A system supplies links between objects. A link service receives a link request from a client. The request identifies a source object. The link service aggregates links from link providers for which the source object is a source of the links, and provides the aggregated links to the client.
FIG. 1
FIG. 3
CLIENT GENERATES A REQUEST FOR LINKS IN TERMS OF A SOURCE NODE (A SOURCE ENTITY)

HMS RECEIVES REQUEST

HMS CHECKS REGISTER INFORMATION TO IDENTIFY PROVIDERS THAT PROVIDE THE REQUESTED LINKS

HMS FORWARDS THE REQUEST TO THE IDENTIFIED PROVIDERS

PROVIDER RETURNS THE REQUESTED LINKS FOR WHICH THE IDENTIFIED NODE IS THE SOURCE TO HMS

HMS AGGREGATES LINKS FROM PROVIDERS AND RETURNS THEM TO THE CLIENT

FIG. 4
FIG. 5

FIG. 8
CLIENT SENDS LINK AND TRAVERSAL REQUEST TO HMS

HMS IDENTIFIES THE PROVIDER THAT SENT THE LINK

HMS FORWARDS THE TRAVERSAL REQUEST AND LINK TO THE IDENTIFIED PROVIDER

PROVIDER RETURNS TRAVERSAL RESULTS (DESTINATION NODE OR ACTION REPRESENTED BY THE LINK) TO THE HMS

HMS RETURNS TRAVERSAL RESULT TO THE CLIENT

CLIENT HANDLER HANDLES THE TRAVERSAL RESULTS

FIG. 6
HMS RECEIVES REQUEST TO REGISTER A PROVIDER

HMS CHECKS SECURITY OF THE REQUESTER

HMS STORES INFORMATION THAT INDICATES THE EXISTENCE OF THE PROVIDER

HMS REQUESTS INFORMATION ABOUT LINK CATEGORIES, LINK TYPES, AND NODE CLASSES FOR WHICH THE PROVIDER PROVIDES LINKS

INFORMATION IS RETURNED BY THE PROVIDER WITH A VERSION IDENTIFIER

THE INFORMATION IS CACHED BY THE HMS IN THE PROVIDER REGISTER

FIG. 7
HMS RECEIVES LINK REQUEST 404 FROM CLIENT

HMS FORWARDS REQUEST TO METADATA PROVIDER 406

METADATA PROVIDER ACCESSES METADATA STORE TO FIND ASSOCIATIONS 408

FOR EACH ASSOCIATION, PROVIDER CREATES A LINK 410

LINKS RETURNED TO HMS 412

HMS AGGREGATES AND RETURNS LINKS TO CLIENT 414

FIG. 10
HYPERMEDIA MANAGEMENT SYSTEM

BACKGROUND OF THE INVENTION

[0001] The present invention relates to hypermedia links. More specifically, the present invention relates to managing hypermedia links between objects.

[0002] A number of different databases will first be discussed, although it will be appreciated that the objects need not reside in a database at all. In conventional relational databases that can be used to store the objects, all data are stored in named tables. The tables are described by their features. In other words, the rows of each table contain items of identical type, and the definitions of the columns of the table (i.e., the column names and the data types stored in the column) describe the attributes of each of the instances of the object. By identifying its name, its column names and the data types of the column contents, a table is completely described. Queries to a relational data base are formulated in a query language. One such language is SQL (Structured Query Language) which is widely used in commercial relational data base systems. The data types offered by SQL can be classified as character arrays (names), numbers, and data types related to date and time. Tables can be modified or combined by several operations of relational algebra such as the application of Boolean operators, projection (i.e., selection of columns) or the Cartesian product.

[0003] Relational databases offer several advantages. Database queries are based on a comparison of the table contents. Thus, no pointers are required in relational databases, and all relations are treated uniformly. Further, the tables are independent (they are not related by pointers), so it is easier to maintain dynamic data sets. The tables are easily expandable by simply adding new columns. Also, it is relatively easy to create user-specific views from relational databases.

[0004] There are, however, a number of disadvantages associated with relational databases as well. For example, access to data by reference to properties is not optimal in the classical relational data model. This can make such databases cumbersome in many applications.

[0005] Another recent technology for database systems is referred to as object oriented data base systems. These systems offer more complex data types in order to overcome the restrictions of conventional relational databases. In the context of object oriented data base models, an "object" includes both data and the methods which can be applied to the object. Each object is a concrete instance of an object class defining the attributes and methods of all its instances. Each instance has its unique identifier by which it can be referred to in the database.

[0006] Object oriented databases operate under a number of principles. One such principle is referred to as inheritance. Inheritance means that new object classes can be derived from another class. The new classes inherit the attributes and methods of the other class (the super-class) and offer additional attributes and operations. An instance of the derived class is also an instance of the super-class. Therefore, the relation between a derived class and its super-class is referred to as the "isa" relation.

[0007] A second principle related to object oriented databases is referred to as “aggregation.” Aggregation means that composite objects may be constructed as consisting of a set of elementary objects. A “container object” can communicate with the objects contained therein by their methods of the contained objects. The relation between the container object and its components is called a “partOf” relation because a component is a part of the container object.

[0008] Yet another principle related to object oriented databases is referred to as encapsulation. According to encapsulation, an application can only communicate with an object through messages. The operations provided by an object define the set of messages which can be understood by the object. No other operations can be applied to the object.

[0009] Another principle related to object oriented databases is referred to as polymorphism. Polymorphism means that derived classes may re-define methods of their super-classes.

[0010] Objects present a variety of advantages. For example, operations are an important part of objects. Because the implementations of the operations are hidden to an application, objects can be more easily used by application programs. Further, an object class can be provided as an abstract description for a wide variety of actual objects, and new classes can be derived from the base class. Thus, if an application knows the abstract description and using only the methods provided by the application can still accommodate objects of the derived classes, because the objects in the derived classes inherit these methods. However, object oriented data bases are not yet as widely used in commercial products as relational databases.

[0011] Yet another database technology attempts to combine the advantages of the wide acceptance of relational data bases and the benefits of the object oriented paradigm. This technology is referred to as object-relational database systems. These databases employ a data model that attempts to add object oriented characteristics to tables. All persistent (database) information is still in tables, but some of the tabular entries can have richer data structure. These data structures are referred to as abstract data types (ADTs). An ADT is a data type that is constructed by combining basic alphanumeric data types. The support for abstract data types presents certain advantages. For example, the operations and methods associated with the new data type can be used to index, store, and retrieve records based on the content of the new data type.

[0012] Some conventional object-relational databases support an extended form of SQL, sometimes referred to as ObjectSQL. The extensions are provided to support the object model (e.g., queries involving object attributes). However, these object-relational databases are still relational because the data is stored in tables of rows and columns, and SQL, with some extensions, is the language for data definition, manipulation, and query. Both the target of a query and the result of a query are still tables. The extended SQL language is often still the primary interface to the database. Therefore, there is no direct support of host object languages and their objects. This forces programmers to continue to translate between objects and tables.

[0013] Thus, in prior object-relational databases, an object can be queried for in terms of the object’s fields, rather than using the relational database column names.
However, a number of problems exist with respect to conventional user interface (UI) technology for object-relational technology and other databases or environments (other than databases) where links between objects are desired.

Conventional user interfaces are hand written. This includes the links between different pieces of data in the databases. Each time such a link is desired, it must be hand written again. For example, if an order entry page has been coded to include a link that references customer information, that link has typically been placed by hand. If the order appears elsewhere in the user interface, the link must be hand coded again. Therefore, if a third party integrates an application to the order object, the order page will not show that information or provide links to it, because such links have not been hand coded, even though there may be a link in a representation of the order object that is independent of the UI.

SUMMARY OF THE INVENTION

A system supplies links between objects. A link service receives a link request from a client. The request identifies a source object. The link service aggregates links from link providers for which the source object is a source of the links, and provides the aggregated links to the client.

In one embodiment, the link request identifies an instance of the source object and the link service aggregates links based on the instance of the source object. In another embodiment, the link request identifies the type of the source object and the link service aggregates the links based on the type.

Link providers can register with the link service so they can provide links in response to requests from clients. When a link provider requests to register with the link service, or another system makes the request on behalf of the link provider, the link service can query the link provider to obtain information indicative of the links that the link provider will provide. Traversal of links yields a link result from the link providers. The traversal result can be a destination object, which is the destination of the link, or an action, or a combination of an action and a destination object (or other targets) which is represented by the link.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of one embodiment of an object-relational data storage system.

FIG. 2 is a block diagram of an environment in which the present invention can be used.

FIG. 3 is a block diagram of a hypermedia service system in accordance with one embodiment of the present invention.

FIG. 4 is a flow diagram illustrating how link requests are processed by the hypermedia service system in accordance with one embodiment of the present invention.

FIG. 5 is an illustration of an exemplary user interface.

FIG. 6 is a flow diagram illustrating how links are traversed by the system in accordance with one embodiment of the present invention.

FIG. 7 is a flow diagram illustrating how hypermedia providers are registered in accordance with one embodiment of the present invention.

FIG. 8 shows an association between objects.

FIG. 9 is a block diagram illustrating another embodiment of a hypermedia system.

FIG. 10 is a flow diagram illustrating the operation of the system shown in FIG. 9.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

The present invention relates to maintaining links between objects. In one exemplary embodiment, the links are between objects in an object-relational (O-R) database, although the objects need not be stored in a database at all, and the present invention can still provide benefits. However, for the sake of the present example, prior to discussing details of the present invention, an environment in which the invention may be used will be discussed.

FIG. 1 is a block diagram illustrating one embodiment of a data storage and accessing system 10 in accordance with the present invention. System 10 includes data access system (or entity persistence system) 12, relational database 16, and class-table mapping 18. System 10 is illustratively an object-relational (O-R) data storage system in which stored data can be referred to in terms of entities (or objects) and their properties, rather than elements of the data base schema, such as tables and columns. FIG. 1 illustrates one mechanism for doing this.

As shown in FIG. 1, the data can be organized in terms of entities 20 (which is used interchangeably with the term objects). Each entity illustratively includes a metadata portion 22 and a remaining attributes portion 24. The metadata portion 22 describes the entity 20, while the remaining attributes 24 define further attributes of entity 20, such as the data stored therein. Each of the attributes in entity 20 is mapped to a corresponding entity table 26 and a specific column 28 in a given entity table 26.

Data access system 12 can receive forms of a request such as a query 30 which specifies an entity, or portions of an entity or group of entities, to be retrieved. Query 30 can illustratively be expressed in terms of objects ("entities") and properties rather than in terms of tables and columns. The particular manner in which queries are expressed does not form part of the present invention.

Data access system 12 receives the query 30 and accesses class-table mapping 18. In this way, data access system 12 can determine the location of the data for the entities identified by query 30. Data access system 12 includes a translator 13 that translates query 30 into a relational database query 32 which is suitable for input to relational data store mechanism 14. In one illustrative embodiment, relational data store mechanism 14 is a server that operates according to the SQL programming language in accessing relational database 16. Therefore, data access system 12 receives queries 30 in terms of objects and translates those queries into an appropriate relational database query 32 that is then provided to the data store mechanism (or server) 14 which actually accesses the data in relational database 16.
Relational data store mechanism retrieves the requested data and returns it in the form of relational database results. The results are returned to data access system which then formulates the relational database results into a requested result set. In an illustrative embodiment, result set is requested in query. Query may request that the results be output in the form of one or more objects or simply as a data set. In any case, data access system arranges the relational database results into the proper format and outputs them as result set.

FIG. 2 illustrates an example of a suitable computing system environment on which the invention may be implemented. The computing system environment is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment.

The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, handheld or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

With reference to FIG. 2, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer. Components of the computer may include, but are not limited to, a processing unit, a system memory, and a system bus that couples various system components including the system memory to the processing unit. The system bus may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

Computer typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier WAV or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, FR, infrared and other wireless media.

The system memory includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) and random access memory (RAM). A basic input/output system (BIOS), containing the basic routines that help to transfer information between elements within computer, such as during startup, is typically stored in ROM. RAM typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit. By way of example, and not limitation, FIG. 2 illustrates operating system, application programs, other program modules, and program data.

The computer may also include other removable/non-removable volatile/nonvolatile computer storage media. By way of example only, FIG. 2 illustrates a hard disk drive that reads from or writes to a removable, nonvolatile magnetic media, a magnetic disk drive that reads from or writes to a removable, nonvolatile magnetic disk, and an optical disk drive that reads from or writes to a removable, nonvolatile optical disk such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive is typically connected to the system bus through a non-removable memory interface such as interface and magnetic disk drive and optical disk drive are typically connected to the system bus by a removable memory interface, such as interface.

The drives and their associated computer storage media discussed above and illustrated in FIG. 2, provide storage of computer readable instructions, data structures,
program modules and other data for the computer 110. In FIG. 2, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies.

A user may enter commands and information into the computer 110 through input devices such as a keyboard 162, a microphone 163, and a pointing device 161, such as a mouse, trackball or touch pad. Other input devices (not shown) may include a joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus 200, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 190.

The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a handheld device, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110. The logical connections depicted in FIG. 2 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user-input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, FIG. 2 illustrates remote application programs 185 as residing on remote computer 180. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

It should be noted that the present invention can be carried out on a computer system such as that described with respect to FIG. 2. However, the present invention can be carried out on a server, a computer devoted to message handling, or on a distributed system in which different portions of the present invention are carried out on different parts of the distributed computing system.

As stated in the background, conventional user interfaces have traditionally had any links between objects hand coded. This presents a number of disadvantages. One aspect of the present invention is a system for creating links among objects or entities based on logical relationships between those objects, when no physical relationship necessarily exists. One embodiment of the invention allows the logical relationships among such entities to be surfaced as links in a hyperspace, thus making the entities themselves the nodes of the hyperspace.

In general, “hypermedia” is referred to as a mechanism for navigating a hyperspace which is comprised of a set of nodes and the hypermedia links that join those nodes. One embodiment of a hypermedia architecture is illustrated in FIG. 3. The hypermedia architecture 200 shows a client 202 that communicates with hypermedia service 204. Hypermedia service 204 accesses a provider register 206 and also communicates with a set of hypermedia providers 208, 210 and 212. Hypermedia service (HMS) 204 is illustratively the central point where clients 202 request hypermedia (i.e., hypermedia links or simply links). Hypermedia providers 208-212 are registered with HMS 204. Providers 208-212 are the points at which the links are actually created. New providers 208-212 can be registered with HMS 204, thus allowing extensibility.

The data that is transferred between client 202 and HMS 204, and between HMS 204 and providers 208-212 conforms, in one illustrative embodiment, to an XML schema attached as an exhibit hereto. The definition of a “link” in the schema includes a link category. Link categories are discussed in greater detail below. Suffice it to say for now that new categories may be defined by a hypermedia provider, thus allowing additional extensibility.

FIG. 4 is a flow diagram better illustrating how a client 202 requests links from hypermedia system 200, and it will be described in conjunction with FIG. 3. In one embodiment, the HMS 204 and link providers 208-212 are objects that expose application programming interfaces (APIs) having methods to accomplish the functionality described. The specific details of the API are described in greater detail in the appendix hereto. Assume that client 202 is displaying a list of customer entities (where the term “entities” is used interchangeably with the term “object”). Assume also that the client wishes to display a set of links to the user for possible traversal. Client 202 first generates a request for links in terms of a source node (i.e., a source entity or a source object). In this example, the client would request the links for the customer object. The request also illustratively specifies which categories of links to be retrieved. The links can represent relationships between nodes, as well as actions that can be performed on nodes, or any other destinations specified.

There are three types of links that can be retrieved: class links, instance links and instance specific links. Class links have the context of a class. They represent either a relationship to a destination node, or an action that can be performed. A class link provides information that is indicative of where a client can traverse to, or what operations can be performed, as they pertain to a particular class.

