software Runtime Component
2	11	2	12	2	11	Link Target Type: Set Definition to an Object Kind to include all objects of that Kind as members
2	11	2	13	2	11	Link Target Type: Set Definition to individual objects to include as members
2	11	2	14	2	11	Link Target Type: Set Definition to Type Value Based Object Set, to include all objects with corresponding type values as members
2	11	2	15	2	11	Link Target Type: Set Definition to Template Set to include all objects using a Template in the Template Set as members
2	11	2	16	2	11	Link Target Type: Set Definition to Set Definition of Type, Link Target to include all objects linked to the targets as members
2	11	2	17	2	11	Link Target Type: Set Definition to Set Def. of Type Object Element Value to include objects with the object element value as members
2	11	2	18	2	11	Link Target Type: Set Definition to Set Definition to include target set members as members of source set (OR Condition)
2	11	2	19	2	11	Link Target Type: Set Definition to Set Def. to include target set members as members if also included in source set (AND Condition)
2	11	2	20	2	11	Link Target Type: Set Definition to Set Def. to remove target set members as members of the source set (NOT Condition)
2	11	2	21	2	11	Link Target Type: Set Definition to Custom Function to determine which objects will be members
2	11	2	22	2	11	Link Target Type: Technical Profile to Data Elements
2	11	2	23	2	11	Link Target Type: Set Definition to Namespace where all objects that have an Object Kind with the ZTNS linked to are included as members
2	11	2	24	2	11	Link Target Type: Set Definition to Namespace where all objects that have an Object ZTNS linked to are included as members
2	11	2	25	2	11	Link Target Type: Technical Profile to Technical Profile to remove the attributes of the target from the Tech Profile of the source
2	11	2	26	2	11	Link Target Type: Statistical Values Object to Statistical Values Dimension
2	11	2	27	2	11	Link Target Type: Type Definition Path to Type Definition
2	11	2	28	2	11	Link Target Type: Data Element to a Function to determine the value of Object Elements that use the Data Element
2	11	2	29	2	11	Link Target Type: Type Definition to a Type Value Based Template Rule
2	11	2	30	2	11	Link Target Type: Set Definition to Generic Objects to include the object referred to by the generic object as a member
2	11	2	31	2	11	Link Target Type: Function Group to a System Message Definition
2	11	2	32	2	11	Link Target Type: Function Group to a Function
2	11	2	33	2	11	Link Target Type: Link Type to Link Based Template Rule
2	11	2	34	2	11	Link Target Type: Object Element to Object Element Value Based Template Rule
2	12	2	1	2	12	Object Kind (Base Technical Profile)
2	12	2	2	2	12	Object Template (Base Technical Profile)
2	12	2	3	2	12	Object Element (Base Technical Profile)
2	12	2	4	2	12	Table Element (Base Technical Profile)
2	12	2	5	2	12	Type Value Based Template Rule (Base Technical Profile)
2	12	2	6	2	12	Link Based Template Rule (Base Technical Profile)
2	12	2	7	2	12	Object Element Value Based Template Rule (Base Tech Profile)
2	12	2	8	2	12	Object Kind Type (Base Technical Profile)
2	12	2	9	2	12	Object Type Definition (Base Technical Profile)
2	12	2	10	2	12	Link Type (Base Technical Profile)
2	12	2	11	2	12	Link Target Type (Base Technical Profile)
2	12	2	12	2	12	Technical Profile (Root Technical Profile)
2	12	2	13	2	12	Object Set Definition (Base Technical Profile)
2	12	2	14	2	12	ZT Instance (Base Technical Profile)
2	12	2	15	2	12	Namespace (Base Technical Profile)
2	12	2	16	2	12	Object Element Usage Type (Base Technical Profile)
2	12	2	17	2	12	Object Template Type (Base Technical Profile)
2	12	2	18	2	12	Language (Base Technical Profile)
2	12	2	19	2	12	User (Base Technical Profile)
2	12	2	20	2	12	User Type (Base Technical Profile)