Instance and instance specific links have the context of an instance. The difference between the two types is
that instance specific links are directly tied to a specific instance of an entity (or object), and instance links are tied to a class, but an instance of that class must be specified in order to traverse the link. An instance or instance specific link provides information indicative of where the client can traverse to, and what operations can be performed, with the particular instance being examined.

All three types of links can be traversed, which is discussed in greater detail below with respect to FIG. 6. Traversal returns the destination node of the link. If the link represents an action, traversal performs the action the link represents, and may or may not return a destination node.

In accordance with one embodiment of the present invention, the type of link is not the only manner in which links may be grouped. Links also illustratively belong to a link category. One example of a link category is a meta-model category. This is described in greater detail below. Briefly, however, links that belong to this link category represent associations between entities. These associations are captured in the metamodel (or object model) of the system.

In addition to being a grouping mechanism, a link category also defines a protocol that is followed by the particular providers that supplied the link. The link category gives client an indication of what type of information the link represents, and what type of object will result from traversal of the link. All links of the same category can be handled in the same manner. In this way, client is able to determine how to handle a link based on the link category to which it belongs.

Therefore, client may generate the hypermedia request by requesting class, instance, or instance specific links, or a combination of these. The client can also specify a set of link categories, and only links from those categories will be returned. If no category is specified, links from all categories will be returned. In any case, once client has generated the request for links, and identified a node which will serve as the source of the link, it provides the request to HMS 204. Generating the request and providing it to HMS 204 is indicated by blocks 214 and 216 in FIG. 4.

HMS 204 then checks register 206 for a particular providers that provide links that have, as a source node, the node identified by client 202 as the source of the links. This is indicated by block 218. In other words, during the registration process, providers 208-212 provided HMS 204 with information about the link categories, link types, and node classes for which the provider provides links. This information is stored in register 206. Therefore, when HMS 204 receives a request from client 202, it checks register 206 to determine which providers should access.

HMS 204 then forwards the request generated by client 202 to the identified providers which will provide links having the source node identified by client 202 as the source of the link. This is indicated by block 220. The providers which receive the request, in turn, return the requested links to HMS 204. This is indicated by block 222. HMS 204 then aggregates all of the links from the providers which have responded with links, and returns those links to client 202. This is indicated by block 224 in FIG. 4.

FIG. 5 shows an exemplary user interface where links have already been requested. In FIG. 5, the user interface generator that generates display 300 is currently rendering a customer list that displays information about a plurality of customer objects 302, 304, 306 and 308. When the user selects a certain customer object, the user interface generates a request requesting links from HMS 204 where the selected customer object 302 is the source link. HMS 204 goes through the process described with respect to FIG. 4 and returns a set of links for which customer entity 302 (which can also be referred to as customer node 302) is the source node. The user interface generating the display can then display the links 310 as desired. In the example shown in FIG. 5, the links for which customer node 302 is the source node include an address link identifying an Address node associated with customer node 302, an orders link which links to an Orders node corresponding to customer node 302 and a general information link which links to a General Information node corresponding to customer node 302.

In one example, to traverse one of the links 310, the user simply selects it (such as by clicking on it with the mouse cursor). FIG. 6 is a flow diagram illustrating what happens when a link is traversed. After being selected by the user, client 202 sends the link, along with the traversal request, to HMS 204. This is indicated by block 350 in FIG. 6. The link provided by client 202 is illustratively a class link or an instance specific link.

HMS 204 then identifies the particular provider that provided the link. This is indicated by block 352. The specific links supplied by the providers in response to requests is maintained by HMS 204 by adding this information to the link during the hypermedia request. This information is then examined during a traversal request. Therefore, HMS 204 can identify the provider which supplied the link.

HMS 204 then forwards the traversal request and link to the identified provider. This is indicated by block 354. The provider traverses the link, returning traversal results to HMS 204. This is indicated by block 356. The traversal results can include a destination node, which is the destination of the link, or performance of an action represented by the link or both, or other targets represented by the link. For example, if the link represents the relationship between a customer and a query of the customer’s orders, traversing the link entails the provider returning the query (which is an entity, like the customer), not the results of executing the query. Likewise, if the link represents the relationship between a customer and a URL of a particular web page containing a map of the customer’s address, traversing the link returns the URL, it does not open a browser window displaying the page pointed to by the URL. In that example, the link does contain a destination (the URL), but the destination is not an entity or an action. Thus, the traversal result may return an entity, some result which is not an entity, or there may be no substantive result returned. If the link represents an action, traversing the link performs the action and the result may indicate this. Combinations of these types of results can occur as well.

HMS 204, in turn, returns the traversal results to client 202. This is indicated by block 358. Client 202 illustratively includes a handler that handles the traversal results. For example, if the traversal
returns a query, the client handler executes that query against the database and displays the results to the user. If the query returns a URL, the client handler opens a browser window displaying the page pointed to by the URL, etc. Handling the traversal results at the client is indicated by block 360 in FIG. 6.

[0067] FIG. 7 is a flow diagram illustrating how a provider is registered with HMS 204. First, HMS 204 receives a request to register a provider. This is indicated by block 362.

[0068] HMS 204 then checks the security of the requester. This is indicated by block 364. This involves determining whether the requester has authorization to register a provider and can be done in any known way. This is indicated by block 366.

[0069] HMS 204 then requests information from the provider about the link categories, link types and node classes for which the provider provides links. This is indicated by block 368.

[0070] The provider returns the information, along with a version identifier. This is indicated by block 370.

[0071] HMS 204 then caches the information in provider register 206, and information confirming the success of the registration is returned to the requester. This is indicated by block 372. HMS 204 is then in position to receive requests for links provided by the newly registered provider.

[0072] A number of other features of the present invention should be noted. It can be seen from the architecture that any number of providers can be added, at any time. Third party providers who integrate applications to the entities or objects stored in the database can be added and links to those third party applications will automatically be returned by HMS 204, so long as an appropriate provider is registered with HMS 204.

[0073] Also, third parties can define new link categories for which their providers will provide links. HMS 204 can operate with no knowledge about what the links are, only knowing that the provider will provide those links. The same is true for new links. An existing provider can add new links and they will be provided when requested. The developer thus need not hard code the links into the system. In one illustrative embodiment, the provider simply need to implement an interface known to HMS 204, such as those described in the Appendix hereto.

[0074] In another illustrative embodiment, the interface implemented by HMS 204 derives from the interface implemented by the providers. Therefore, each HMS 204 can also be a provider and can thus be operably connected to another HMS 204.

[0075] Also, HMS 204 can be implemented both as an XML web service and as a class which can be called directly if client 202 resides on the same server as HMS 204. The provider can also be either deployed remotely as XML web services or on the same server as HMS 204.

[0076] It can thus be seen that, in accordance with one embodiment of the present invention, the links are requested based on object types (or object classes) or specific object instances. The presentation of the nodes is decoupled from the nodes themselves. The nodes are instances of objects rather than presentation elements, such as web pages. This allows client 202 to process or handle the destination node of the link in any manner it wishes.

[0077] FIGS. 8-10 illustrate yet another embodiment of the present invention. In that embodiment, one of providers 208-212 provides links between associated objects based on association metadata. In other words, when developing an application, it is common for a set of objects (such as business objects or entities in a business application) to be defined. Such objects in a business application may include, for example, a “Customer” object, a “SalesPerson” object and an “Order” object. These objects (entities) are interrelated through different associations. For instance, an “Order” has both a “Customer” and a “SalesPerson” associated with it. Since these associations exist in the problem domain of the application, they are associations that the end user typically understands. Therefore, it may be beneficial to allow the end user to navigate between these associations.

[0078] The information that defines these associations is captured in a metamodel (or object model) of the applications as they are being developed. This information is typically stored as metadata. For example, FIG. 8 depicts a relationship between an “Order” entity and a “Customer” entity that is modeled during the application development process.

[0079] There are known tools which can be run against object models generated during development of an application. Such tools compile the models into association metadata. In accordance with an illustrative embodiment of the invention, this is done and the association metadata is stored.

[0080] FIG. 9 is a block diagram of an embodiment of hypermedia system 201 that takes advantage of the stored association metadata. A number of the items shown in FIG. 9 are similar to those shown in FIG. 3 and are similarly numbered. However, FIG. 9 shows that system 201 also includes a metadata hypermedia provider 400 which is connected to a metadata store 402.

[0081] The metadata associations developed during the application development process (such as the information shown in FIG. 8 which illustrates an association between an “Order” entity and a “Customer” entity) is stored in metadata store 402. FIG. 10 is a flow diagram illustrating the operation of system 201 in accordance with one embodiment of the present invention, and will be described in conjunction with FIG. 9.

[0082] It is assumed that metadata hypermedia provider 400 has properly registered with HMS 204 and its link and identification data resides in provider register 206. Client 202 first generates a hypermedia request (or link request) specifying which objects are the source of the links sought, and which categories of links are to be retrieved. This request is received by HMS 204. This is indicated by block 404 in FIG. 10. HMS 204 then forwards the request on to the appropriate providers, which in this case will include metadata hypermedia provider 400. This is indicated by block 406.

[0083] Provider 400 analyzes association information contained in metadata store 402. One illustrative design of metadata hypermedia provider 400 is discussed in greater detail in the Appendix hereto. Briefly, however, provider 400 examines each association in metadata store 402 which has
been requested and determines whether the user has rights to access the associated entities. Provider 400 can determine whether the user has rights to access the associated entities by accessing a security subsystem, or in any other suitable way. Accessing security does not form part of the present invention. Provider 400 then creates a link for each association for which the user has access, and places association information in the link. This is indicated by block 408.

[0084] Provider 400 identifies (using terminology defined by the Unified Modeling Language (UML)) simple associations and composition associations; these can have a variety of cardinalities, such as 1-1, 1-many or many-many relationships. Provider 400 also identifies inheritance associations. For each association located by provider 400, provider 400 creates a link between the source node and the associated node. This is indicated by block 410.

[0085] The links are returned from provider 400 to HMS 204 as indicated by block 412, and HMS 204 aggregates all returned links and forwards them on to client 202. This is indicated by block 414. In one embodiment, provider 400 does not return the associated node, but instead returns a query whose results, if executed, include only the associated node.

[0086] Although the present invention has been described with reference to particular embodiments, workers skilled in the art will recognize that changes may be made in form and detail without departing from the spirit and scope of the invention.
Appendix A

Hypermedia Subsystem Design

Pages 1-17
Hypermedia Subsystem Design

Subsystem Overview and Design

1. Introduction

Problem Statement

Users of applications built upon the .NET Business Platform need a means of performing navigation among those applications and to external applications and resources. To this end, navigational links that are pervasive throughout these applications and “at the user’s fingertips” rather than buried within menus are the most desirable.

To accomplish this, developers of applications built upon the .NET Business Platform need a means of providing navigational links. For example, an application developer may wish to provide navigation from a Business entity to the list of the Customer’s orders, to a Customer Maintenance form, or perhaps to an external source, such as Expedia, to provide a map of the Customer’s delivery address. Once such a navigational link has been provided, it should be made available wherever a Customer business entity is present, throughout all applications.

The .NET Explorer (project Iceberg) is an example of an application that could leverage such navigation, presenting links for business entities that appear in query results. Currently the Dynamics / eEnterprise Explorer, a product similar to the .NET Explorer, allows a similar type of navigation to Dynamics and eEnterprise windows via its Gotos. The AnyView Creator allows application developers to use VBScript or Dexterity to create custom Gotos for objects created in AnyView Creator. The navigational links provided via Gotos are limited in use however, as they only allow navigation to Dynamics and eEnterprise windows from the AnyView Browser. The solution provided by the Hypermedia Subsystem is more versatile in that navigational links provided for an object via the Hypermedia Subsystem shall be available for use within any application built upon the .NET Business Platform, and facilitate a much richer navigation than simply going to Dynamics and eEnterprise windows.

Document Scope

This document is an overview of the Hypermedia Subsystem and its design as a phase 1 deliverable of the Iceberg project. The requirements covered in the Requirements section of this document span multiple phases of the Iceberg project, however, the majority of this document only addresses the phase 1 requirements – the exception is the Subsystem Description, which contains a brief discussion of planned phase 2 features. Requirements for later phases are being considered only to the extent that they impact the phase 1 design.

This document considers only the Hypermedia Subsystem, and does not describe any UI for managing hypermedia or how clients of this subsystem will render hypermedia. Throughout this document are some examples of how hypermedia may be used; however, this document does not define a complete list. For more information on specific uses of the Hypermedia Subsystem, the dependent subassemblies’ design must be consulted.

Additional information and related, but external, areas is included in the Appendices of this document. These areas are Appendix 1: Hypermedia Provider Authoring, and Appendix 2: Discussion of Link Handler Architecture.

For more information about the Iceberg project see the Iceberg Software Requirements and the Glacier Website.

Subsystem Description

High Level Overview

The Hypermedia Subsystem is responsible for the aggregation of hypermedia within the .NET Business Platform. From the hypermedia client’s point of view, this consists primarily of fulfilling a client’s request for hypermedia. To do this the Hypermedia Subsystem retrieves and aggregates hypermedia from a set of providers, and then returns it to the client. From the hypermedia provider’s point of view, the Hypermedia Subsystem is where the provider is registered, so that it can provide hypermedia to hypermedia clients. The primary external interface to the Hypermedia Subsystem, for both hypermedia clients and hypermedia providers, is the hypermedia service.

In general, a hypermedia system consists of a set of nodes and the links that relate those nodes to one another. Nodes are grouped into node classes, which are sets of nodes that have the same properties and are related to other nodes in the same way. In the Hypermedia Subsystem implemented by the .NET Business Framework nodes are instances of .NET classes.
class to be the source node of a link it must be an Entity as defined by the .NET Business Framework. In this context, a node class is simply the .NET class.

For the remainder of this document, the terms class and instance will be used interchangeably with the terms node class and node.

**Nodes and Links**

Hypermedia is represented in the Hypermedia Subsystem as links. There are three types of links (class, instance, and instance specific), which represent relationships between nodes, as well as actions that can be performed on nodes.

Class links have the context of a class. They represent either a relationship to a destination node, or an action that can be performed. A class link provides an answer to the question "Where can I go, or what can I do that pertains to a particular class?" Note that in this question we only have the context of a class.

Instance and instance specific links have the context of an instance. The difference between the two types is that instance specific links are directly tied to a specific instance, and instance links are tied to a class, but an instance of that class must be specified in order to traverse the link. An instance or instance specific link provides an answer to the question "Where can I go, and what can I do, with this particular instance?" Note that in this question we have the context of an instance.

All three types of links can be traversed, which will return the destination node of the link. If the link represents an action, traversal also performs the action the link represents.

As stated above, there are three types of links: class, instance, and instance specific. The type of the link is not the only manner in which links may be grouped. Links also belong to a category. One example of a link category is the metamodel category. Links that belong to this link category represent associations between Entities. These associations are captured in the metamodel (object model) of the system. In addition to being a grouping mechanism, a link category also defines a protocol that is followed by the hypermedia provider that supplied the link. The link category (protocol) gives a client of the Hypermedia Subsystem an indication of what type of information the link represents, and what type of object will result from traversal of the link. All links of the same category can be handled in the same manner. In this way, the hypermedia client will be able to determine how to handle a link based on the link category to which it belongs.

**Providers and Registration**

All of the providers supplied by the Hypermedia Subsystem are actually supplied by hypermedia providers. A hypermedia provider is a class or web service that implements the IHypermediaProvider interface, which is defined in the Hypermedia Subsystem. The proper implementation of this interface allows the hypermedia provider to supply hypermedia via the hypermedia service. (See Appendix I: Hypermedia Provider Authoring for a discussion of implementing the interface.)

The link types, link categories, and node classes for which the hypermedia provider supplies hypermedia are communicated through the IHypermediaProvider interface. If one or more providers supplies hypermedia for a particular node class, the node class is said to be supported by the Hypermedia Subsystem. The IHypermediaProvider interface also provides methods for the retrieval and traversal of hypermedia.

In addition to implementing the IHypermediaProvider interface, the hypermedia subsystem must know where the provider exists. This is done through a registration process.