2	12	2	21	2	12	Code Range (Base Technical Profile)
2	12	2	22	2	12	Code Scheme (Base Technical Profile)
2	12	2	23	2	12	Code Scheme Type (Base Technical Profile)
2	12	2	24	2	12	System Message Definition (Base Technical Profile)
2	12	2	25	2	12	System Message Definition Type (Base Technical Profile)
2	12	2	26	2	12	Parameter (Base Technical Profile)
2	12	2	27	2	12	Parameter Type (Base Technical Profile)
2	12	2	28	2	12	System Message Definition Category (Base Technical Profile)
2	12	2	29	2	12	System Message Definition Parameter Category (Base Technical Profile)
2	12	2	30	2	12	Table (Base Technical Profile)
2	12	2	31	2	12	Data Element (Base Technical Profile)
2	12	2	32	2	12	Data Type (Base Technical Profile)
2	12	2	33	2	12	Extended Key Definition (Base Technical Profile)
2	12	2	34	2	12	Extended Key Definition Type (Base Tech Profile)
2	12	2	35	2	12	Object Element Usage Type (Base Technical Profile)
2	12	2	36	2	12	Function (Base Technical Profile)
2	12	2	37	2	12	Class/Program (Base Technical Profile)
2	12	2	38	2	12	Class/Program Type (Base Technical Profile)
2	12	2	39	2	12	Resource (Base Technical Profile)
2	12	2	40	2	12	Resource Protcol (Base Technical Profile)
2	12	2	41	2	12	Resource Name Extension (Base Technical Profile)
2	12	2	42	2	12	Software Runtime Profile (Base Technical Profile)
2	12	2	43	2	12	Software Runtime Component (Base Technical Profile)
2	12	2	44	2	12	Patch Level (Base Technical Profile)
2	12	2	45	2	12	Language & Patch Level Composite Extended Key (Base Technical Profile)
2	12	2	46	2	12	Object Set Definition Type (Base Technical Profile)
2	12	2	47	2	12	Link Type Usage Type (Base Technical Profile)
2	12	2	48	2	12	Type Value Based Object Set (Base Technical Profile)
2	12	2	49	2	12	Template Set (Base Technical Profile)
2	12	2	50	2	12	Generic Object (Base Tech Profile)
2	12	2	51	2	12	Generic Object Element Value (Base Tech Profile)
2	12	2	52	2	12	Function Group (Base Tech Profile)
2	12	2	53	2	12	Function Group Type (Base Tech Profile)
2	12	2	54	2	12	Parameter Value Group (Base Tech Profile)
2	12	2	55	2	12	Object Template Path (Base Tech Profile)
2	12	2	56	2	12	Type Definition Path (Base Tech Profile)
2	12	2	57	2	12	Parameter Value (Base Tech Profile)
2	12	2	58	2	12	Statistical Values Object (Base Tech Profile)
2	12	2	59	2	12	Statistical Values Dimension (Base Tech Profile)
2	12	2	60	2	12	Statistic Type (Base Tech Profile) sum, count, mean, mode, median, std dev
2	12	2	61	2	12	Unit of Measure (Base Tech Profile)
2	12	2	62	2	12	Unit of Measure Dimension (Base Tech Profile) length, weight, time, currency
2	12	2	63	2	12	Type Definition Usage Type
2	12	2	64	2	12	System Message Element Definition
2	12	2	65	2	12	System Message Element Definition Type
2	12	2	66	2	12	Numeric Data Type Details
2	13	2	1	2	13	Set of all Object Kinds
2	13	2	2	2	13	Set of all Object Templates
2	13	2	3	2	13	Set of all Object Elements
2	13	2	4	2	13	Set of all Table Elements
2	13	2	5	2	13	Set of all Type Value Based Template Rules
2	13	2	6	2	13	Set of all Link Based Template Rule
2	13	2	7	2	13	Set of all Object Element Value Based Template Rules
2	13	2	8	2	13	Set of all Object Kind Types
2	13	2	9	2	13	Set of all Object Type Definitions
2	13	2	10	2	13	Set of all Link Types
2	13	2	11	2	13	Set of all Link Target Types
2	13	2	12	2	13	Set of all Technical Profiles
2	13	2	13	2	13	Set of all Object Set Definitions
2	13	2	14	2	13	Set of all ZT Instances
2	13	2	15	2	13	Set of all Namespaces
2	13	2	16	2	13	Set of all Object Element Usage Types
2	13	2	17	2	13	Set of all Object Template Types
2	13	2	18	2	13	Set of all Languages
2	13	2	19	2	13	Set of all Users
2	13	2	20	2	13	Set of all User Types
2	13	2	21	2	13	Set of all Code Ranges
2	13	2	22	2	13	Set of all Code Schemes
2	13	2	23	2	13	Set of all Code Scheme Types
2	13	2	24	2	13	Set of all System Message Definitions

2	13	2	25	2	13	Set of all System Message Definition Types
2	13	2	26	2	13	Set of all Parameters
2	13	2	27	2	13	Set of all Parameter Types
2	13	2	28	2	13	Set of all System Message Definition Categories
2	13	2	29	2	13	Set of all System Message Parameter Categories
2	13	2	30	2	13	Set of all Tables
2	13	2	31	2	13	Set of all Data Elements
2	13	2	32	2	13	Set of all Data Types
2	13	2	33	2	13	Set of all Extended Key Definitions
2	13	2	34	2	13	Set of all Extended Key Definition Types
2	13	2	35	2	13	Set of all Object Element Usage Types
2	13	2	36	2	13	Set of all Functions
2	13	2	37	2	13	Set of all Class/Programs
2	13	2	38	2	13	Set of all Class/Program Types
2	13	2	39	2	13	Set of all Resources
2	13	2	40	2	13	Set of all Resource Protcols
2	13	2	41	2	13	Set of all Resource Name Extensions
2	13	2	42	2	13	Set of all Software Runtime Profiles
2	13	2	43	2	13	Set of all Software Runtime Components
2	13	2	44	2	13	Set of all Patch Levels
2	13	2	45	2	13	Set of all Language & Patch Level Composite Extended Keys
2	13	2	46	2	13	Set of all Object Set Definition Types
2	13	2	47	2	13	Set of all Link Type Usage Types
2	13	2	48	2	13	Set of all Type Value Based Object Sets
2	13	2	49	2	13	Set of all Template Sets
2	13	2	50	2	13	Set of all Generic Objects
2	13	2	51	2	13	Set of all Generic Object Element Values
2	13	2	52	2	13	Set of all Function Groups
2	13	2	53	2	13	Set of all Function Group Types
2	13	2	54	2	13	Set of all Parameter Value Groups
2	13	2	55	2	13	Set of all Object Template Paths
2	13	2	56	2	13	Set of all Type Definition Paths
2	13	2	57	2	13	Set of all Parameter Values
2	13	2	58	2	13	Set of all Statistical Values Objects
2	13	2	59	2	13	Set of all Statistical Values Dimensions
2	13	2	60	2	13	Set of all Statistic Types
2	13	2	61	2	13	Set of all Units of Measure
2	13	2	62	2	13	Set of all Unit of Measure Dimensions
2	13	2	63	2	13	Set of all Type Definition Usage Types
2	13	2	64	2	13	Set of all System Message Element Definitions
2	13	2	65	2	13	Set of all System Message Element Definintion Types
2	13	2	66	2	13	Set of all Numeric Data Type Details
2	18	2	1	2	18	English (language)
2	18	2	2	2	18	Spanish (language)
2	21	2	1	2	21	Object Kind (Code Range)
2	21	2	1	2	204	1000
2	21	2	2	2	21	Object Template (Code Range)
2	21	2	2	2	204	2000
2	21	2	3	2	21	Object Element (Code Range)
2	21	2	3	2	204	3000
2	21	2	4	2	21	Table Element (Code Range)
2	21	2	4	2	204	4000
2	21	2	5	2	21	Type Value Based Template Rule (Code Range)
2	21	2	5	2	204	5000
2	21	2	6	2	21	Link Based Template Rule (Code Range)
2	21	2	6	2	204	6000
2	21	2	7	2	21	Object Element Value Based Template Rule (Code Range)
2	21	2	7	2	204	7000
2	21	2	8	2	21	Object Kind Type (Code Range)
2	21	2	8	2	204	8000
2	21	2	9	2	21	Object Type Definition (Code Range)
2	21	2	9	2	204	9000
2	21	2	10	2	21	Link Type (Code Range)
2	21	2	10	2	204	10000
2	21	2	11	2	21	Link Target Type (Code Range)
2	21	2	11	2	204	11000
2	21	2	12	2	21	Technical Profile (Code Range)
2	21	2	12	2	204	12000
2	21	2	13	2	21	Object Set Definition (Code Range)
2	21	2	13	2	204	13050

2	21	2	14	2	21	ZT Instance (Code Range)
2	21	2	14	2	204	14000
2	21	2	15	2	21	Namespace (Code Range)
2	21	2	15	2	204	15050
2	21	2	16	2	21	Object Element Usage Type (Code Range)
2	21	2	16	2	204	16000
2	21	2	17	2	21	Object Template Type (Code Range)
2	21	2	17	2	204	17000
2	21	2	18	2	21	Language (Code Range)
2	21	2	18	2	204	18000
2	21	2	19	2	21	User (Code Range)
2	21	2	19	2	204	19000
2	21	2	20	2	21	User Type (Code Range)
2	21	2	20	2	204	20000
2	21	2	21	2	21	Code Range (Code Range)
2	21	2	21	2	204	21000
2	21	2	22	2	21	Code Scheme (Code Range)
2	21	2	22	2	204	22000
2	21	2	23	2	21	Code Scheme Type (Code Range)
2	21	2	23	2	204	23000
2	21	2	24	2	21	System Message Definition (Code Range)
2	21	2	24	2	204	24000
2	21	2	25	2	21	System Message Definition Type (Code Range)
2	21	2	25	2	204	25000
2	21	2	26	2	21	Parameter (Code Range)
2	21	2	26	2	204	26000
2	21	2	27	2	21	Parameter Type (Code Range)
2	21	2	27	2	204	27000
2	21	2	28	2	21	System Message Definition Category (Code Range)
2	21	2	28	2	204	28000
2	21	2	29	2	21	System Message Parameter Category (Code Range)
2	21	2	29	2	204	29000
2	21	2	30	2	21	Table (Code Range)
2	21	2	30	2	204	30000
2	21	2	31	2	21	Data Element (Code Range)
2	21	2	31	2	204	31000
2	21	2	32	2	21	Data Type (Code Range)
2	21	2	32	2	204	32000
2	21	2	33	2	21	Extended Key Definition (Code Range)
2	21	2	33	2	204	33000
2	21	2	34	2	21	Extended Key Definition Type (Code Range)
2	21	2	34	2	204	34000
2	21	2	35	2	21	Object Element Usage Type (Code Range)
2	21	2	35	2	204	35000
2	21	2	36	2	21	Function (Code Range)
2	21	2	36	2	204	36000
2	21	2	37	2	21	Class/Program (Code Range)
2	21	2	37	2	204	37000
2	21	2	38	2	21	Class/Program Type (Code Range)
2	21	2	38	2	204	38000
2	21	2	39	2	21	Resource (Code Range)
2	21	2	39	2	204	39060
2	21	2	40	2	21	Resource Protcol (Code Range)
2	21	2	40	2	204	40000
2	21	2	41	2	21	Resource Name Extension (Code Range)
2	21	2	41	2	204	41000
2	21	2	42	2	21	Software Runtime Profile (Code Range)
2	21	2	42	2	204	42000
2	21	2	43	2	21	Software Runtime Component (Code Range)
2	21	2	43	2	204	43000
2	21	2	44	2	21	Patch Level (Code Range)
2	21	2	44	2	204	44000
2	21	2	45	2	21	Language & Patch Level Composite Extended Key (Code Range)
2	21	2	45	2	204	45000
2	21	2	46	2	21	Object Set Definiton Type (Code Range)
2	21	2	46	2	204	46000
2	21	2	47	2	21	Link Type Usage Type (Code Range)
2	21	2	47	2	204	47000
2	21	2	48	2	21	Type Value Based Object Set (Code Range)
2	21	2	48	2	204	48000

2	21	2	49	2	21	Template Set (Code Range)
2	21	2	49	2	204	49000
2	21	2	50	2	21	Generic Object (Code Range)
2	21	2	50	2	204	50000
2	21	2	51	2	21	Generic Object Element Value (Code Range)
2	21	2	51	2	204	51000
2	21	2	52	2	21	Function Group (Code Range)
2	21	2	52	2	204	52000
2	21	2	53	2	21	Function Group Type (Code Range)
2	21	2	53	2	204	53000
2	21	2	54	2	21	Parameter Value Group (Code Range)
2	21	2	54	2	204	54000
2	21	2	55	2	21	Object Template Path (Code Range)
2	21	2	55	2	204	55000
2	21	2	56	2	21	Type Definition Path (Code Range)
2	21	2	56	2	204	56000
2	21	2	57	2	21	Parameter Value (Code Range)
2	21	2	57	2	204	57000
2	21	2	58	2	21	Statistical Values Object (Code Range)
2	21	2	58	2	204	58000
2	21	2	59	2	21	Statistical Values Dimension (Code Range)
2	21	2	59	2	204	59000
2	21	2	60	2	21	Statistic Type (Code Range)
2	21	2	60	2	204	60000
2	21	2	61	2	21	Unit of Measure (Code Range)
2	21	2	61	2	204	61000
2	21	2	62	2	21	Unit of Measure Dimension (Code Range)
2	21	2	62	2	204	62000
2	21	2	63	2	21	Type Definition Usage Type (Code Range)
2	21	2	63	2	204	63000
2	21	2	64	2	21	System Message Element Definition (Code Range)
2	21	2	64	2	204	64000
2	21	2	65	2	21	System Message Element Definition Type (Code Range)
2	21	2	65	2	204	65000
2	21	2	66	2	21	Numeric Data Type Details (Code Range)
2	21	2	66	2	204	66000
2	22	2	1	2	22	Code Scheme for pure numeric Object Codes
2	22	2	1	2	205	0123456789
2	24	2	1	2	24	Unexpected Server Error (System Message)
2	24	2	1	2	200	Short Msg: Error in class: &(ds.com/base,1) and Method: &(ds.com/base,2) please investigate short
2	24	2	1	2	201	Unexpected Server Error Long SysMsg: Error in Class: &(ds.com/base,1) and Method: &(ds.com/base,2) please investigate long: &(ds.com/base,3) &(ds.com/base,4)
2	24	2	2	2	24	Unexpected Parse Error (System Message)
2	26	2	1	2	26	Class Name (System Message Parameter)
2	26	2	2	2	26	Class Method (System Message Parameter)
2	26	2	3	2	26	Program source file name (System Message Parameter)
2	26	2	4	2	26	Variable System Message Text (System Message Parameter)
2	26	2	5	2	26	Object Code before increment
2	26	2	6	2	26	Object Code after increment
2	26	2	7	2	26	Code String used for incrementing code
2	31	2	211	2	31	Resource Path/Name--Patch Level Dependent (Data Element)
2	32	2	211	2	32	Resource Path/Name--Patch Level Dependent (Data Type)
2	33	2	1	2	33	No (blank) extended key (extended key definition)
2	33	2	2	2	33	Language (extended key definition)
2	33	2	3	2	33	SoftwarePatch Level (extended key definition)
2	33	2	4	2	33	Language & Patch Level (composite extended key definition)
2	36	2	1	2	36	Function to increment a code (standard)
2	36	2	1	2	36	Function for standard code increment
2	36	2	2	2	36	Function for Server Message Parse
2	36	2	3	2	36	Function for Server Message Query
2	37	2	1	2	37	Class for standard code increment
2	37	2	2	2	37	Class for Server Message Parse
2	37	2	3	2	37	Class for Server Message Query
2	58	2	1	2	58	Standard Link Type Usage Type
2	58	2	2	2	58	Link Type Usage Type for Server Generated Links