**Hypermedia Retrieval**

Clients of the Hypermedia Subsystem will be able to make a variety of requests for hypermedia. When they do so, they will be returned a set of links.

A hypermedia client may request class, instance, and instance specific links from the hypermedia service. When they make the request they specify which type(s) they wish to retrieve. If only class or instance links are being retrieved a ClassKey must be specified in the request. This ClassKey is used to specify the class for which links are being requested. If instance specific links are being requested an EntityKey must be specified in the request. This EntityKey is used to specify the instance for which links are being requested.

Hypermedia clients will also be able to limit the hypermedia returned by specifying what link categories they are interested in. Additional methods of limiting the hypermedia being returned may also be supported in the future.

**A Closer Look at Instance and Instance Specific Hypermedia**

Links which apply to instances fall into two groups: those that are applicable to (available for) all instances of a given class, and those that are applicable to (available for) only a subset of the instances of a given class. This is an important distinction, and it plays a role in the balance of speed versus simplicity.

In many cases, a link is applicable to all the instances of a given class. If this is the case, and a hypermedia client has a set of instances to request hypermedia for, it doesn't make sense for the hypermedia client to request this group of links for each instance. That is where the concept of instance links comes into play. The client would request instance links, specifying the
ClassKey that identifies which class the instances belong to, and only when the link is traversed does the client specify which instance to use.

In some cases, a link is applicable to only a subset of the instances of a given class. For example, if there were a Timecard entity, a workflow link such as an "approve timescard" link would only be available for a subset of the Timecard entities since the availability of the link depends on the state of the Timecard in workflow. This is where the client must request instance specific links. To make this request the client must specify the EntityKey of the instance they are requesting links for. No further context is needed to traverse the link.

.NET vs. Non-.NET Clients
It cannot be (and isn’t) assumed that all hypermedia clients will be .NET clients. Therefore, the Hypermedia Subsystem will support both .NET and non-.NET clients. This will be accomplished primarily through the use of XML.

All communication with the hypermedia service is done through XML documents. If the client does happen to be a .NET client, it will be able to convert the XML documents into instances of the .NET classes provided by the Hypermedia Subsystem.

Definitions

**class**
A link that represents either a relationship to a destination node, or an action that can be performed. A class link has the context of a particular node class.

**destination node**
A node at which traversal of a link terminates. Destination nodes are not required to belong to a supported node class.

**handle (a link)**
A term used to describe the process by which a hypermedia client performs some processing using the destination node of a link traversal. Typically this will include giving the end user some indication that the traversal took place. Handling a link is not part of traversing a link, but rather is a separate step, external to the Hypermedia Subsystem, which is performed by the client. For example, in the .NET Explorer, when the user navigates from one Entity to another Entity, say from a Customer to the Orders for that Customer, they do so via a (metadata) link. Traversing this link returns a query as the destination node. Handling the link is when the client executes that query and displays the results to the user.

**hypermedia**
A generic term to refer to the links available through the Hypermedia Subsystem.

**hypermedia service**
A class or web service that implements the IHypermediaProvider interface.

**hypermedia service**
This term is used to refer to the class and/or the web service which is the main interface to the Hypermedia Subsystem for both hypermedia providers and hypermedia clients.

**link**
A generic term used to refer to any type of link without specifying the type. The different types of links are class, instance, and instance specific. In the context of the Hypermedia Subsystem the term link does not refer to a URL.

**link category**
A property of a link which serves as a method of categorization of links and identifies a protocol that is followed by the hypermedia provider. The link category indicates what type of information the link represents, and what type of object will result from traversal of the link. All links of the same category can be handled in the same manner. Link categories are defined by hypermedia providers.

**link type**
The type of the link – class, instance, or instance specific.

**node**
A term used to describe both the source and destination of a link. Nodes are represented in the Hypermedia Subsystem by instances of .NET classes.

**instance link**
A link that represents either a relationship between a source node and a destination node, or an action that can be performed on a particular source node. An instance link has the context of an instance. The link itself maintains the context of a class, but an instance of that class must be specified to traverse the link.
instance specific link
A link that represents either a relationship between a source node and a destination node, or an action that can be performed on a particular source node. An instance specific link has the context of an instance, and maintains this context on the link itself.

node class
A set of nodes that have the same properties and are related to other nodes in the same way. Node classes are represented in the Hypermedia Subsystem by .NET classes. (i.e. There is a 1:1 correspondence between a node class and a .NET class).

source node
A node from which traversal of a link originates. Source nodes must belong to a supported node class. Source nodes must be Entities as defined by the .NET Business Framework.

supported node class
A node class for which there exists one or more registered hypermedia providers that supply links for it.

traversing
The process of navigating the relationship represented by the link and returning the destination node. For example, if the link represents the relationship between a Customer and the Customer’s Orders, traversing the link returns the query, not the results of executing the query. Likewise, if the link represents the relationship between a Customer and the URL of an Expedia page containing a map of the Customer’s address, traversing the link returns the URL – it does not open a browser window displaying the page pointed to by the URL. If the link represents an action, traversing the link performs the action.

unsupported node class
A node class for which there exists no registered hypermedia providers that supply links for it.

For other definitions, see the Bedrock Glossary.

References
AnyView Creator For Dynamics / eEnterprise 6.0 - Installation, Configuration, and User Documentation
Hypermedia Schema Documentation

2. Use Cases, Scenarios and Requirements

Actors

Hypermedia Client
A hypermedia client uses the Hypermedia Subsystem to retrieve and traverse hypermedia.

A client of the Hypermedia Subsystem may request hypermedia from the hypermedia service. Once a link is traversed, it is the client’s responsibility for determining how to handle the object resulting from traversal of the link.

Hypermedia Administrative Client
A hypermedia administrative client configures the Hypermedia Subsystem. This includes registering and unregistering hypermedia providers.

Hypermedia Provider
A hypermedia provider is any class or web service that implements the IHypermediaProvider interface (exposing the methods of this interface as WebMethods). For a hypermedia provider to supply hypermedia via the hypermedia service, it must be registered with the hypermedia service.

Hypermedia Client Use Cases
The Hypermedia Client Use Cases diagram depicts the primary use cases that the Hypermedia Subsystem is interested in realizing for the Hypermedia Client actor.

In the sequence diagrams and other diagrams used to realize these use cases, the IHypermediaProvider interface is used interchangeably with the Hypermedia Provider actor.
Get Class Links

Get Instance Links

Get Instance Specific Links

Traverse Link

Get List of Supported Node Classes

Get List of Categories

Register Hypermedia Provider

UnRegister Hypermedia Provider

Get Schema Location

get supported hypermedia information

demand task permission

demand task permission

Security Subsystem

Explorer Services
Use Case Descriptions

Get Class Links
Overview:
A client may wish to retrieve a set of class links. This use case describes how a hypermedia client requests class links.

Starting Point:
The client wishes to retrieve a set of class links.

Ending Point:
The requested links have been returned to the client.

Measurable Result:
The requested links have been returned to the client.

Flow of Events:
A request is made to retrieve all the class links that apply to a particular class. Any parameters of the request that would limit the hypermedia being returned, such as a list of link categories, are taken into consideration, and only hypermedia that matches the parameters is aggregated and returned. The hypermedia service determines which providers supply links to fulfill the request. It then forwards the request on to the appropriate providers. In the request, the hypermedia service indicates what version of provider information it had in its cache to determine that this provider should be called. The set of links returned by the providers is aggregated and returned to the client.

Alternate Flow of Events:

Get Instance Links
Overview:
A hypermedia client may wish to retrieve the instance links available for instances of a given class. In this use case only the links that are applicable to all instances of the specified class are returned. This use case describes how a hypermedia client requests instance links.

Starting Point:
The client wishes to retrieve a set of instance links available for nodes of a given class.

Ending Point:
The requested links have been returned to the client.

Measurable Result:
The requested links have been returned to the client.

Flow of Events:
A request is made to retrieve all the instance links that apply to all instances of a particular class. Any parameters of the request that would limit the hypermedia being returned, such as a list of link categories, are taken into consideration, and only hypermedia that matches the parameters is aggregated and returned. The hypermedia service determines which providers supply links to fulfill the request. It then forwards the request on to the appropriate providers. In the request, the hypermedia service indicates what version of provider information it had in its cache to determine that this provider should be called. The set of links returned by the providers is aggregated and returned to the client.

Alternate Flow of Events:

Get Instance Specific Links
Overview:
A hypermedia client may wish to retrieve the instance specific links available for a given instance. In this use case only the links that are specific to the instance specified in the request are returned. This use case describes how a hypermedia client requests instance specific links.

Starting Point:
The client wishes to retrieve a set of instance specific links for a given instance.

Ending Point:
The requested links have been returned to the client.

Measurable Result:
The requested links have been returned to the client.

Flow of Events:
A request is made to retrieve all the instance specific links that apply to a particular instance of a class. Any parameters of the request that would limit the hypermedia being returned, such as a list of link categories, are taken into consideration, and only hypermedia that matches the parameters is aggregated and returned. The hypermedia service determines which providers supply links to fulfill the request. It then forwards the request on to the appropriate providers. In the request, the hypermedia service indicates what version of provider information it had in its cache to determine that this provider should be called. The set of links returned by the providers is aggregated and returned to the client.

Alternate Flow of Events:

Get List of Categories
Overview:
When a client requests hypermedia, it can specify a list of link categories. If such a list is supplied, the hypermedia retrieved will be limited to links that belong to those link categories. This use case describes how a client retrieves a complete list of all link categories the hypermedia service can return.

Starting Point:
The client requests a list of all link categories from the hypermedia service.

Ending Point:
The list of all link categories has been returned to the client.

Measurable Result:
The list of all link categories has been returned to the client.

Flow of Events:
A client makes a request to retrieve a list of all the link categories the hypermedia service can return. The hypermedia service examines the information about each hypermedia provider that was cached when it was registered to determine the complete list of link categories. The list is returned to the client.

Alternate Flow of Events:

Outstanding Issues:

Get List of Supported Node Classes
Overview:
A hypermedia client may wish to retrieve a list of all the node classes supported by the Hypermedia Subsystem. With this information, the client would not have to make calls to the hypermedia service for unsupported node classes. This use case describes how a hypermedia client retrieves a list of all supported node classes from the hypermedia service.

Starting Point:
The client requests a list of all supported node classes.

Ending Point:
The list of all supported node classes has been returned to the client.

Measurable Result:
The list of all supported node classes has been returned to the client.

Flow of Events:
A client makes a request to retrieve a list of all supported node classes. The hypermedia service examines the information about each hypermedia provider that was cached when it was registered to determine the complete list of supported node classes. The list is returned to the client.

Alternate Flow of Events:

Get Schema Location
Overview:
The XML produced and consumed by the hypermedia service conforms to a schema. Clients of the hypermedia service may wish to retrieve or reference this schema. This use case describes how a hypermedia client retrieves the location of the schema used by the hypermedia subsystem.

Starting Point:
The client requests the location of the schema.

Ending Point:
The location of the schema has been returned to the client.
Measurable Result:
The location of the schema has been returned to the client.

Flow of Events:
A client makes a request to retrieve the location of the schema used by the hypermedia service. The hypermedia service determines the location of the schema and returns it to the client.

Alternate Flow of Events:

Register Hypermedia Provider
Overview:
A hypermedia provider must be registered with the hypermedia service. This allows hypermedia service to know that the provider exists, and that hypermedia can be requested from it. When a hypermedia provider is registered, information about it is cached by the hypermedia service. This use case describes how a hypermedia provider is registered with the hypermedia service.

Starting Point:
A hypermedia provider has been installed and requires registration with the hypermedia service.

Ending Point:
The hypermedia provider has been registered with the hypermedia service.

Measurable Result:
The hypermedia provider has been registered with the hypermedia service, and the hypermedia service has cached information about the link categories, link types, and node classes for which the provider supplies hypermedia.

Flow of Events:
A request is made to register a hypermedia provider with the hypermedia service. The hypermedia service first demands that the user making the request has the security rights to do so. The hypermedia service then stores information about the existence of the provider. The hypermedia service then requests information about the link categories, link types, and node classes for which the provider supplies hypermedia. This information is cached by the hypermedia service. Additionally, when this information is returned by the provider, it is accompanied by a version identifier; this information is cached as well.

Traverse Link
Overview:
When a hypermedia client has a link, they can traverse that link to obtain the destination node. This use case describes how a link traversal is performed.

Starting Point:
The client has a link that they wish to traverse.

Ending Point:
The link has been traversed.

Measurable Result:
The link has been traversed (i.e. the destination node has been returned to the client and/or the action the link represents has been performed).

Flow of Events:
A hypermedia client has a class link or an instance-specific link. The client requests that the hypermedia service traverse the link. The hypermedia service delegates the traversal of the link to the hypermedia provider that supplied it. The provider traverses the link, returning the destination node to the hypermedia service. The hypermedia service then returns the destination node to the client.

Alternate Flow of Events:
A hypermedia client has an instance link. The flow of events is the same as above with the exception that the client must specify the instance for which the traversal should take place in the traverse request.

UnRegister Hypermedia Provider
Overview:
If a hypermedia provider is uninstalled, or some other reason exists so that the provider should no longer be used, it must be unregistered with the hypermedia service. This use case describes how a hypermedia provider is unregistered with the hypermedia service.

Starting Point:
A hypermedia provider must be unregistered with the hypermedia service.

Ending Point:
The hypermedia provider is unregistered with the hypermedia service.

Measurable Result:
The hypermedia provider is no longer registered with the hypermedia service, and the information previously cached about the provider has been removed from the cache.

Flow of Events:
A request is made to unregister a hypermedia provider. The hypermedia service first demands that the user making the request has the security rights to do so. The hypermedia service then deletes any information it holds about the existence of the provider. Any information stored about the provider (including the link categories, link types and node classes for which it supplies hypermedia) is removed from the hypermedia service's cache of information.

Alternate Flow of Events:

Outstanding Issues:

**Hypermedia Provider Use Cases**

The Hypermedia Provider Use Cases diagram depicts the use cases that the Hypermedia Subsystem is interested in realizing for the Hypermedia Provider actor.

In the sequence diagrams and other diagrams used to realize these use cases, the IHypermediaProvider interface is used interchangeably with the Hypermedia Provider actor.

---

**Use Case Descriptions**

**Create Link**

Overview:
A hypermedia provider must be able to create links in order to fulfill requests for hypermedia. This use case describes how a hypermedia provider creates a link.

Starting Point:
A hypermedia provider has some data or domain knowledge from which it wishes to create a link.

Ending Point:
A link has been created.

Measurable Result:
A link has been created.

Flow of Events:
A hypermedia provider has some data or domain knowledge from which it can create a link to represent a relationship or action that can be performed. The provider calls the constructor on the Link class. The provider then sets the values of the appropriate properties on the link.

Alternate Flow of Events:

See Also:

3. Constraints (optional)
None documented at this time.

4. Decomposition

Hypermedia Service Architecture
The Hypermedia Service Architecture class diagram gives an overview of the classes that comprise the hypermedia service portion of the Hypermedia Subsystem.

This diagram does not show all the attributes and operations available on the classes shown. Only the attributes and operations that are architecturally significant or relative to the discussion of the subsystem from a high level are shown.

For a particular implementation of any of the interfaces included on this diagram, the class diagrams and other documentation of the implementing subsystem will need to be consulted.
Classes

HypermediaService
The HypermediaService is the class that implements all of the logic for the Hypermedia Subsystems realization of the IHypermediaService interface. Clients of the hypermedia service that exist on the same machine as the hypermedia service may use this class to retrieve hypermedia.

The HypermediaService provides a means for implementers of the IHypermediaProvider interface to be registered with the service. The HypermediaService maintains a list of all registered implementers of the IHypermediaProvider interface.

The HypermediaService is also responsible for fulfilling requests for hypermedia. When requests are made upon the HypermediaService to retrieve hypermedia the HypermediaService determines which providers can supply hypermedia to fulfill the request. It then retrieves the hypermedia from these providers and aggregates it. The aggregated collection of hypermedia is then returned to the client.