Apprendix 8--Base Type Values

Listed below are Type Values from the Base Specification. The numbers in the column headings correspond to Table Elements listed in Appendix 5.

1-OKIC	2-OKC	3-OIC	4-OC	13-TDIC	14-TDC	15-TVIC	16-TVC
2	1	2	1	2	22	2	1
2	1	2	1	2	5	2	1
2	1	2	2	2	22	2	2
2	1	2	2	2	5	2	2
2	1	2	3	2	22	2	3
2	1	2	3	2	5	2	3
2	1	2	4	2	22	2	4
2	1	2	4	2	5	2	4
2	1	2	5	2	22	2	5
2	1	2	5	2	5	2	5
2	1	2	6	2	22	2	6
2	1	2	6	2	5	2	6
2	1	2	7	2	22	2	7
2	1	2	7	2	5	2	7
2	1	2	8	2	22	2	8
2	1	2	8	2	5	2	8
2	1	2	9	2	22	2	9
2	1	2	9	2	5	2	9
2	1	2	10	2	22	2	10
2	1	2	10	2	5	2	10
2	1	2	11	2	22	2	11
2	1	2	11	2	5	2	11
2	1	2	12	2	22	2	12
2	1	2	12	2	5	2	12
2	1	2	13	2	22	2	13
2	1	2	13	2	5	2	13
2	1	2	14	2	22	2	14
2	1	2	14	2	5	2	14
2	1	2	15	2	22	2	15
2	1	2	15	2	5	2	15
2	1	2	16	2	22	2	16
2	1	2	16	2	5	2	16
2	1	2	17	2	22	2	17
2	1	2	17	2	5	2	17
2	1	2	18	2	22	2	18
2	1	2	18	2	5	2	18
2	1	2	19	2	22	2	19
2	1	2	19	2	5	2	19
2	1	2	20	2	22	2	20
2	1	2	20	2	5	2	20
2	1	2	21	2	22	2	21
2	1	2	21	2	5	2	21
2	1	2	22	2	22	2	22
2	1	2	22	2	5	2	22
2	1	2	23	2	22	2	23
2	1	2	23	2	5	2	23
2	1	2	24	2	22	2	24
2	1	2	24	2	5	2	24
2	1	2	25	2	22	2	25
2	1	2	25	2	5	2	25
2	1	2	26	2	22	2	26
2	1	2	26	2	5	2	26
2	1	2	27	2	22	2	27
2	1	2	27	2	5	2	27
2	1	2	28	2	22	2	28
2	1	2	28	2	5	2	28
2	1	2	29	2	22	2	29
2	1	2	29	2	5	2	29
2	1	2	30	2	22	2	30
2	1	2	30	2	5	2	30
2	1	2	31	2	22	2	31
2	1	2	31	2	5	2	31
2	1	2	32	2	22	2	32
2	1	2	32	2	5	2	32
2	1	2	33	2	22	2	33
2	1	2	33	2	5	2	33
2	1	2	34	2	22	2	34
2	1	2	34	2	5	2	34
2	1	2	35	2	22	2	35

2	1	2	35	2	5	2	35
2	1	2	36	2	22	2	36
2	1	2	36	2	5	2	36
2	1	2	37	2	22	2	37
2	1	2	37	2	5	2	37
2	1	2	38	2	22	2	38
2	1	2	38	2	5	2	38
2	1	2	39	2	22	2	39
2	1	2	39	2	5	2	39
2	1	2	40	2	22	2	40
2	1	2	40	2	5	2	40
2	1	2	41	2	22	2	41
2	1	2	41	2	5	2	41
2	1	2	42	2	22	2	42
2	1	2	42	2	5	2	42
2	1	2	43	2	22	2	43
2	1	2	43	2	5	2	43
2	1	2	44	2	22	2	44
2	1	2	44	2	5	2	44
2	1	2	45	2	22	2	45
2	1	2	45	2	5	2	45
2	1	2	46	2	22	2	46
2	1	2	46	2	5	2	46
2	1	2	47	2	22	2	47
2	1	2	47	2	5	2	47
2	1	2	48	2	22	2	48
2	1	2	48	2	5	2	48
2	1	2	49	2	22	2	49
2	1	2	49	2	5	2	49
2	1	2	50	2	22	2	50
2	1	2	50	2	5	2	50
2	1	2	51	2	22	2	51
2	1	2	51	2	5	2	51
2	1	2	52	2	22	2	52
2	1	2	52	2	5	2	52
2	1	2	53	2	22	2	53
2	1	2	53	2	5	2	53
2	1	2	54	2	22	2	54
2	1	2	54	2	5	2	54
2	1	2	55	2	22	2	55
2	1	2	55	2	5	2	55
2	1	2	56	2	22	2	56
2	1	2	56	2	5	2	56
2	1	2	57	2	22	2	57
2	1	2	57	2	5	2	57
2	1	2	58	2	22	2	58
2	1	2	58	2	5	2	58
2	1	2	59	2	22	2	59
2	1	2	59	2	5	2	59
2	1	2	60	2	22	2	60
2	1	2	60	2	5	2	60
2	1	2	61	2	22	2	61
2	1	2	61	2	5	2	61
2	1	2	62	2	22	2	62
2	1	2	62	2	5	2	62
2	1	2	63	2	22	2	63
2	1	2	63	2	5	2	63
2	1	2	64	2	22	2	64
2	1	2	64	2	5	2	64
2	1	2	65	2	22	2	65
2	1	2	65	2	5	2	65
2	1	2	66	2	22	2	66
2	1	2	66	2	5	2	66
2	2	2	1	2	3	2	1
2	2	2	2	2	3	2	2
2	2	2	3	2	3	2	3
2	2	2	4	2	3	2	4
2	2	2	5	2	3	2	5
2	2	2	6	2	3	2	6
2	2	2	7	2	3	2	7

2	2	2	8	2	3	2	8
2	2	2	9	2	3	2	9
2	2	2	10	2	3	2	10
2	2	2	11	2	3	2	11
2	2	2	12	2	3	2	12
2	2	2	13	2	3	2	13
2	2	2	14	2	3	2	14
2	2	2	15	2	3	2	15
2	2	2	16	2	3	2	16
2	2	2	17	2	3	2	17
2	2	2	18	2	3	2	18
2	2	2	19	2	3	2	19
2	2	2	20	2	3	2	20
2	2	2	21	2	3	2	21
2	2	2	22	2	3	2	22
2	2	2	23	2	3	2	23
2	2	2	24	2	3	2	24
2	2	2	25	2	3	2	25
2	2	2	26	2	3	2	26
2	2	2	27	2	3	2	27
2	2	2	28	2	3	2	28
2	2	2	29	2	3	2	29
2	2	2	30	2	3	2	30
2	2	2	31	2	3	2	31
2	2	2	32	2	3	2	32
2	2	2	33	2	3	2	33
2	2	2	34	2	3	2	34
2	2	2	35	2	3	2	35
2	2	2	36	2	3	2	36
2	2	2	37	2	3	2	37
2	2	2	38	2	3	2	38
2	2	2	39	2	3	2	39
2	2	2	40	2	3	2	40
2	2	2	41	2	3	2	41
2	2	2	42	2	3	2	42
2	2	2	43	2	3	2	43
2	2	2	44	2	3	2	44
2	2	2	45	2	3	2	45
2	2	2	46	2	3	2	46
2	2	2	47	2	3	2	47
2	2	2	48	2	3	2	48
2	2	2	49	2	3	2	49
2	2	2	50	2	3	2	50
2	2	2	51	2	3	2	51
2	2	2	52	2	3	2	52
2	2	2	53	2	3	2	53
2	2	2	54	2	3	2	54
2	2	2	55	2	3	2	55
2	2	2	56	2	3	2	56
2	2	2	57	2	3	2	57
2	2	2	58	2	3	2	58
2	2	2	59	2	3	2	59
2	2	2	60	2	3	2	60
2	2	2	61	2	3	2	61
2	2	2	62	2	3	2	62
2	2	2	63	2	3	2	63
2	2	2	64	2	3	2	64
2	2	2	65	2	3	2	65
2	2	2	66	2	3	2	66
2	3	2	211	2	9	2	211
2	9	2	1	2	4	2	1
2	9	2	2	2	4	2	11
2	9	2	3	2	4	2	1
2	9	2	4	2	4	2	1
2	9	2	5	2	4	2	21
2	9	2	6	2	4	2	22
2	9	2	7	2	4	2	13
2	9	2	8	2	4	6	9
2	9	2	9	2	4	2	31
2	9	2	10	2	4	2	32

2	9	2	11	2	4	2	33
2	9	2	12	2	4	2	34
2	9	2	13	2	4	2	9
2	9	2	14	2	4	2	13
2	9	2	15	2	4	2	36
2	9	2	16	2	4	2	37
2	9	2	17	2	4	2	44
2	9	2	18	2	4	2	13
2	9	2	19	2	4	2	14
2	9	2	20	2	4	2	2
2	9	2	21	2	4	2	8
2	9	2	22	2	4	2	2
2	9	2	23	2	4	2	2
2	9	2	24	2	4	2	10
2	9	2	25	2	4	2	2
2	9	2	26	2	4	2	58
2	9	2	27	2	4	2	54
2	9	2	28	2	4	2	36
2	9	2	30	2	4	2	2
2	9	2	31	2	4	2	26
2	9	2	32	2	4	2	60
2	9	2	33	2	4	2	13
2	9	2	34	2	4	2	56
2	9	2	35	2	4	2	61
2	9	2	36	2	4	2	62
2	9	2	37	2	4	2	3
2	9	2	38	2	4	2	2
2	9	2	39	2	4	2	13
2	9	2	40	2	4	2	13
2	9	2	41	2	4	2	13
2	9	2	42	2	4	2	56
2	9	2	43	2	4	2	2
2	9	2	44	2	4	2	13
2	9	2	45	2	4	2	2
2	9	2	46	2	4	2	13
2	9	2	47	2	4	2	3
2	9	2	48	2	4	2	13
2	9	2	49	2	4	2	3
2	10	2	1	2	2	2	1
2	10	2	2	2	2	2	2
2	10	2	3	2	2	2	3
2	10	2	4	2	2	2	4
2	10	2	5	2	2	2	5
2	10	2	6	2	2	2	6
2	10	2	7	2	2	2	7
2	10	2	8	2	2	2	8
2	10	2	9	2	2	2	9
2	10	2	10	2	2	2	10
2	10	2	11	2	2	2	11
2	10	2	12	2	2	2	12
2	10	2	13	2	2	2	13
2	10	2	14	2	2	2	14
2	10	2	15	2	2	2	15
2	10	2	16	2	2	2	16
2	10	2	17	2	2	2	17
2	10	2	18	2	2	2	18
2	10	2	19	2	2	2	19
2	10	2	20	2	2	2	20
2	10	2	21	2	2	2	21
2	10	2	22	2	2	2	22
2	10	2	23	2	2	2	23
2	10	2	24	2	2	2	24
2	10	2	25	2	2	2	25
2	10	2	26	2	2	2	26
2	10	2	27	2	2	2	27
2	10	2	28	2	2	2	28
2	10	2	29	2	2	2	29
2	10	2	30	2	2	2	30
2	10	2	31	2	2	2	31
2	10	2	32	2	2	2	32