HypermediaWebService
The HypermediaWebService is the primary class used by remote clients of the Hypermedia Subsystem to retrieve hypermedia.

This class exposes the functionality of the HypermediaService class as a set of WebMethods.


IHypermediaProvider
IHypermediaProvider is the interface that must be implemented by all web services that wish to provide hypermedia via the Hypermedia Subsystem.

The implementer of the IHypermediaProvider interface must also be registered with the HypermediaService for its links to be made available via the HypermediaService.

IHypermediaService
IHypermediaService is the interface that is implemented by the HypermediaService and HypermediaWebService classes.
Request / Response Architecture

The Request / Response Architecture class diagram gives an overview of the classes that comprise the non-hypermedia service portion of the Hypermedia Subsystem. These are the common classes used by hypermedia clients, hypermedia providers, and the hypermedia service.

This diagram does not show all the attributes and operations available on the classes shown. Only the attributes and operations that are architecturally significant or relative to the discussion of the subsystem from a high level are shown.

For a particular implementation of any of the interfaces included on this diagram, the class diagrams and other documentation of the implementing subsystem will need to be consulted.

Classes

HypermediaRequest

A helper class used to create and parse the XML documents that represent a request to retrieve hypermedia.
HypermediaResponse
A helper class used to create and parse the XML documents that represent a response to a request to retrieve hypermedia.

IXmlSerializable
An interface used within the Hypermedia Subsystem which defines how to serialize an object to an XML document.

Link
A Link represents either a relationship to a destination node, or an action that can be performed.

Through inheritance from LinkBase, a Link has a LinkType associated with it. The LinkType determines to what level of granularity the link applies (i.e. does it apply to a specific node or to a node class).

A Link can be traversed.

ProviderInfoRequest
A helper class used to create and parse the XML documents that represent a request for information from a hypermedia provider.

ProviderInfoResponse
A helper class used to create and parse the XML documents that represent a response to a request for information from a hypermedia provider.

Response
The base class for all helper classes used to create and parse the XML documents that represent a response from the hypermedia service. This class itself is also used to create and parse certain XML documents that represent a response from the hypermedia service.

SerializationBase
The abstract base class for all objects that can be serialized to and deserialized from XML documents.

TraverseRequest
A helper class used to create and parse the XML documents that represent a request to traverse a link.

TraverseResponse
A helper class used to create and parse the XML documents that represent a response to a request to traverse a link.

5. Responsibility Mapping

Use Case Realizations
None documented at this time.
6. Subsystem Interface

The following diagram shows the IHypermediaProvider interface which hypermedia providers must implement to supply hypermedia through the hypermedia service.

```
<<Interface>>
 IHypermediaProvider

• GetHypermedia(hypermediaRequest: XElement): XElement
• GetHypermediaLink(hypermediaLinkRequest: XElement): XElement
• TraverseLink(link: XElement): XElement
```

The following diagram shows the IHypermediaService interface. This is the interface through which clients interact with the hypermedia subsystem.

```
<class>
 IHypermediaProvider

(from Hypermedia)

<<Interface>>
 IHypermediaService

• GetHypermediaSchemaLocation(): string
• RegisterHypermediaProvider(providerUri: string): XElement
• UnregisterHypermediaProvider(providerUri: string): XElement
```

Appendix 1: Hypermedia Provider Authoring

Overview

Hypermedia providers are responsible for a number of things, and must be authored correctly for the links they provide to be of use to clients of the Hypermedia Subsystem. This appendix is intended to give an overview of what must be done to author a hypermedia provider. It also teaches for how to implement a hypermedia provider.
Example Implementation

First we will look at one method of implementing a hypermedia provider. Consider the following diagram.

In this diagram there are two classes shown which belong to the implementation of the hypermedia provider. They are the HypermediaProviderExample and the HypermediaProviderExampleWebService classes. As indicated by the notes on the diagram, the HypermediaProviderExample class contains all of the logic needed to realize the IHypermediaProvider interface. The HypermediaProviderExampleWebService class simply wraps calls to HypermediaProviderExample, exposing them as WebMethods.

If the provider is implemented in this manner, then it can be registered with the hypermedia service in two ways. First, if the provider is on the same server as the hypermedia service, the assembly which contains its implementation (in this case the HypermediaProviderExample class) may be registered with the hypermedia service. Second, the URL of the web service (in this case the web service implemented as HypermediaProviderExampleWebService) may be registered with the hypermedia service.

An additional benefit to this style of implementation is that the logic contained in the HypermediaProviderExample can be tested without having to go through the web service layer. This should help to simplify and categorize test cases into two groups. Testing the logic (the HypermediaProviderExample class), and testing the web service (the HypermediaProviderExampleWebService class).

Example Code

```java
public class HypermediaProviderExample : IHypermediaProvider {
    public HypermediaProviderExample() {
        // TODO: Initialize the provider.
    }

    public XmlDocument GetHypermedia(XmlElement hypermediaRequest) {
        // TODO: Return the requested hypermedia.
        * * * The request will be an XmlElement with of type
        * * * <hypermediaRequest>.

        * * * The return value should be an XmlElement of type
        * * * <hypermediaResponse>.
        */
```
public XmlElement GetProviderInfo(XmlElement providerInfoRequest) {
    /* TODO: Return the requested information about the provider. */
    /* the request will be an XmlElement of type 
    * <providerInfoRequest>. 
    * The return value should be an XmlElement of type 
    * <providerInfoResponse>. */
    return null;
}

public XmlElement TraverseLink(XmlElement link) { 
    /* TODO: Traverse the link, and return a traverse response. */
    /* the request will be an XmlElement of type 
    * <traverseRequest>. 
    * The return value should be an XmlElement of type 
    * <traverseResponse>. */
    return null;
}

public class HypermediaProviderExampleWebService : WebService, IHypermediaProvider
{
    private HypermediaProviderExample hypermediaProviderExample;
    public HypermediaProviderExampleWebService() {
        hypermediaProviderExample = new HypermediaProviderExample();
    }
    [WebMethod]
    public XmlElement GetHypermedia(XmlElement hypermediaRequest) {
        return hypermediaProviderExample.GetHypermedia(hypermediaRequest);
    }

    [WebMethod]
    public XmlElement GetProviderInfo(XmlElement providerInfoRequest) {
        return hypermediaProviderExample.GetProviderInfo(providerInfoRequest);
    }

    [WebMethod]
    public XmlElement TraverseLink(XmlElement link) {
        return hypermediaProviderExample.TraverseLink(link);
    }
}

Appendix 2: Discussion of Link Handler Architecture

Link handlers are the code that takes the destination node of a link and performs some operation on it. For example, taking a URL destination node and opening the URL in a browser window, or executing a query destination node and displaying the results in a result viewer. Link Handlers are not discussed earlier in this document because they are external to the Hypermedia Subsystem. This appendix is a discussion of a possible architecture for a handler subsystem.

For a given environment each link category should be handled in the same manner. For example, on an IES5 client all links that are of the same category, say Microsoft.BusinessFramework.Metamodel, should be handled in the same manner. However, if a client does need to handle a link differently due to some special requirements, they should be able to use a different handler when necessary.

To facilitate this type of behavior, link handlers should describe which link categories they handle and for which environments. The handler subsystem should know about all the available handlers (either through some registration or discovery scheme). The handler subsystem should also facilitate identifying a "default" link handler for a given environment and link category. A
client would then be able to use the handler subsystem to discover the proper "default" link handler to use for a given link. The client should also be able to identify a specific handler to use when handling a link.

Appendix 3: Link Category Naming Convention

Requirement 3.3.3 states that a hypermedia provider may provide a category of links that has not previously been defined. This means that a provider, potentially written by some third party, has to be able to define a new link category. There is the possibility of two different third parties each creating a new link category with different behavior, but with the same name.

This poses a problem, since a link category is defined to be a property of a link which identifies a protocol that is followed by the hypermedia provider so that all links of the same category can be handled in the same manner.

Thus the following naming convention has been defined for link categories.

The name of a link category should adhere to the following pattern:

\[ \text{CompanyName}.\text{TechnologyName}.\text{CategoryName} \]

An example of this would be the link category defined by the business framework which represents associations in the metamodel. This link category is named \text{Microsoft.BizTalk.BizTalkMetamodel}.

\text{CompanyName} scopes the link category to the company which has ownership of defining the protocol that hypermedia providers that supply links of this category must follow.

\text{TechnologyName} scopes the link category to the technology to which it pertains.

\text{CategoryName} is the unique name of the link category. This name must be unique among all other link categories with the same \text{CompanyName}.\text{TechnologyName}.
Appendix B

Schema Hypermedia.xsd

Pages 1-34
Schema Hypermedia.xsd

Elements
- hypermediaRequest
- hypermediaResponse
- link
- providerInfoRequest
- providerInfoResponse
- response
- traverseRequest
- traverseResponse

Complex types
- DestinationNode
- DiagnosticInfo
- ExceptionInfo
- ExceptionInfoCollection
- HypermediaRequest
- HypermediaResponse

Simple types
- LinkCategory
- LinkType
- ResponseCode
- SerializationType

Dynamic ViewId

Element hypermediaRequest

Diagram

Type: HypermediaRequest
Children: linkCategories, key, dynamicViewId, linkTypes
Attributes: Name, Type
source  <xsd element name="hypermediaRequest" type="HypermediaRequest"/>

element hypermediaResponse
diagram

type  HypermediaResponse
children  responseCode diagnosticInfo classLinks instanceLinks instanceSpecificLinks
source  <xsd element name="hypermediaResponse" type="HypermediaResponse"/>
<xsd element name="link" type="Link">
element `providerInfoRequest`

diagram

```
providerInfoRequest  getSupportedHypermediaInfo
```

- The `getSupportedHypermediaInfo` element is used to specify that data about the hypermedia a provider supports should be included in the ProviderInfo element returned as part of the return.
- From the following method:

type `ProviderInfoRequest`
children `getSupportedHypermediaInfo`
source `<xsd:element name="providerInfoRequest" type="ProviderInfoRequest"/>
```

element `providerInfoResponse`

diagram

```
providerInfoResponse  responseCode  diagnosticInfo  supportedNodeClasses
```

- The `responseCode` element is set when the operation succeeds.
- The `diagnosticInfo` element is set when the operation fails.
- The `supportedNodeClasses` element is set when the operation succeeds and hypermedia support is not available.

type `ProviderInfoResponse`
children `responseCode  diagnosticInfo  supportedNodeClasses`
attributes
- Name
- Type
- Use
- Default
- Fixed
- version
- xsd:string
- required

source `<xsd:element name="providerInfoResponse" type="ProviderInfoResponse"/>
```
element response
diagram

response

type Response
children responseCode diagnosticInfo
source <xsd:element name="response" type="Response"/>

element traverseRequest
diagram

traverseRequest

type TraverseRequest
children link key
source <xsd:element name="traverseRequest" type="TraverseRequest"/>

element traverseResponse
diagram

traverseResponse

type TraverseResponse
children responseCode diagnosticInfo destinationNode displayMessage
attributes Name Type Use Default Fixed
source <xsd:complexType name="DestinationNode">
  <xsd:sequence>
    <xsd:any processContents="skip"/>
  </xsd:sequence>
  <xsd:attribute name="type" type="xsd:string" use="required"/>
</xsd:complexType>

complexType DiagnosticInfo

diagram


children exceptions

used by Response/diagnosticInfo

source <xsd:complexType name="DiagnosticInfo">
  <xsd:sequence>
    <xsd:element name="exceptions" type="ExceptionInfoCollection" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
The ExceptionInfo complex type is used to hold an XML representation of the data contained in an instance of the System.Exception class.

```
<xs:complexType name="ExceptionInfo">
  <xs:attribute name="Name" type="xs:string" use="required"/>
  <xs:attribute name="Type" type="xs:string" use="required"/>
  <xs:attribute name="Use" type="xs:string" use="required"/>
  <xs:attribute name="Default" type="xs:string" use="required"/>
  <xs:attribute name="Fixed" type="xs:string" use="required"/>
  <xs:attribute name="message" type="xs:string" use="required"/>
  <xs:attribute name="source" type="xs:string" use="required"/>
  <xs:attribute name="stackTrace" type="xs:string" use="required"/>
  <xs:attribute name="innerException" type="xs:string" use="required"/>
  <xs:attribute name="provider" type="xs:anyURI" use="required"/>
</xs:complexType>
```
The ExceptionInfo complex type is used to hold an XML representation of the data contained in an instance of the System.Exception class.

```
<xsd:complexType name="ExceptionInfo">
  <xsd:sequence>
    <xsd:element name="message" type="xsd:string"/>
    <xsd:element name="source" type="xsd:string"/>
    <xsd:element name="stackTrace" type="xsd:string"/>
    <xsd:element name="innerException" type="ExceptionInfo" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

**Diagram:**
- **ExceptionInfo:message**
  - Diagram: `message`
  - Type: xsd:string
  - Source: `<xsd:element name="message" type="xsd:string"/>

- **ExceptionInfo:source**
  - Diagram: `source`
  - Type: xsd:string
  - Source: `<xsd:element name="source" type="xsd:string"/>

- **ExceptionInfo:stackTrace**
  - Diagram: `stackTrace`
  - Type: xsd:string
  - Source: `<xsd:element name="stackTrace" type="xsd:string"/>

- **ExceptionInfo:innerException**
  - Diagram: `innerException`
  - Source: `<xsd:element name="innerException" type="ExceptionInfo"/>`
type ExceptionInfo
children message source stackTrace innerException
attributes Name Type Use Default Fixed
exceptionType xsd:string required
provider xsd:anyURI

source <xsd element name="exception" type="ExceptionInfo" minOccurs="0" maxOccurs="unbounded"/>

complexType HypermediaRequest
diagram

type extension of SerializationBase
children linkCategories key dynamicViewId linkTypes
generated by hypermediaRequest

attributes Name Type Use Default Fixed
providerInfoVersion xsd:string
The LinkCategories element is used to restrict the hypermedia returned to hypermedia belonging to the specified link categories.