2	10	2	33	2	2	2	33
2	10	2	34	2	2	2	34
2	21	2	1	2	6	2	1
2	21	2	2	2	6	2	1
2	21	2	3	2	6	2	1
2	21	2	4	2	6	2	1
2	21	2	5	2	6	2	1
2	21	2	6	2	6	2	1
2	21	2	7	2	6	2	1
2	21	2	8	2	6	2	1
2	21	2	9	2	6	2	1
2	21	2	10	2	6	2	1
2	21	2	11	2	6	2	1
2	21	2	12	2	6	2	1
2	21	2	13	2	6	2	1
2	21	2	14	2	6	2	1
2	21	2	15	2	6	2	1
2	21	2	16	2	6	2	1
2	21	2	17	2	6	2	1
2	21	2	18	2	6	2	1
2	21	2	19	2	6	2	1
2	21	2	20	2	6	2	1
2	21	2	21	2	6	2	1
2	21	2	22	2	6	2	1
2	21	2	23	2	6	2	1
2	21	2	24	2	6	2	1
2	21	2	25	2	6	2	1
2	21	2	26	2	6	2	1
2	21	2	27	2	6	2	1
2	21	2	28	2	6	2	1
2	21	2	29	2	6	2	1
2	21	2	30	2	6	2	1
2	21	2	31	2	6	2	1
2	21	2	32	2	6	2	1
2	21	2	33	2	6	2	1
2	21	2	34	2	6	2	1
2	21	2	35	2	6	2	1
2	21	2	36	2	6	2	1
2	21	2	37	2	6	2	1
2	21	2	38	2	6	2	1
2	21	2	39	2	6	2	1
2	21	2	40	2	6	2	1
2	21	2	41	2	6	2	1
2	21	2	42	2	6	2	1
2	21	2	43	2	6	2	1
2	21	2	44	2	6	2	1
2	21	2	45	2	6	2	1
2	21	2	46	2	6	2	1
2	21	2	47	2	6	2	1
2	21	2	48	2	6	2	1
2	21	2	49	2	6	2	1
2	21	2	50	2	6	2	1
2	21	2	51	2	6	2	1
2	21	2	52	2	6	2	1
2	21	2	53	2	6	2	1
2	21	2	54	2	6	2	1
2	21	2	55	2	6	2	1
2	21	2	56	2	6	2	1
2	21	2	57	2	6	2	1
2	21	2	58	2	6	2	1
2	21	2	59	2	6	2	1
2	21	2	60	2	6	2	1
2	21	2	61	2	6	2	1
2	21	2	62	2	6	2	1
2	21	2	63	2	6	2	1
2	21	2	64	2	6	2	1
2	21	2	65	2	6	2	1
2	21	2	66	2	6	2	1
2	22	2	1	2	15	2	1
2	31	2	211	2	11	2	3

2	33	2	2	2	13	2	33
2	33	2	3	5	9	2	33
2	33	2	4	2	20	2	51
2	36	2	1	2	16	2	1
2	36	2	2	2	16	2	2
2	36	2	3	2	16	2	3

Appendix 9--Base Links

Listed below are Links from the Base Specification. The numbers in the column headings correspond to Table Elements listed in Appendix 5.

1-OKIC	2-OKC	3-OIC	4-OC	17-LTIC	18-LTC	19-LToKIC	20-LToKC	21-LToIC	22-LToC
2	2	2	1	2	1	2	12	2	1
2	2	2	2	2	1	2	12	2	2
2	2	2	3	2	1	2	12	2	3
2	2	2	4	2	1	2	12	2	4
2	2	2	5	2	1	2	12	2	5
2	2	2	6	2	1	2	12	2	6
2	2	2	7	2	1	2	12	2	7
2	2	2	8	2	1	2	12	2	8
2	2	2	9	2	1	2	12	2	9
2	2	2	10	2	1	2	12	2	10
2	2	2	11	2	1	2	12	2	11
2	2	2	12	2	1	2	12	2	12
2	2	2	13	2	1	2	12	2	13
2	2	2	14	2	1	2	12	2	14
2	2	2	15	2	1	2	12	2	15
2	2	2	16	2	1	2	12	2	16
2	2	2	17	2	1	2	12	2	17
2	2	2	18	2	1	2	12	2	18
2	2	2	19	2	1	2	12	2	19
2	2	2	20	2	1	2	12	2	20
2	2	2	21	2	1	2	12	2	21
2	2	2	22	2	1	2	12	2	22
2	2	2	23	2	1	2	12	2	23
2	2	2	24	2	1	2	12	2	24
2	2	2	25	2	1	2	12	2	25
2	2	2	26	2	1	2	12	2	26
2	2	2	27	2	1	2	12	2	27
2	2	2	28	2	1	2	12	2	28
2	2	2	29	2	1	2	12	2	29
2	2	2	30	2	1	2	12	2	30
2	2	2	31	2	1	2	12	2	31
2	2	2	32	2	1	2	12	2	32
2	2	2	33	2	1	2	12	2	33
2	2	2	34	2	1	2	12	2	34
2	2	2	35	2	1	2	12	2	35
2	2	2	36	2	1	2	12	2	36
2	2	2	37	2	1	2	12	2	37
2	2	2	38	2	1	2	12	2	38
2	2	2	39	2	1	2	12	2	39
2	2	2	40	2	1	2	12	2	40
2	2	2	41	2	1	2	12	2	41
2	2	2	42	2	1	2	12	2	42
2	2	2	43	2	1	2	12	2	43
2	2	2	44	2	1	2	12	2	44
2	2	2	45	2	1	2	12	2	45
2	2	2	46	2	1	2	12	2	46
2	2	2	47	2	1	2	12	2	47
2	2	2	48	2	1	2	12	2	48
2	2	2	49	2	1	2	12	2	49
2	2	2	50	2	1	2	12	2	50
2	2	2	51	2	1	2	12	2	51
2	2	2	52	2	1	2	12	2	52
2	2	2	53	2	1	2	12	2	53
2	2	2	54	2	1	2	12	2	54
2	2	2	55	2	1	2	12	2	55
2	2	2	56	2	1	2	12	2	56
2	2	2	57	2	1	2	12	2	57
2	2	2	58	2	1	2	12	2	58

2	2	2	59	2	1	2	12	2	59
2	2	2	60	2	1	2	12	2	60
2	2	2	61	2	1	2	12	2	61
2	2	2	62	2	1	2	12	2	62
2	2	2	63	2	1	2	12	2	63
2	2	2	64	2	1	2	12	2	64
2	2	2	65	2	1	2	12	2	65
2	2	2	66	2	1	2	12	2	66
2	11	2	1	2	6	2	2	2	12
2	11	2	2	2	6	2	2	2	12
2	11	2	3	2	6	2	2	2	3
2	11	2	4	2	6	2	2	2	9
2	11	2	5	2	6	2	2	2	10
2	11	2	6	2	6	2	2	2	2
2	11	2	7	2	6	2	2	2	26
2	11	2	8	2	6	2	2	2	26
2	11	2	9	2	6	2	2	2	39
2	11	2	10	2	6	2	2	2	43
2	11	2	11	2	6	2	2	2	43
2	11	2	12	2	6	2	2	2	1
2	11	2	13	2	6	2	2	2	47
2	11	2	14	2	6	2	2	2	9
2	11	2	15	2	6	2	2	2	2
2	11	2	16	2	6	2	2	2	13
2	11	2	17	2	6	2	2	2	13
2	11	2	18	2	6	2	2	2	13
2	11	2	19	2	6	2	2	2	13
2	11	2	20	2	6	2	2	2	13
2	11	2	21	2	6	2	2	2	36
2	11	2	22	2	6	2	2	2	31
2	11	2	23	2	6	2	2	2	15
2	11	2	24	2	6	2	2	2	15
2	11	2	25	2	6	2	2	2	12
2	11	2	26	2	6	2	2	2	59
2	11	2	27	2	6	2	2	2	9
2	11	2	28	2	6	2	2	2	36
2	11	2	29	2	6	2	2	2	5
2	11	2	30	2	6	2	2	2	50
2	11	2	31	2	6	2	2	2	24
2	11	2	32	2	6	2	2	2	36
2	12	2	1	2	3	2	3	2	1
2	12	2	1	2	4	2	9	2	5
2	12	2	1	2	4	2	9	2	21
2	12	2	1	2	4	2	9	2	22
2	12	2	1	2	4	2	9	2	23
2	12	2	2	2	3	2	3	2	2
2	12	2	2	2	4	2	9	2	3
2	12	2	2	2	5	2	10	2	1
2	12	2	3	2	3	2	3	2	3
2	12	2	3	2	4	2	9	2	9
2	12	2	4	2	3	2	3	2	4
2	12	2	5	2	3	2	3	2	5
2	12	2	5	2	4	2	9	2	38
2	12	2	5	2	4	2	9	2	39
2	12	2	6	2	3	2	3	2	6
2	12	2	7	2	3	2	3	2	7
2	12	2	8	2	3	2	3	2	8
2	12	2	9	2	3	2	3	2	9
2	12	2	9	2	4	2	9	2	4
2	12	2	9	2	4	2	9	2	24
2	12	2	9	2	4	2	9	2	25
2	12	2	9	2	4	2	9	2	14
2	12	2	9	2	4	2	9	2	29
2	12	2	9	2	5	2	10	2	29
2	12	2	10	2	3	2	3	2	10
2	12	2	10	2	4	2	9	2	2
2	12	2	11	2	3	2	3	2	11
2	12	2	11	2	5	2	10	2	6
2	12	2	12	2	3	2	3	2	12
2	12	2	12	2	5	2	10	2	3

2	12	2	12	2	5	2	10	2	4
2	12	2	12	2	5	2	10	2	5
2	12	2	12	2	5	2	10	2	22
2	12	2	12	2	5	2	10	2	2
2	12	2	13	2	3	2	3	2	13
2	12	2	13	2	4	2	9	2	19
2	12	2	13	2	5	2	10	2	12
2	12	2	13	2	5	2	10	2	14
2	12	2	13	2	5	2	10	2	15
2	12	2	13	2	5	2	10	2	19
2	12	2	13	2	5	2	10	2	24
2	12	2	13	2	5	2	10	2	30
2	12	2	14	2	3	2	3	2	14
2	12	2	15	2	3	2	3	2	15
2	12	2	15	2	3	2	3	2	215
2	12	2	16	2	3	2	3	2	16
2	12	2	17	2	3	2	3	2	17
2	12	2	18	2	3	2	3	2	18
2	12	2	19	2	3	2	3	2	19
2	12	2	20	2	3	2	3	2	20
2	12	2	21	2	3	2	3	2	21
2	12	2	21	2	4	2	9	2	6
2	12	2	21	2	3	2	3	2	202
2	12	2	21	2	3	2	3	2	203
2	12	2	21	2	3	2	3	2	204
2	12	2	21	2	3	2	3	2	206
2	12	2	21	2	3	2	3	2	207
2	12	2	21	2	3	2	3	2	208
2	12	2	21	2	3	2	3	2	209
2	12	2	22	2	3	2	3	2	22
2	12	2	22	2	4	2	9	2	15
2	12	2	22	2	3	2	3	2	205
2	12	2	23	2	3	2	3	2	23
2	12	2	24	2	3	2	3	2	24
2	12	2	24	2	5	2	10	2	7
2	12	2	24	2	3	2	3	2	200
2	12	2	24	2	3	2	3	2	201
2	12	2	25	2	3	2	3	2	25
2	12	2	26	2	3	2	3	2	26
2	12	2	27	2	3	2	3	2	27
2	12	2	28	2	3	2	3	2	28
2	12	2	29	2	3	2	3	2	29
2	12	2	30	2	3	2	3	2	30
2	12	2	31	2	3	2	3	2	31
2	12	2	31	2	4	2	9	2	10
2	12	2	31	2	4	2	9	2	11
2	12	2	31	2	3	2	3	2	214
2	12	2	32	2	3	2	3	2	32
2	12	2	33	2	3	2	3	2	33
2	12	2	33	2	4	2	9	2	12
2	12	2	33	2	4	2	9	2	13
2	12	2	33	2	4	2	9	2	20
2	12	2	34	2	3	2	3	2	34
2	12	2	35	2	3	2	3	2	35
2	12	2	36	2	3	2	3	2	36
2	12	2	36	2	4	2	9	2	16
2	12	2	36	2	5	2	10	2	8
2	12	2	37	2	3	2	3	2	37
2	12	2	37	2	5	2	10	2	9
2	12	2	38	2	3	2	3	2	38
2	12	2	39	2	3	2	3	2	39
2	12	2	39	2	3	2	3	2	210
2	12	2	39	2	3	2	3	2	211
2	12	2	39	2	3	2	3	2	212
2	12	2	39	2	3	2	3	2	213
2	12	2	40	2	3	2	3	2	40
2	12	2	41	2	3	2	3	2	41
2	12	2	42	2	3	2	3	2	42
2	12	2	42	2	5	2	10	2	10
2	12	2	43	2	3	2	3	2	43