```
<xsdelement name="linkCategories" type="LinkCategoryCollection" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>The LinkCategories element is used to restrict the hypermedia returned to hypermedia belonging to the specified link categories.</xsd:documentation>
  </xsd:annotation>
</xsdelement>
```

```
<xsdelement name="key" type="String">
  <xsd:annotation>
    <xsd:documentation></xsd:documentation>
  </xsd:annotation>
</xsdelement>
```
type Key
source <xsd element name="key" type="Key"/>

element HypermediaRequest/dynamicViewld
diagram dynamicViewld
type xsd:string
source <xsd element name="dynamicViewld" type="xsd string"/>

element HypermediaRequest/linkTypes
diagram linkTypes
children linkType
source <xsd element name="linkTypes">
<xsd complexType>
<xsd:sequence>
<xsd:element name="linkType" type="LinkType" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd element>
complexType HypermediaResponse

diagram

type extension of Response
children responseCode diagnosticInfo classLinks instanceLinks instanceSpecificLinks
used by hypermediaResponse

element HypermediaResponse/classLinks
diagram

type LinkCollection
children link
source <xsd element name="classLinks" type="LinkCollection" minOccurs="0"/>
element HypermediaResponse/instanceLinks

diagram

type LinkCollection
children link
source `<xsd:element name="instanceLinks" type="LinkCollection" minOccurs="0"/>

element HypermediaResponse/instanceSpecificLinks

diagram

type LinkCollection
children link
source `<xsd:element name="instanceSpecificLinks" type="LinkCollection" minOccurs="0"/>

complexType IKey

diagram

used by Link?key TraverseRequest?key HypermediaRequest?key

source `<xsd:complexType name="IKey">
  `<xsd:sequence>
    `<xsd:element name="any" processContents="skip"/>
  `<xsd:sequence>
`
The Link complex type is used to hold an XML representation of the data contained in an instance of the Microsoft.BusinessFramework.Hypermedia Link class.
source <xsd:complexType name="Link">
  <xsd:annotation>
    <xsd:documentation>The Link complex type is used to hold an XML representation of the data contained in an instance of the Microsoft Business Framework Hypermedia Link class.</xsd:documentation>
  </xsd:annotation>
  <xsd:extension base="SchemaLinkBase">
    <xsd:sequence>
      <xsd:element name="linkType" type="LinkType"/>
    </xsd:sequence>
  </xsd:complexType>
</source>

element Link/displayName
diagram
<displayName/>
type xsd:string
source <xsd:element name="displayName" type="xsd:string"/>

element Link/linkType
diagram
<linkType/>
type LinkType
facets Class
  <enumeration>
    <enumeration>
      <enumeration>
        <enumeration>
          <enumeration>Instance</enumeration>
        </enumeration>
      </enumeration>
    </enumeration>
  </enumeration>
source <xsd:element name="linkType" type="LinkType"/>
element Link/key
diagram

type IKey
source <xsd element name="IKey" type="IKey"/>

element Link/associatedProperties
diagram

type PropertyIdentifierList
children propertyIdentifier
source <xsd element name="associatedProperties" type="PropertyIdentifierList" minOccurs="0"/>

element Link/providerData
diagram

source <xsd element name="providerData" minOccurs="0">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="propertyIdentifier" type="propertyIdentifier"/>
    </xsd:sequence>
    <xsd:sequence>
      <xsd:element name="PropertyIdentifierList" type="PropertyIdentifierList"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
complexType LinkCategoryCollection

documentation

The LinkCategoryCollection complex type is used to hold an XML representation of a collection of link category names.

source
<s:complexType name="LinkCategoryCollection">
  <s:documentation>The LinkCategoryCollection complex type is used to hold an XML representation of a collection of link category names.</s:documentation>
  <s:sequence>
    <s:element name="linkCategory" type="LinkCategory" minOccurs="0" maxOccurs="unbounded"/>
  </s:sequence>
</s:complexType>

element LinkCategoryCollection/linkCategory

documentation

source
<s:element name="linkCategory" type="LinkCategory" minOccurs="0" maxOccurs="unbounded"/>

complexType LinkCollection
documentation

The LinkCollection complex type is used to hold an XML representation of the data contained in an instance of the Microsoft Business Framework Hypermedia LinkCollection class.
The LinkCollection complex type is used to hold an XML representation of the data contained in an instance of the Microsoft Business Framework Hypermedia LinkCollection class.

```
<xs:complexType name="LinkCollection">
  <xs:annotation>
    <xs:documentation>The LinkCollection complex type is used to hold an XML representation of the data contained in an instance of the Microsoft Business Framework Hypermedia LinkCollection class.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="link" type="Link" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

```
type Link
children displayName linkType linkCategory linkSource sourceNodeClass destinationNodeClass key associatedProperties providerData
attributes Name Type Use Default Fixed
isAction xsd:boolean
```
required

isTraversable
xsd:boolean
required

id
xsd ID
optional

source <xsd:element name="link" type="Link" minOccurs="0" maxOccurs="unbounded"> complexType NodeClass
diagram

The NodeClass complex type is used to hold an XML representation of the System.Type.FullName property. It is used in the Microsoft.BusinessFramework.Hypermedia namespace to identify either the source node class or destination node class of a piece of hypermedia.


used by Link|destinationNodeClass Link|sourceNodeClass
documents

Name
Type
Use
Default
Fixed

name
xsd:string
required

The NodeClass complex type is used to hold an XML representation of the value the System.Type.FullName property. It is used in the Microsoft.BusinessFramework.Hypermedia namespace to identify either the source node class or destination node class of a piece of hypermedia.

source <xsd:complexType name="NodeClass">
  <xsd:documentation>The NodeClass complex type is used to hold an XML representation of the value the System.Type.FullName property. It is used in the Microsoft.BusinessFramework.Hypermedia namespace to identify either the source node class or destination node class of a piece of hypermedia.</xsd:documentation>
</xsd:complexType>
The PropertyIdentifier complex type is used to hold an XML representation of the data contained in an instance of the Microsoft BusinessFramework Hypermedia PropertyIdentifier class.
complexType PropertyldentifierList
diagram

class children property identifier
generated by Link associated Properties
annotation The PropertyldentifierList complex type is used to hold an XML representation of an instance of the Microsoft Business Framework Hypermedia PropertyldentifierList class that contains only Microsoft Business Framework Hypermedia Propertyldentifier objects.

source <xsd:complexType name="PropertyldentifierList">
    
    <xsd:annotation>
    <xsd:documentation>The PropertyldentifierList complex type is used to hold an XML representation of an instance of the Microsoft Business Framework Hypermedia PropertyldentifierList class that contains only Microsoft Business Framework Hypermedia Propertyldentifier objects.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
    <xsd:element name="propertyldentifier" type="Propertyldentifier" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    </xsd:complexType>

element PropertyldentifierList/propertyldentifier
diagram

type Propertyldentifier
attributes Name
Type
Use
Default
Fixed
nodeClass Name
xsd string
required

name
complexType ProviderInfoRequest

description

The ProviderInfoRequest complex type is used as input to the following methods:

type extension of SerializationBase
children getSupportedHypermediaInfo
used by providerInfoRequest

annotation

The ProviderInfoRequest complex type is used as input to the following method:

source

<xsd:complexType name="ProviderInfoRequest">
  <xsd:documentation>The ProviderInfoRequest complex type is used as input to the following method:
  <xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="SerializationBase">
      </xsd:extension>
    </xsd:complexContent>
    <xsd:element name="getSupportedHypermediaInfo" type="xsd:boolean">
      <xsd:annotation>The GetSupportedHypermediaInfo element is used to specify that data about the
      hypermedia a provider supports should be included in the ProviderInfo element returned as part of the return
      </xsd:annotation>
    </xsd:element>
  </xsd:annotation>
</xsd:complexType>
element ProviderInfoRequest/getSupportedHypermediaInfo
diagram
getSupportedHypermediaInfo
The `getSupportedHypermediaInfo` element is used to specify that data about the hypermedia a provider supports should be included in the ProviderInfo element returned as part of the return from the following method:

type xsd:boolean
annotation documentation
The `getSupportedHypermediaInfo` element is used to specify that data about the hypermedia a provider supports should be included in the ProviderInfo element returned as part of the return from the following method:

source
<xsd:element name="getSupportedHypermediaInfo" type="xsd:boolean"/>
<xsd:annotation>
<xsd:documentation>The `getSupportedHypermediaInfo` element is used to specify that data about the hypermedia a provider supports should be included in the ProviderInfo element returned as part of the return from the following method:
</xsd:annotation>
</xsd:element>

complexType ProviderInfoResponse
diagram
ProviderInfoResponse
diagnosticInfo
type extension of Response
children responseCode diagnosticInfo supportedNodeClasses
used by providerInfoResponse
attributes Name Type Use Default Fixed

version xsd:string
required

25
The ProviderInfo complex type is contained in a HypermediaResult element in the return from the following method:

Source: `<xsd:complexType name="ProviderInfoResponse">
  <xsd:annotation>
    <xsd:documentation>The ProviderInfo complex type is contained in a HypermediaResult element in the return from the following method: Microsoft BusinessFramework Hypermedia IHypermediaProvider.GetProviderInfo.</xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="Response">
      <xsd:sequence>
        <xsd:element name="supportedNodeClasses" type="SupportedNodeClassCollection" minOccurs="0"/>
        <xsd:attribute name="version" type="xsd:string" use="required"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>`

Element `ProviderInfoResponse/supportedNodeClasses`

Diagram: [Diagram of supportedNodeClasses]


Type: `SupportedNodeClassCollection`

Children: `supportedNodeClass`

Source: `<xsd:element name="supportedNodeClasses" type="SupportedNodeClassCollection" minOccurs="0"/>`

ComplexType `Response`

Diagram: [Diagram of Response]


Type: `SerializationBase`

Children: `responseCode`, `diagnosticInfo`

Used by: `response`

Source: `<xsd:complexType name="Response">
  <xsd:complexContent>
    <xsd:extension base="SerializationBase">
      <xsd:element name="responseCode"/>
      <xsd:element name="diagnosticInfo"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>`

26
element Response/responseCode

diagram

type ResponseCode

SUCCESS
SUCCESSWITHINFO
UNKNOWNFAILURE
INVALIDPROVIDERINFOVERSION

source <xsd:element name="responseCode" type="ResponseCode"/>

element Response/diagnosticInfo

diagram

type DiagnosticInfo

exceptions

source <xsd:element name="diagnosticInfo" type="DiagnosticInfo" minOccurs="0"/>

complexType SerializationBase

diagram


use by HypermediaRequest Link ProviderInfoRequest Response TraverseRequest

source <xsd:complexType name="SerializationBase"/>
complexType SupportedNodeClass

diagram


attributes

  name xsd:string required

linkCategory LinkCategory required

source

<xs:complexType name="SupportedNodeClass">
  <xs:attribute name="name" type="xsd:string" use="required"/>
  <xs:attribute name="linkCategory" type="LinkCategory" use="required"/>
  <xs:attribute name="linkType" type="LinkType" use="required"/>
</xs:complexType>

complexType SupportedNodeClassCollection

diagram


children supportedNodeClass

source

<xs:complexType name="SupportedNodeClassCollection">
  <xs:sequence>
    <xs:element name="supportedNodeClass" type="SupportedNodeClass" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
element SupportedNodeClassCollection/supportedNodeClass


type SupportedNodeClass

attributes
Name
Type
Use
Default
Fixed

name
xsd.string
required

linkCategory
LinkCategory
required

linkType
LinkType
required

source 
<xsd:element name="supportedNodeClass" type="SupportedNodeClass" minOccurs="0" maxOccurs="unbounded"/>

complexType TraverseRequest
diagram


type extension of SerializationBase

children link key

used by traverseRequest

source 
<xsd:complexType name="TraverseRequest">
  <xsd:complexContent>
    <xsd:extension base="SerializationBase">
      <xsd:sequence>
        <xsd:element name="link" type="Link"/>
        <xsd:element name="key" type="IKey" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
element TraverseRequest/link

displayName
linkType
linkCategory
linkSource
sourceNodeClass
destinationNodeClass
key
associatedProperties
providerData

type Link
children displayName linkType linkCategory linkSource sourceNodeClass destinationNodeClass key associatedProperties providerData
attributes Name Type Use Default Fixed
IsAction xsd:boolean required
IsTraversable xsd:boolean required
id xsd:ID optional

source <xsd element name="link" type="Link"/>
element TraverseRequest/key

type Key
source <xs:element name="key" type="Key" minOccurs="0"/>

complexType TraverseResponse
diagram

type extension of Response
children responseCode diagnosticInfo destinationNode displayMessage
used by traverseResponse

attributes Name Type Use Default Fixed serializationType SerializationType required

source <xsd:complexType name="TraverseResponse">
<xsd:complexContent>
<xsd:extension base="Response">
<xsd:sequence>
<xsd:element name="destinationNode" type="DestinationNode" minOccurs="0"/>
<xsd:element name="diagnosticInfo" type="DiagnosticInfo"/>
<xsd:element name="displayMessage" type="xsd:string"/>
<xsd:sequence>
<xsd:attribute name="serializationType" type="SerializationType" use="required"/>
</xsd:extension>
</xsd:complexType>
element TraverseResponse/destinationNode
diagram

typeDestinationNode
attributes Name Type Use Default Fixed
type xsd:string required

source <xsd element name="destinationNode" type="DestinationNode" minOccurs="0"/>

element TraverseResponse/displayMessage
diagram

type xsd:string

source <xsd element name="displayMessage" type="xsd:string"/>

simpleType LinkCategory
type restriction of xsd:string
used by Link/linkCategory LinkCategoryCollection/linkCategory SupportedNodeClass@linkCategory

facets

source <xsd simpleType name="LinkCategory">
<xsd restriction base="xsd:string">
<xsd minLength value="0"/>
</xsd restriction>
</xsd simpleType>
simpleType **LinkType**

type restriction of xsd:string
used by **Response/ResponseCode**

**SupportedNodeClass/GlinkType**

facets
- Class
- Instance
- InstanceSpecific

annotation The LinkTypeName simple type is used to hold an XML representation of a value from the Microsoft Business Framework Hypermedia LinkType enum.

source

```xml
<xsd:simpleType name="LinkType">  
  <xsd:annotation>  
    <xsd:documentation>The LinkTypeName simple type is used to hold an XML representation of a value from the Microsoft Business Framework Hypermedia LinkType enum.</xsd:documentation>  
  </xsd:annotation>  
  <xsd:restriction base="xsd:string">  
    <xsd:enumeration value="Class"/>  
    <xsd:enumeration value="Instance"/>  
    <xsd:enumeration value="InstanceSpecific"/>  
  </xsd:restriction>
</xsd:simpleType>
```

simpleType **ResponseCode**

type restriction of xsd:string

facets
- Success
- SuccessWithinInfo
- UnknownFailure
- InvalidProviderInfoVersion
source  <xsd:simpleType name="ResponseCode">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Success"/>
        <xsd:enumeration value="SuccessWithInfo"/>
        <xsd:enumeration value="UnknownFailure"/>
        <xsd:enumeration value="InvalidProviderInfoVersion"/>
    </xsd:restriction>
</xsd:simpleType>

simpleType SerializationType
type restriction of xsd:string
used by TraverseResponse@serializationType

faces SoapFormatter

TypeConverter

source  <xsd:simpleType name="SerializationType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="SoapFormatter"/>
        <xsd:enumeration value="TypeConverter"/>
    </xsd:restriction>
</xsd:simpleType>

XML Schema documentation generated with XML Spy Schema Editor www.xmlspy.com
Appendix C
Detailed Class Report
Pages 1-7
Detailed Class Report

Model name: HypermediaService
Root file: C:\depot\bedrock3\nbf\HypermediaService\HypermediaService.mdl

Package Microsoft
  HypermediaService: Microsoft
  Stereotype: namespace

Package BusinessFramework
  HypermediaService: Microsoft::BusinessFramework
  Stereotype: namespace

Package Hypermedia
  HypermediaService: Microsoft::BusinessFramework::Hypermedia
  Stereotype: namespace

Package Service
  HypermediaService: Microsoft::BusinessFramework::Hypermedia::Service
  Stereotype: namespace

Class HypermediaService
  HypermediaService: Microsoft::BusinessFramework::Hypermedia::Service::HypermediaService
  Stereotype:
  Visibility: 0 - PUBLIC
  Multiplicity:
  Persistence: 0 - PERSISTENT
  Documentation: Implements the <see cref="HypermediaService"/> interface.

Operation GetHypermedia
  Visibility: 0 - PUBLIC

  Parameter XmlElement
    Kind: 3 - RETURN
    TypeExpression: XmlElement
    DefaultValueExpression:

  Parameter hypermediaRequest
    Kind: 0 - IN
    TypeExpression: XmlElement
    DefaultValueExpression:

Operation GetHypermediaSchemaLocation
  Visibility: 0 - PUBLIC
Parameter string
  Kind: 3 - RETURN
  TypeExpression: string
  DefaultValueExpression:

Operation GetProviderInfo
  Visibility: 0 - PUBLIC

Parameter XmlElement
  Kind: 3 - RETURN
  TypeExpression: XmlElement
  DefaultValueExpression:

Parameter providerInfoRequest
  Kind: 0 - IN
  TypeExpression: XmlElement
  DefaultValueExpression:

Operation RegisterHypermediaProvider
  Visibility: 0 - PUBLIC

Parameter XmlElement
  Kind: 3 - RETURN
  TypeExpression: XmlElement
  DefaultValueExpression:

Parameter providerUri
  Kind: 0 - IN
  TypeExpression: string
  DefaultValueExpression:

Operation TraverseLink
  Visibility: 0 - PUBLIC

Parameter XmlElement
  Kind: 3 - RETURN
  TypeExpression: XmlElement
  DefaultValueExpression:

Parameter traverseRequest
  Kind: 0 - IN
  TypeExpression: XmlElement
  DefaultValueExpression:

Operation UnregisterHypermediaProvider
  Visibility: 0 - PUBLIC

Parameter XmlElement
Class HypermediaWebService

HypermediaService::Microsoft::BusinessFramework::Hypermedia::Service::HypermediaWebService
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Implements the <see cref="HypermediaService"/> interface to expose the functionality of the <see cref="HypermediaService"/> class as a <see cref="WebService"/>.

Operation GetHypermedia

Visibility: 0 - PUBLIC

Parameter XmlElement
Kind: 3 - RETURN
TypeExpression: XmlElement
DefaultValueExpression:

Parameter hypermediaRequest
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

Operation GetHypermediaSchemaLocation

Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

Operation GetProviderInfo

Visibility: 0 - PUBLIC

Parameter XmlElement
Kind: 3 - RETURN
TypeExpression: XmlElement
DefaultValueExpression:
Parameter providerInfoRequest
   Kind: 0 - IN
   TypeExpression: XmlElement
   DefaultValueExpression:

Operation GetRegisteredHypermediaProviders
   Visibility: 0 - PUBLIC

Parameter XmlElement
   Kind: 3 - RETURN
   TypeExpression: XmlElement
   DefaultValueExpression:

Operation HypermediaWebService
   Visibility: 0 - PUBLIC

Operation RegisterHypermediaProvider
   Visibility: 0 - PUBLIC

Parameter XmlElement
   Kind: 3 - RETURN
   TypeExpression: XmlElement
   DefaultValueExpression:

Parameter providerUri
   Kind: 0 - IN
   TypeExpression: string
   DefaultValueExpression:

Operation TraverseLink
   Visibility: 0 - PUBLIC

Parameter XmlElement
   Kind: 3 - RETURN
   TypeExpression: XmlElement
   DefaultValueExpression:

Parameter traverseRequest
   Kind: 0 - IN
   TypeExpression: XmlElement
   DefaultValueExpression:

Operation UnregisterHypermediaProvider
   Visibility: 0 - PUBLIC

Parameter XmlElement
   Kind: 3 - RETURN
   TypeExpression: XmlElement
Interface IHypermediaService

HypermediaService::Microsoft::BusinessFramework::Hypermedia::Service::IHypermediaService
Sterotype:  
Visibility: 0 - PUBLIC  
Multiplicity:  
Persistence: 0 - PERSISTENT

Documentation: Specifies the methods that need to be implemented by the hypermedia service.

Operation GetHypermediaSchemaLocation
Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN  
TypeExpression: string  
DefaultValueExpression:

Operation GetRegisteredHypermediaProviders
Visibility: 0 - PUBLIC

Parameter XmlElement
Kind: 3 - RETURN  
TypeExpression: XmlElement  
DefaultValueExpression:

Operation RegisterHypermediaProvider
Visibility: 0 - PUBLIC

Parameter XmlElement
Kind: 3 - RETURN  
TypeExpression: XmlElement  
DefaultValueExpression:

Parameter providerUri
Kind: 0 - IN  
TypeExpression: string  
DefaultValueExpression:

Operation UnregisterHypermediaProvider
Visibility: 0 - PUBLIC

**Parameter** XmlElement
   Kind: 3 - RETURN
   TypeExpression: XmlElement
   DefaultValueExpression:

**Parameter** providerUri
   Kind: 0 - IN
   TypeExpression: string
   DefaultValueExpression:

Class ServiceConfig
   HypermediaService::Microsoft::BusinessFramework::Hypermedia::Service::ServiceConfig
   Stereotype: 
   Visibility: 0 - PUBLIC
   Multiplicity: 
   Persistence: 0 - PERSISTENT

   Documentation: Used to read hypermedia service configuration settings from the BusinessFramework.config file.

**Attribute** ProviderRegistrationPath
   TypeExpression: string
   DefaultValueExpression:

**Attribute** RequestTimeout
   TypeExpression: int
   DefaultValueExpression:

**Operation** ProviderRegistrationPath
   Visibility: 0 - PUBLIC

   **Parameter** value
      Kind: 0 - IN
      TypeExpression: string
      DefaultValueExpression:

**Operation** ProviderRegistrationPath
   Visibility: 0 - PUBLIC

   **Parameter** string
      Kind: 3 - RETURN
      TypeExpression: string
      DefaultValueExpression:

**Operation** RequestTimeout
   Visibility: 0 - PUBLIC
**Parameter** value
   Kind: 0 - IN
   TypeExpression: int
   DefaultValueExpression:

**Operation** RequestTimeout
   Visibility: 0 - PUBLIC

**Parameter** int
   Kind: 3 - RETURN
   TypeExpression: int
   DefaultValueExpression:

**Package References**
   HypermediaService::References
   Stereotype: modelReferences
Appendix D
Detailed Class Report
Pages 1-29
Detailed Class Report

Model name: Hypermedia
Root file: C:\depot\bedrock3\nbf\Hypermedia\Hypermedia.mdx

Package Microsoft
Hypermedia: Microsoft
Stereotype: namespace

Package BusinessFramework
Hypermedia: Microsoft: BusinessFramework
Stereotype: namespace

Package Hypermedia
Hypermedia: Microsoft: BusinessFramework: Hypermedia
Stereotype: namespace

Class CollectionEventArgs
Hypermedia: Microsoft: BusinessFramework: Hypermedia: CollectionEventArgs
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Provides data for the events of a collection.

Attribute EventType
TypeExpression: CollectionEventArgs
DefaultExpression:

Attribute Index
TypeExpression: int
DefaultExpression:

Attribute Item
TypeExpression: object
DefaultExpression:

Operation CollectionEventArgs
Visibility: 3 - PACKAGE

Parameter index
Kind: 0 - IN
TypeExpression: int
DefaultExpression:

Parameter item
Parameter **eventType**
Kind: 0 - IN
TypeExpression: CollectionEventType
DefaultValueExpression:

**Operation EventType**
Visibility: 3 - PACKAGE

Parameter **CollectionEventType**
Kind: 3 - RETURN
TypeExpression: CollectionEventType
DefaultValueExpression:

**Operation Index**
Visibility: 0 - PUBLIC

Parameter **int**
Kind: 3 - RETURN
TypeExpression: int
DefaultValueExpression:

**Operation Item**
Visibility: 0 - PUBLIC

Parameter **object**
Kind: 3 - RETURN
TypeExpression: object
DefaultValueExpression:

**Class CollectionEventHandler**
Hypermedia: Microsoft::BusinessFramework: Hypermedia::CollectionEventHandler
Stereotype: delegate
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Represents the method that will handle the event triggered by a collection.

**Operation Invoke**
Visibility: 0 - PUBLIC

Parameter **sender**
Kind: 0 - IN
**Enumeration CollectionEventType**

Hypermedia: Microsoft: BusinessFramework: Hypermedia: CollectionEventType
Stereotype: Visibility: 3 - PACKAGE
Multiplicity: Persistence: 0 - PERSISTENT

Documentation: Specifies a type of event that may occur in a collection.

**Enumeration Literal Inserting**

value:

**Enumeration Literal Setting**

value:

**Class DiagnosticInfo**

Hypermedia: Microsoft: BusinessFramework: Hypermedia: DiagnosticInfo
Stereotype: Visibility: 0 - PUBLIC
Multiplicity: Persistence: 0 - PERSISTENT

Documentation: A <see cref="DiagnosticInfo"/> object contains diagnostic information that can be included in a <see cref="Response"/> to help determine the cause of a failure.

**Attribute Exceptions**

TypeExpression: ExceptionInfo[]
Default:Expression:"

**Operation AddException**

Visibility: 0 - PUBLIC

**Parameter int**

Kind: 3 - RETURN
TypeExpression: int
Default:Expression:"

**Parameter exception**
Kind: 0 - IN
TypeExpression: ExceptionInfo
DefaultValueExpression:

**Operation DiagnosticInfo**
Visibility: 0 - PUBLIC

**Operation DiagnosticInfo**
Visibility: 0 - PUBLIC

**Parameter** `exception`
Kind: 0 - IN
TypeExpression: ExceptionInfo
DefaultValueExpression:

**Operation DiagnosticInfo**
Visibility: 3 - PACKAGE

**Parameter** `xmlDiagnosticInfo`
Kind: 0 - IN
TypeExpression: Xml.DiagnosticInfo
DefaultValueExpression:

**Operation Exceptions**
Visibility: 0 - PUBLIC

**Parameter** `ExceptionInfo[]`
Kind: 3 - RETURN
TypeExpression: ExceptionInfo[]
DefaultValueExpression:

**Operation ToXmlObject**
Visibility: 3 - PACKAGE

**Parameter** `Xml.DiagnosticInfo`
Kind: 3 - RETURN
TypeExpression: Xml.DiagnosticInfo
DefaultValueExpression:

**Class ExceptionInfo**
Hypermime::Microsoft::BusinessFramework::Hypermime::ExceptionInfo
Stereotype: Visibility: 0 - PUBLIC
Multiplicity: Persistence: 0 - PERSISTENT
Documentation: `<see cref="ExceptionInfo"> contains information about an instance of `<see cref="Exception"> that was thrown by an `<see cref="IHypermediaProvider">.

**Attribute ExceptionType**
TypeExpression: `string`
DefaultExpression:

**Attribute InnerExceptionInfo**
TypeExpression: `ExceptionInfo`
DefaultExpression:

**Attribute Message**
TypeExpression: `string`
DefaultExpression:

**Attribute Provider**
TypeExpression: `Uri`
DefaultExpression:

**Attribute Source**
TypeExpression: `string`
DefaultExpression:

**Attribute StackTrace**
TypeExpression: `string`
DefaultExpression:

**Operation ExceptionInfo**
Visibility: 0 - PUBLIC

**Parameter** `exception`
Kind: 0 - IN
TypeExpression: `Exception`
DefaultExpression:

**Operation ExceptionInfo**
Visibility: 3 - PACKAGE

**Parameter** `xmlExceptionInfo`
Kind: 0 - IN
TypeExpression: `Xml.ExceptionInfo`
DefaultExpression:

**Operation ExceptionType**
Visibility: 0 - PUBLIC

**Parameter** `string`
Kind: 3 - RETURN
TypeExpression: `string`
DefaultValueExpression:

**Operation InnerExceptionInfo**
Visibility: 0 - PUBLIC

**Parameter** ExceptionInfo
Kind: 3 - RETURN
TypeExpression: ExceptionInfo
DefaultValueExpression:

**Operation Message**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation Provider**
Visibility: 0 - PUBLIC

**Parameter** value
Kind: 0 - IN
TypeExpression: Uri
DefaultValueExpression:

**Operation Provider**
Visibility: 0 - PUBLIC

**Parameter** Uri
Kind: 3 - RETURN
TypeExpression: Uri
DefaultValueExpression:

**Operation Source**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation StackTrace**
Visibility: 0 - PUBLIC

**Parameter** string
Class HypermediaConfig
Hypermedia::Microsoft::BusinessFramework::Hypermedia::HypermediaConfig
Stereotype: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Used to read hypermedia configuration settings from the BusinessFramework.config file.

Attribute HypermediaSchemaPath
TypeExpression: string
DefaultValueExpression:

Attribute Validate
TypeExpression: bool
DefaultValueExpression:

Operation HypermediaSchemaPath
Visibility: 0 - PUBLIC

Parameter value
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

Operation HypermediaSchemaPath
Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

Operation Validate
Visibility: 0 - PUBLIC

**Parameter value**
Kind: 0 - IN
TypeExpression: bool
DefaultValueExpression:

**Operation Validate**
Visibility: 0 - PUBLIC

**Parameter bool**
Kind: 3 - RETURN
TypeExpression: bool
DefaultValueExpression:

---

**Class HypermediaRequest**
Hypermedia:Microsoft::BusinessFramework::Hypermedia::HypermediaRequest
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Represents a request to retrieve hypermedia.

**Attribute DynamicViewId**
TypeExpression: Guid
DefaultValueExpression:

**Attribute ElementName**
TypeExpression: string
DefaultValueExpression:

**Attribute Key**
TypeExpression: IKey
DefaultValueExpression:

**Attribute LinkCategories**
TypeExpression: StringCollection
DefaultValueExpression:

**Attribute LinkTypes**
TypeExpression: LinkType
DefaultValueExpression:

**Attribute ProviderInfoVersion**
TypeExpression: string
DefaultValueExpression:
**Attribute XmlConversionType**

TypeExpression: Type
DefaultValueExpression:

**Operation DynamicViewld**

Visibility: 0 - PUBLIC

Parameter Guid

Kind: 3 - RETURN
TypeExpression: Guid
DefaultValueExpression:

**Operation ElementName**

Visibility: 0 - PUBLIC

Parameter string

Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation FromXmlObject**

Visibility: 1 - PROTECTED

Parameter xmlSerializationBase

Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

**Operation HypermediaRequest**

Visibility: 0 - PUBLIC

Parameter key

Kind: 0 - IN
TypeExpression: IKey
DefaultValueExpression:

Parameter dynamicViewld

Kind: 0 - IN
TypeExpression: Guid
DefaultValueExpression:

Parameter linkTypes

Kind: 0 - IN
TypeExpression: LinkType
DefaultValueExpression:

**Operation HypermediaRequest**
Visibility: 0 - PUBLIC

Parameter xmlRepresentation
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

Operation HypermediaRequest
Visibility: 1 - PROTECTED

Parameter xmlRepresentation
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

Parameter xmlConversionType
Kind: 0 - IN
TypeExpression: Type
DefaultValueExpression:

Parameter elementName
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

Operation Initialize
Visibility: 0 - PUBLIC

Parameter key
Kind: 0 - IN
TypeExpression: iKey
DefaultValueExpression:

Parameter dynamicViewId
Kind: 0 - IN
TypeExpression: Guid
DefaultValueExpression:

Parameter linkTypes
Kind: 0 - IN
TypeExpression: LinkType
DefaultValueExpression:

Operation Key
Visibility: 0 - PUBLIC

Parameter iKey
Kind: 3 - RETURN
TypeExpression: iKey
DefaultValueExpression:
**Operation LinkCategories**
Visibility: 0 - PUBLIC

**Parameter** StringCollection
Kind: 3 - RETURN
TypeExpression: StringCollection
DefaultValueExpression:

**Operation LinkTypes**
Visibility: 0 - PUBLIC

**Parameter** LinkType
Kind: 3 - RETURN
TypeExpression: LinkType
DefaultValueExpression:

**Operation ProviderInfoVersion**
Visibility: 0 - PUBLIC

**Parameter** value
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

**Operation ProviderInfoVersion**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation ToXmlObject**
Visibility: 1 - PROTECTED

**Parameter** xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

**Operation XmlConversionType**
Visibility: 1 - PROTECTED

**Parameter** Type
Kind: 3 - RETURN
Class HypermediaResponse
Hypermedia::Microsoft::BusinessFramework::Hypermedia::HypermediaResponse
Stereotype: 
Visibility: 0 - PUBLIC
Multiplicity: 
Persistence: 0 - PERSISTENT

Documentation: Represents a response to a request to retrieve hypermedia.