2	12	2	43	2	4	2	9	2	7
2	12	2	43	2	4	2	9	2	8
2	12	2	43	2	4	2	9	2	17
2	12	2	43	2	4	2	9	2	18
2	12	2	43	2	5	2	10	2	11
2	12	2	44	2	3	2	3	2	44
2	12	2	45	2	3	2	3	2	45
2	12	2	46	2	3	2	3	2	46
2	12	2	47	2	3	2	3	2	47
2	12	2	48	2	3	2	3	2	48
2	12	2	48	2	3	2	3	2	224
2	12	2	48	2	3	2	3	2	225
2	12	2	49	2	3	2	3	2	49
2	12	2	50	2	3	2	3	2	50
2	12	2	50	2	3	2	3	2	220
2	12	2	50	2	3	2	3	2	221
2	12	2	50	2	3	2	3	2	222
2	12	2	50	2	3	2	3	2	223
2	12	2	51	2	3	2	3	2	51
2	12	2	52	2	3	2	3	2	52
2	12	2	52	2	5	2	10	2	31
2	12	2	52	2	5	2	10	2	32
2	12	2	53	2	3	2	3	2	53
2	12	2	54	2	3	2	3	2	54
2	12	2	55	2	3	2	3	2	55
2	12	2	56	2	3	2	3	2	56
2	12	2	56	2	5	2	10	2	27
2	12	2	57	2	3	2	3	2	57
2	12	2	58	2	3	2	3	2	58
2	12	2	58	2	4	2	9	2	32
2	12	2	58	2	4	2	9	2	33
2	12	2	58	2	4	2	9	2	35
2	12	2	58	2	4	2	9	2	37
2	12	2	58	2	5	2	10	2	26
2	12	2	59	2	3	2	3	2	59
2	12	2	59	2	4	2	9	2	34
2	12	2	59	2	3	2	3	2	216
2	12	2	59	2	3	2	3	2	217
2	12	2	59	2	3	2	3	2	218
2	12	2	59	2	3	2	3	2	219
2	12	2	60	2	3	2	3	2	60
2	12	2	61	2	3	2	3	2	61
2	12	2	61	2	4	2	9	2	36
2	12	2	62	2	3	2	3	2	62
2	12	2	63	2	3	2	3	2	63
2	12	2	64	2	3	2	3	2	64
2	12	2	65	2	3	2	3	2	65
2	12	2	66	2	3	2	3	2	66
2	13	2	1	2	12	2	1	2	1
2	13	2	2	2	12	2	1	2	2
2	13	2	3	2	12	2	1	2	3
2	13	2	4	2	12	2	1	2	4
2	13	2	5	2	12	2	1	2	5
2	13	2	6	2	12	2	1	2	6
2	13	2	7	2	12	2	1	2	7
2	13	2	8	2	12	2	1	2	8
2	13	2	9	2	12	2	1	2	9
2	13	2	10	2	12	2	1	2	10
2	13	2	11	2	12	2	1	2	11
2	13	2	12	2	12	2	1	2	12
2	13	2	13	2	12	2	1	2	13
2	13	2	14	2	12	2	1	2	14
2	13	2	15	2	12	2	1	2	15
2	13	2	16	2	12	2	1	2	16
2	13	2	17	2	12	2	1	2	17
2	13	2	18	2	12	2	1	2	18
2	13	2	19	2	12	2	1	2	19
2	13	2	20	2	12	2	1	2	20
2	13	2	21	2	12	2	1	2	21
2	13	2	22	2	12	2	1	2	22

2	13	2	23	2	12	2	1	2	23
2	13	2	24	2	12	2	1	2	24
2	13	2	25	2	12	2	1	2	25
2	13	2	26	2	12	2	1	2	26
2	13	2	27	2	12	2	1	2	27
2	13	2	28	2	12	2	1	2	28
2	13	2	29	2	12	2	1	2	29
2	13	2	30	2	12	2	1	2	30
2	13	2	31	2	12	2	1	2	31
2	13	2	32	2	12	2	1	2	32
2	13	2	33	2	12	2	1	2	33
2	13	2	34	2	12	2	1	2	34
2	13	2	35	2	12	2	1	2	35
2	13	2	36	2	12	2	1	2	36
2	13	2	37	2	12	2	1	2	37
2	13	2	38	2	12	2	1	2	38
2	13	2	39	2	12	2	1	2	39
2	13	2	40	2	12	2	1	2	40
2	13	2	41	2	12	2	1	2	41
2	13	2	42	2	12	2	1	2	42
2	13	2	43	2	12	2	1	2	43
2	13	2	44	2	12	2	1	2	44
2	13	2	45	2	12	2	1	2	45
2	13	2	46	2	12	2	1	2	46
2	13	2	47	2	12	2	1	2	47
2	13	2	48	2	12	2	1	2	48
2	13	2	49	2	12	2	1	2	48
2	13	2	50	2	12	2	1	2	50
2	13	2	51	2	12	2	1	2	51
2	13	2	52	2	12	2	1	2	52
2	13	2	53	2	12	2	1	2	53
2	13	2	54	2	12	2	1	2	54
2	13	2	55	2	12	2	1	2	55
2	13	2	56	2	12	2	1	2	56
2	13	2	57	2	12	2	1	2	57
2	13	2	58	2	12	2	1	2	58
2	13	2	59	2	12	2	1	2	59
2	13	2	60	2	12	2	1	2	60
2	13	2	61	2	12	2	1	2	61
2	13	2	62	2	12	2	1	2	62
2	13	2	63	2	12	2	1	2	63
2	13	2	64	2	12	2	1	2	64
2	13	2	65	2	12	2	1	2	65
2	13	2	66	2	12	2	1	2	66
2	24	2	1	2	7	2	26	2	1
2	24	2	1	2	7	2	26	2	2
2	24	2	1	2	7	2	26	2	3
2	24	2	1	2	7	2	26	2	4
2	24	2	1	2	8	2	3	2	200
2	24	2	1	2	9	2	3	2	201
2	37	2	1	2	9	2	39	6	1
2	37	2	3	2	9	2	39	6	3