Attribute ClassLinks
TypeExpression: LinkCollection
DefaultValueExpression:

Attribute ElementName
TypeExpression: string
DefaultValueExpression:

Attribute InstanceLinks
TypeExpression: LinkCollection
DefaultValueExpression:

Attribute InstanceSpecificLinks
TypeExpression: LinkCollection
DefaultValueExpression:

Attribute XmlConversionType
TypeExpression: Type
DefaultValueExpression:

Operation ClassLinks
Visibility: 0 - PUBLIC

Parameter LinkCollection
Kind: 3 - RETURN
TypeExpression: LinkCollection
DefaultValueExpression:

Operation ElementName
Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:
Operation FromXmlObject
Visibility: 1 - PROTECTED

Parameter xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

Operation HypermediaResponse
Visibility: 0 - PUBLIC

Parameter responseCode
Kind: 0 - IN
TypeExpression: ResponseCode
DefaultValueExpression:

Operation HypermediaResponse
Visibility: 0 - PUBLIC

Parameter xmlRepresentation
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

Operation HypermediaResponse
Visibility: 1 - PROTECTED

Parameter xmlRepresentation
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

Parameter xmlConversionType
Kind: 0 - IN
TypeExpression: Type
DefaultValueExpression:

Parameter elementName
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

Operation InstanceLinks
Visibility: 0 - PUBLIC

Parameter LinkCollection
Kind: 3 - RETURN
TypeExpression: LinkCollection
Operation InstanceSpecificLinks
Visibility: 0 - PUBLIC

Parameter LinkCollection
Kind: 3 - RETURN
TypeExpression: LinkCollection
DefaultValueExpression:

Operation ToXmlObject
Visibility: 1 - PROTECTED

Parameter xmlSerializationBase
Kind: 0 - IN
TypeExpression: XmlSerializationBase
DefaultValueExpression:

Operation XmlConversionType
Visibility: 1 - PROTECTED

Parameter Type
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

Interface IHypermediaProvider
Hypermedia::Microsoft::BusinessFramework::Hypermedia::IHypermediaProvider
Stereotype:
Visibility: 0 - PUBLIC
 Multiplicity:
 Persistence: 0 - PERSISTENT

Documentation: Specifies the methods that need to be implemented to provide hypermedia.

Operation GetHypermedia
Visibility: 0 - PUBLIC

Parameter XmlElement
Kind: 3 - RETURN
TypeExpression: XmlElement
DefaultValueExpression:

Parameter hypermediaRequest
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:
**Operation GetProviderInfo**  
Visibility: 0 - PUBLIC

**Parameter** XmlElement  
Kind: 3 - RETURN  
TypeExpression: XmlElement  
DefaultValueExpression:

**Parameter** providerInfoRequest  
Kind: 0 - IN  
TypeExpression: XmlElement  
DefaultValueExpression:

**Operation TraverseLink**  
Visibility: 0 - PUBLIC

**Parameter** XmlElement  
Kind: 3 - RETURN  
TypeExpression: XmlElement  
DefaultValueExpression:

**Parameter** traverseRequest  
Kind: 0 - IN  
TypeExpression: XmlElement  
DefaultValueExpression:

**Interface IXmlRepresentable**

Hypermedia: Microsoft::BusinessFramework::Hypermedia::IXmlRepresentable  
Stereotype:  
Visibility: 0 - PUBLIC  
Multiplicity:  
Persistence: 0 - PERSISTENT  
Documentation: Specifies how to convert an `<see cref="Object"/>` into an XML representation of that `<see cref="Object"/>`.

**Operation ToXml**  
Visibility: 0 - PUBLIC

**Parameter** XmlElement  
Kind: 3 - RETURN  
TypeExpression: XmlElement  
DefaultValueExpression:

**Operation ToXml**  
Visibility: 0 - PUBLIC
Class Link
Hypermedia:Microsoft:BusinessFramework::Hypermedia::Link
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Represents either a relationship to a destination node, or an action that can be performed.

Attribute AssociatedProperties
TypeExpression: PropertyIdentifierList
DefaultValueExpression:

Attribute DestinationNodeClass
TypeExpression: Type
DefaultValueExpression:

Attribute DisplayName
TypeExpression: string
DefaultValueExpression:

Attribute ElementName
TypeExpression: string
DefaultValueExpression:

Attribute IsAction
TypeExpression: bool
DefaultValueExpression:

Attribute IsTraversable
TypeExpression: bool
DefaultValueExpression:

Attribute Key
TypeExpression: IKey
DefaultValueExpression:

Attribute LinkCategory
**Attribute LinkSource**
TypeExpression: Uri
DefaultValueExpression:

**Attribute LinkType**
TypeExpression: LinkType
DefaultValueExpression:

**Attribute ProviderData**
TypeExpression: XmlDocument
DefaultValueExpression:

**Attribute SourceNodeClass**
TypeExpression: Type
DefaultValueExpression:

**Attribute XmlConversionType**
TypeExpression: Type
DefaultValueExpression:

**Operation AssociatedProperties**
Visibility: 0 - PUBLIC

**Parameter** PropertyIdentifierList
Kind: 3 - RETURN
TypeExpression: PropertyIdentifierList
DefaultValueExpression:

**Operation DestinationNodeClass**
Visibility: 0 - PUBLIC

**Parameter** Type
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

**Operation DisplayName**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation DisplayName**
Visibility: 0 - PUBLIC
Parameter value
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

Operation ElementName
Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

Operation FromXmlObject
Visibility: 1 - PROTECTED

Parameter xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

Operation IsAction
Visibility: 0 - PUBLIC

Parameter bool
Kind: 3 - RETURN
TypeExpression: bool
DefaultValueExpression:

Operation IsTraversable
Visibility: 0 - PUBLIC

Parameter bool
Kind: 3 - RETURN
TypeExpression: bool
DefaultValueExpression:

Operation Key
Visibility: 0 - PUBLIC

Parameter IKey
Kind: 3 - RETURN
TypeExpression: IKey
DefaultValueExpression:

Operation Link
Visibility: 1 - PROTECTED

**Parameter** xmlRepresentation
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

**Parameter** xmlConversionType
Kind: 0 - IN
TypeExpression: Type
DefaultValueExpression:

**Parameter** elementName
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

**Operation Link**
Visibility: 0 - PUBLIC

**Parameter** xmlRepresentation
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

**Operation Link**
Visibility: 3 - PACKAGE

**Parameter** xmlLink
Kind: 0 - IN
TypeExpression: XmlLink
DefaultValueExpression:

**Operation Link**
Visibility: 0 - PUBLIC

**Parameter** linkType
Kind: 0 - IN
TypeExpression: LinkType
DefaultValueExpression:

**Parameter** linkCategory
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

**Parameter** destinationNodeClass
Kind: 0 - IN
TypeExpression: Type
DefaultValueExpression:
Parameter key
  Kind: 0 - IN
  TypeExpression: IKey
  DefaultValueExpression:

Parameter isAction
  Kind: 0 - IN
  TypeExpression: bool
  DefaultValueExpression:

Operation LinkCategory
  Visibility: 0 - PUBLIC

Parameter string
  Kind: 3 - RETURN
  TypeExpression: string
  DefaultValueExpression:

Operation LinkSource
  Visibility: 0 - PUBLIC

Parameter value
  Kind: 0 - IN
  TypeExpression: Uri
  DefaultValueExpression:

Operation LinkSource
  Visibility: 0 - PUBLIC

Parameter Uri
  Kind: 3 - RETURN
  TypeExpression: Uri
  DefaultValueExpression:

Operation LinkType
  Visibility: 0 - PUBLIC

Parameter LinkType
  Kind: 3 - RETURN
  TypeExpression: LinkType
  DefaultValueExpression:

Operation ProviderData
  Visibility: 0 - PUBLIC

Parameter value
  Kind: 0 - IN
Operation ProviderData

Visibility: 0 - PUBLIC

Parameter XmlDocument
Kind: 3 - RETURN
TypeExpression: XmlDocument
DefaultValueExpression:

Operation SourceNodeClass

Visibility: 0 - PUBLIC

Parameter Type
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

Operation ToXmlObject

Visibility: 1 - PROTECTED

Parameter xmlSerializationBase
Kind: 3 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

Operation ToXmlObject

Visibility: 3 - PACKAGE

Parameter XmlLink
Kind: 3 - RETURN
TypeExpression: Xml Link
DefaultValueExpression:

Operation XmlConversionType

Visibility: 1 - PROTECTED

Parameter Type
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

Class LinkCollection
Hypermedia::Microsoft::BusinessFramework::Hypermedia::LinkCollection
Stereotype: 
Visibility: 0 - PUBLIC
Multiplicity: 
Persistence: 0 - PERSISTENT

Documentation: Represents a collection of <see cref="Link"/> objects.

**Operation Add**
Visibility: 0 - PUBLIC

**Parameter int**
Kind: 3 - RETURN
TypeExpression: Int
DefaultValueExpression:

**Parameter link**
Kind: 0 - IN
TypeExpression: Link
DefaultValueExpression:

**Operation Clear**
Visibility: 0 - PUBLIC

**Operation Clone**
Visibility: 0 - PUBLIC

**Parameter object**
Kind: 3 - RETURN
TypeExpression: object
DefaultValueExpression:

**Operation Contains**
Visibility: 0 - PUBLIC

**Parameter bool**
Kind: 3 - RETURN
TypeExpression: bool
DefaultValueExpression:

**Parameter link**
Kind: 0 - IN
TypeExpression: Link
DefaultValueExpression:

**Operation IndexOf**
Visibility: 0 - PUBLIC

**Parameter int**
Kind: 3 - RETURN
TypeExpression: int
DefaultValueExpression:

Parameter link
Kind: 0 - IN
TypeExpression: Link
DefaultValueExpression:

Operation Insert
Visibility: 0 - PUBLIC

Parameter index
Kind: 0 - IN
TypeExpression: int
DefaultValueExpression:

Parameter link
Kind: 0 - IN
TypeExpression: Link
DefaultValueExpression:

Operation LinkCollection
Visibility: 0 - PUBLIC

Operation LinkCollection
Visibility: 0 - PUBLIC

Parameter links
Kind: 0 - IN
TypeExpression: LinkCollection
DefaultValueExpression:

Operation LinkCollection
Visibility: 3 - PACKAGE

Parameter xmlLinks
Kind: 0 - IN
TypeExpression: Xml.Link[]
DefaultValueExpression:

Operation Remove
Visibility: 0 - PUBLIC

Parameter link
Kind: 0 - IN
TypeExpression: Link
DefaultValueExpression:
**Operation RemoveAt**
Visibility: 0 - PUBLIC

**Parameter** `index`
Kind: 0 - IN
TypeExpression: int
DefaultValueExpression:

**Operation ToXmLObject**
Visibility: 3 - PACKAGE

**Parameter** `XmlLink[]`
Kind: 3 - RETURN
TypeExpression: XmlLink[]
DefaultValueExpression:

**Enumeration LinkType**
Hypermedia::Microsoft::BusinessFramework::Hypermedia::LinkType
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Specifies the scope of a `<see cref="Link"/>`.

**Enumeration Literal Class**
value: 1 << 0

**Enumeration Literal Instance**
value: 1 << 1

**Enumeration Literal InstanceSpecific**
value: 1 << 2

**Class PropertyIdentifier**
Hypermedia::Microsoft::BusinessFramework::Hypermedia::PropertyIdentifier
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Identifies a property on a node class.

**Attribute FullName**
TypeExpression: string
DefaultValueExpression:

Attribute Name
TypeExpression: string
DefaultValueExpression:

Attribute NodeClassName
TypeExpression: string
DefaultValueExpression:

Attribute PropertyType
TypeExpression: Type
DefaultValueExpression:

Operation Clone
Visibility: 0 - PUBLIC

Parameter object
Kind: 3 - RETURN
TypeExpression: object
DefaultValueExpression:

Operation Equals
Visibility: 0 - PUBLIC

Parameter bool
Kind: 3 - RETURN
TypeExpression: bool
DefaultValueExpression:

Parameter obj
Kind: 0 - IN
TypeExpression: object
DefaultValueExpression:

Operation FullName
Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

Operation GetHashCode
Visibility: 0 - PUBLIC

Parameter int
Kind: 3 - RETURN
TypeExpression: int
DefaultValueExpression:

**Operation Name**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation NodeClass_**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation operator !=**
Visibility: 0 - PUBLIC

**Parameter** bool
Kind: 3 - RETURN
TypeExpression: bool
DefaultValueExpression:

**Parameter** left
Kind: 0 - IN
TypeExpression: PropertyIdentifier
DefaultValueExpression:

**Parameter** right
Kind: 0 - IN
TypeExpression: PropertyIdentifier
DefaultValueExpression:

**Operation operator ==**
Visibility: 0 - PUBLIC

**Parameter** bool
Kind: 3 - RETURN
TypeExpression: bool
DefaultValueExpression:

**Parameter** left
Kind: 0 - IN
TypeExpression: PropertyIdentifier
DefaultValueExpression:
Parameter right
  Kind: 0 - IN
  TypeExpression: PropertyIdentifier
  DefaultValueExpression:

Operation PropertyIdentifier
  Visibility: 0 - PUBLIC

Parameter nodeClassName
  Kind: 0 - IN
  TypeExpression: string
  DefaultValueExpression:

Parameter name
  Kind: 0 - IN
  TypeExpression: string
  DefaultValueExpression:

Operation PropertyIdentifier
  Visibility: 0 - PUBLIC

Parameter propertyIdentifier
  Kind: 0 - IN
  TypeExpression: PropertyIdentifier
  DefaultValueExpression:

Operation PropertyIdentifier
  Visibility: 3 - PACKAGE

Parameter xmlProperty
  Kind: 0 - IN
  TypeExpression: Xml.PropertyIdentifier
  DefaultValueExpression:

Operation PropertyType
  Visibility: 0 - PUBLIC

Parameter Type
  Kind: 3 - RETURN
  TypeExpression: Type
  DefaultValueExpression:

Operation ToXmlObject
  Visibility: 3 - PACKAGE

Parameter Xml.PropertyIdentifier
  Kind: 3 - RETURN
Class PropertyIdentifierList

Hypermedia::Microsoft::BusinessFramework::Hypermedia::PropertyIdentifierList
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Represents an ordered list of unique <see cref="PropertyIdentifier"/> objects.

Operation Add
Visibility: 0 - PUBLIC

Parameter int
Kind: 3 - RETURN
TypeExpression: int
DefaultValueExpression:

Parameter property
Kind: 0 - IN
TypeExpression: PropertyIdentifier
DefaultValueExpression:

Operation Clear
Visibility: 0 - PUBLIC

Operation Clone
Visibility: 0 - PUBLIC

Parameter object
Kind: 3 - RETURN
TypeExpression: object
DefaultValueExpression:

Operation Insert
Visibility: 0 - PUBLIC

Parameter index
Kind: 0 - IN
TypeExpression: int
DefaultValueExpression:

Parameter property
Kind: 0 - IN
TypeExpression: PropertyIdentifier
DefaultValueExpression:
**Operation PropertyIdentifierList**  
Visibility: 0 - PUBLIC

**Operation PropertyIdentifierList**  
Visibility: 0 - PUBLIC

Parameter properties  
Kind: 0 - IN  
TypeExpression: ReadOnlyPropertyIdentifierList  
DefaultValueExpression:

**Operation PropertyIdentifierList**  
Visibility: 3 - PACKAGE

Parameter xmlProperties  
Kind: 0 - IN  
TypeExpression: Xml.PropertyIdentifier[]  
DefaultValueExpression:

**Operation Remove**  
Visibility: 0 - PUBLIC

Parameter property  
Kind: 0 - IN  
TypeExpression: PropertyIdentifier  
DefaultValueExpression:

**Operation RemoveAt**  
Visibility: 0 - PUBLIC

Parameter index  
Kind: 0 - IN  
TypeExpression: int  
DefaultValueExpression:

**Class ProviderInfoRequest**  
Hypermedia:M:Microsoft:BusinessFramework::Hypermedia::ProviderInfoRequest  
Stereotype:
Visibility: 0 - PUBLIC  
Multiplicity:  
Persistence: 0 - PERSISTENT

Documentation: Represents a request for provider information.
Appendix E

Detailed Class Report

Pages 1-55
Detailed Class Report

Model name: Hypermedia
Root file: C:\depot\bedrock3\inbh\Hypermedia\Hypermedia.mdx

Package Microsoft
Hypermedia::Microsoft
Stereotype: namespace

Package BusinessFramework
Hypermedia::Microsoft::BusinessFramework
Stereotype: namespace

Package Hypermedia
Hypermedia::Microsoft::BusinessFramework::Hypermedia
Stereotype: namespace

Class CollectionEventArgs
Hypermedia::Microsoft::BusinessFramework::Hypermedia::CollectionEventArgs
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Provides data for the events of a collection.

Attribute EventType
TypeExpression: CollectionEventType
DefaultValueExpression:

Attribute Index
TypeExpression: int
DefaultValueExpression:

Attribute Item
TypeExpression: object
DefaultValueExpression:

Operation CollectionEventArgs
Visibility: 3 - PACKAGE

Parameter index
Kind: 0 - IN
TypeExpression: int
DefaultValueExpression:

Parameter item
Kind: 0 - IN
TypeExpression: object
DefaultValueExpression:

Parameter eventType
Kind: 0 - IN
TypeExpression: CollectionEventType
DefaultValueExpression:

Operation EventType
Visibility: 3 - PACKAGE

Parameter CollectionEventType
Kind: 3 - RETURN
TypeExpression: CollectionEventType
DefaultValueExpression:

Operation Index
Visibility: 0 - PUBLIC

Parameter int
Kind: 3 - RETURN
TypeExpression: int
DefaultValueExpression:

Operation Item
Visibility: 0 - PUBLIC

Parameter object
Kind: 3 - RETURN
TypeExpression: object
DefaultValueExpression:

Class CollectionEventHandler
Hypermedia: Microsoft::BusinessFramework::Hypermedia::CollectionEventHandler
Stereotype: delegate
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Represents the method that will handle the event triggered by a collection.

Operation Invoke
Visibility: 0 - PUBLIC

Parameter sender
Kind: 0 - IN
TypeExpression: object
DefaultValueExpression:

Parameter eventArgs
   Kind: 0 - IN
   TypeExpression: CollectionEventArgs
   DefaultValueExpression:

Enumeration CollectionEventType
   Hypermedia: Microsoft: BusinessFramework: Hypermedia:: CollectionEventType
   Stereotype:  
   Visibility: 3 - PACKAGE
   Multiplicity:  
   Persistence: 0 - PERSISTENT
   Documentation: Specifies a type of event that may occur in a collection.

   Enumeration Literal Inserting
      value:

   Enumeration Literal Setting
      value:

Class DiagnosticInfo
   Hypermedia: Microsoft: BusinessFramework: Hypermedia:: DiagnosticInfo
   Stereotype:  
   Visibility: 0 - PUBLIC
   Multiplicity:  
   Persistence: 0 - PERSISTENT
   Documentation: A <see cref="DiagnosticInfo"/> object contains diagnostic information that can be included in a <see cref="Response"/> to help determine the cause of a failure.

Attribute Exceptions
   TypeExpression: ExceptionInfo[]
   DefaultValueExpression:

Operation AddException
   Visibility: 0 - PUBLIC
   
Parameter int
   Kind: 3 - RETURN
   TypeExpression: int
   DefaultValueExpression:

Parameter exception
Kind: 0 - IN
TypeExpression: ExceptionInfo
DefaultValueExpression:

**Operation DiagnosticInfo**
Visibility: 0 - PUBLIC

**Operation DiagnosticInfo**
Visibility: 0 - PUBLIC

**Parameter** exception
Kind: 0 - IN
TypeExpression: ExceptionInfo
DefaultValueExpression:

**Operation DiagnosticInfo**
Visibility: 3 - PACKAGE

**Parameter** xmlDiagnosticInfo
Kind: 0 - IN
TypeExpression: Xml.DiagnosticInfo
DefaultValueExpression:

**Operation Exceptions**
Visibility: 0 - PUBLIC

**Parameter** ExceptionInfo[]
Kind: 3 - RETURN
TypeExpression: ExceptionInfo[]
DefaultValueExpression:

**Operation ToXmlObject**
Visibility: 3 - PACKAGE

**Parameter** Xml.DiagnosticInfo
Kind: 3 - RETURN
TypeExpression: Xml.DiagnosticInfo
DefaultValueExpression:

**Class ExceptionInfo**

Hypermedia::Microsoft::BusinessFramework::Hypermedia::ExceptionInfo
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT
Documentation: &lt;see cref="ExceptionInfo"&gt; contains information about an instance of &lt;see cref="Exception"&gt; that was thrown by an &lt;see cref="IHypermediaProvider"&gt;.

**Attribute ExceptionType**

TypeExpression: string
DefaultValueExpression:

**Attribute InnerExceptionInfo**

TypeExpression: ExceptionInfo
DefaultValueExpression:

**Attribute Message**

TypeExpression: string
DefaultValueExpression:

**Attribute Provider**

TypeExpression: Uri
DefaultValueExpression:

**Attribute Source**

TypeExpression: string
DefaultValueExpression:

**Attribute StackTrace**

TypeExpression: string
DefaultValueExpression:

**Operation ExceptionInfo**

Visibility: 0 - PUBLIC

**Parameter exception**

Kind: 0 - IN
TypeExpression: Exception
DefaultValueExpression:

**Operation ExceptionInfo**

Visibility: 3 - PACKAGE

**Parameter xmlExceptionInfo**

Kind: 0 - IN
TypeExpression: Xml.ExceptionInfo
DefaultValueExpression:

**Operation ExceptionType**

Visibility: 0 - PUBLIC

**Parameter string**

Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation InnerExceptionInfo**
Visibility: 0 - PUBLIC

**Parameter** ExceptionInfo
Kind: 3 - RETURN
TypeExpression: ExceptionInfo
DefaultValueExpression:

**Operation Message**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation Provider**
Visibility: 0 - PUBLIC

**Parameter** value
Kind: 0 - IN
TypeExpression: Uri
DefaultValueExpression:

**Operation Provider**
Visibility: 0 - PUBLIC

**Parameter** Uri
Kind: 3 - RETURN
TypeExpression: Uri
DefaultValueExpression:

**Operation Source**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation StackTrace**
Visibility: 0 - PUBLIC

**Parameter** string
Operation ToXmlObject
Visibility: 3 - PACKAGE

Parameter Xml.ExceptionInfo
Kind: 3 - RETURN
TypeExpression: Xml.ExceptionInfo
DefaultValueExpression:

Class HypermediaConfig
Hypermedia:.:Microsoft:.:BusinessFramework:.:Hypermedia:.:HypermediaConfig
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Used to read hypermedia configuration settings from the BusinessFramework.config file.

Attribute HypermediaSchemaPath
TypeExpression: string
DefaultValueExpression:

Attribute Validate
TypeExpression: bool
DefaultValueExpression:

Operation HypermediaSchemaPath
Visibility: 0 - PUBLIC

Parameter value
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

Operation HypermediaSchemaPath
Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

Operation Validate
Visibility: 0 - PUBLIC

**Parameter** value

Kind: 0 - IN
TypeExpression: bool
DefaultValueExpression:

**Operation Validate**

Visibility: 0 - PUBLIC

**Parameter** bool

Kind: 3 - RETURN
TypeExpression: bool
DefaultValueExpression:

**Class HypermediaRequest**

Hypermedia::Microsoft::BusinessFramework::Hypermedia::HypermediaRequest
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity: 0 - PERSISTENT

Documentation: Represents a request to retrieve hypermedia.

**Attribute DynamicViewId**

TypeExpression: Guid
DefaultValueExpression:

**Attribute ElementName**

TypeExpression: string
DefaultValueExpression:

**Attribute Key**

TypeExpression: IKey
DefaultValueExpression:

**Attribute LinkCategories**

TypeExpression: StringCollection
DefaultValueExpression:

**Attribute LinkTypes**

TypeExpression: LinkType
DefaultValueExpression:

**Attribute ProviderInfoVersion**

TypeExpression: string
DefaultValueExpression:
**Attribute XmlConversionType**
TypeExpression: Type
DefaultValueExpression:

**Operation DynamicViewId**
Visibility: 0 - PUBLIC

**Parameter** Guid
Kind: 3 - RETURN
TypeExpression: Guid
DefaultValueExpression:

**Operation ElementName**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation FromXmlObject**
Visibility: 1 - PROTECTED

**Parameter** xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

**Operation HypermediaRequest**
Visibility: 0 - PUBLIC

**Parameter** key
Kind: 0 - IN
TypeExpression: IKey
DefaultValueExpression:

**Parameter** dynamicViewId
Kind: 0 - IN
TypeExpression: Guid
DefaultValueExpression:

**Parameter** linkTypes
Kind: 0 - IN
TypeExpression: LinkType
DefaultValueExpression:

**Operation HypermediaRequest**
Visibility: 0 - PUBLIC

Parameter xmlRepresentation
   Kind: 0 - IN
   TypeExpression: XmlElement
   DefaultValueExpression:

Operation HypermediaRequest
   Visibility: 1 - PROTECTED

Parameter xmlRepresentation
   Kind: 0 - IN
   TypeExpression: XmlElement
   DefaultValueExpression:

Parameter xmlConversionType
   Kind: 0 - IN
   TypeExpression: Type
   DefaultValueExpression:

Parameter elementName
   Kind: 0 - IN
   TypeExpression: string
   DefaultValueExpression:

Operation Initialize
   Visibility: 0 - PUBLIC

Parameter key
   Kind: 0 - IN
   TypeExpression: IKey
   DefaultValueExpression:

Parameter dynamicViewId
   Kind: 0 - IN
   TypeExpression: Guid
   DefaultValueExpression:

Parameter linkTypes
   Kind: 0 - IN
   TypeExpression: LinkType
   DefaultValueExpression:

Operation Key
   Visibility: 0 - PUBLIC

Parameter iKey
   Kind: 3 - RETURN
   TypeExpression: IKey
   DefaultValueExpression:
**Operation LinkCategories**
Visibility: 0 - PUBLIC

**Parameter** StringCollection
Kind: 3 - RETURN
TypeExpression: StringCollection
DefaultValueExpression:

**Operation LinkTypes**
Visibility: 0 - PUBLIC

**Parameter** LinkType
Kind: 3 - RETURN
TypeExpression: LinkType
DefaultValueExpression:

**Operation ProviderInfoVersion**
Visibility: 0 - PUBLIC

**Parameter** value
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

**Operation ProviderInfoVersion**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation ToXmlObject**
Visibility: 1 - PROTECTED

**Parameter** xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

**Operation XmlConversionType**
Visibility: 1 - PROTECTED

**Parameter** Type
Kind: 3 - RETURN
Class HypermediaResponse

Hypermedia: Microsoft.BussinessFramework::Hypermedia::HypermediaResponse
Stereotype: 0 - PUBLIC
Multiplicity: 0 - PERSISTENT

Documentation: Represents a response to a request to retrieve hypermedia.

**Attribute ClassLinks**

TypeExpression: LinkCollection
DefaultValueExpression:

**Attribute ElementName**

TypeExpression: string
DefaultValueExpression:

**Attribute InstanceLinks**

TypeExpression: LinkCollection
DefaultValueExpression:

**Attribute InstanceSpecificLinks**

TypeExpression: LinkCollection
DefaultValueExpression:

**Attribute XmlConversionType**

TypeExpression: Type
DefaultValueExpression:

**Operation ClassLinks**

Visibility: 0 - PUBLIC

Parameter LinkCollection
Kind: 3 - RETURN
TypeExpression: LinkCollection
DefaultValueExpression:

**Operation ElementName**

Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:
**Operation FromXmlObject**
Visibility: 1 - PROTECTED

**Parameter** xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

**Operation HypermediaResponse**
Visibility: 0 - PUBLIC

**Parameter** responseCode
Kind: 0 - IN
TypeExpression: ResponseCode
DefaultValueExpression:

**Operation HypermediaResponse**
Visibility: 0 - PUBLIC

**Parameter** xmlRepresentation
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

**Operation HypermediaResponse**
Visibility: 1 - PROTECTED

**Parameter** xmlRepresentation
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

**Parameter** xmlConversionType
Kind: 0 - IN
TypeExpression: Type
DefaultValueExpression:

**Parameter** elementName
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

**Operation InstanceLinks**
Visibility: 0 - PUBLIC

**Parameter** LinkCollection
Kind: 3 - RETURN
TypeExpression: LinkCollection
defaultValueExpression:

**Operation InstanceSpecificLinks**
Visibility: 0 - PUBLIC

**Parameter** LinkCollection
Kind: 3 - RETURN
TypeExpression: LinkCollection
DefaultValueExpression:

**Operation ToXmlObject**
Visibility: 1 - PROTECTED

**Parameter** xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

**Operation XmlConversionType**
Visibility: 1 - PROTECTED

**Parameter** Type
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

**Interface IHypermediaProvider**
Hypermedia: Microsoft::BusinessFramework::Hypermedia::IHypermediaProvider
Stereotype: 
Visibility: 0 - PUBLIC
Multiplicity: 
Persistence: 0 - PERSISTENT

Documentation: Specifies the methods that need to be implemented to provide hypermedia.

**Operation GetHypermedia**
Visibility: 0 - PUBLIC

**Parameter** XmlElement
Kind: 3 - RETURN
TypeExpression: XmlElement
DefaultValueExpression:

**Parameter** hypermediaRequest
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:
**Operation GetProviderInfo**
Visibility: 0 - PUBLIC

**Parameter XmlElement**
Kind: 3 - RETURN
TypeExpression: XmlElement
DefaultValueExpression:

**Parameter providerInfoRequest**
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

**Operation TraverseLink**
Visibility: 0 - PUBLIC

**Parameter XmlElement**
Kind: 3 - RETURN
TypeExpression: XmlElement
DefaultValueExpression:

**Parameter traverseRequest**
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

**Interface IXmlRepresentable**
Hypermedia::Microsoft::BusinessFramework::Hypermedia::IXmlRepresentable
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Specifies how to convert an <see cref="Object"/> into an XML representation of that <see cref="Object"/>.

**Operation ToXml**
Visibility: 0 - PUBLIC

**Parameter XmlElement**
Kind: 3 - RETURN
TypeExpression: XmlElement
DefaultValueExpression:

**Operation ToXml**
Visibility: 0 - PUBLIC
Parameter XmlElement
   Kind: 3 - RETURN
   TypeExpression: XmlElement
   DefaultValueExpression:

Parameter parentNode
   Kind: 0 - IN
   TypeExpression: XmlNode
   DefaultValueExpression:

Class Link
   Hypermedia::Microsoft::BusinessFramework::Hypermedia::Link
   Stereotype:
   Visibility: 0 - PUBLIC
   Multiplicity:
   Persistence: 0 - PERSISTENT

   Documentation: Represents either a relationship to a destination node, or an action that can be performed.

Attribute AssociatedProperties
   TypeExpression: PropertyIdentifierList
   DefaultValueExpression:

Attribute DestinationNodeClass
   TypeExpression: Type
   DefaultValueExpression:

Attribute DisplayName
   TypeExpression: string
   DefaultValueExpression:

Attribute ElementName
   TypeExpression: string
   DefaultValueExpression:

Attribute IsAction
   TypeExpression: bool
   DefaultValueExpression:

Attribute IsTraversable
   TypeExpression: bool
   DefaultValueExpression:

Attribute Key
   TypeExpression: IKey
   DefaultValueExpression:

Attribute LinkCategory
TypeExpression: string
DefaultValueExpression:

Attribute LinkSource
TypeExpression: Uri
DefaultValueExpression:

Attribute LinkType
TypeExpression: LinkType
DefaultValueExpression:

Attribute ProviderData
TypeExpression: XmlDocument
DefaultValueExpression:

Attribute SourceNodeClass
TypeExpression: Type
DefaultValueExpression:

Attribute XmlConversionType
TypeExpression: Type
DefaultValueExpression:

Operation AssociatedProperties
Visibility: 0 - PUBLIC

Parameter PropertyIdentifierList
Kind: 3 - RETURN
TypeExpression: PropertyIdentifierList
DefaultValueExpression:

Operation DestinationNodeClass
Visibility: 0 - PUBLIC

Parameter Type
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

Operation DisplayName
Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

Operation DisplayName
Visibility: 0 - PUBLIC
Parameter value
  Kind: 0 - IN
  TypeExpression: string
  DefaultValueExpression:

Operation ElementName
  Visibility: 0 - PUBLIC

Parameter string
  Kind: 3 - RETURN
  TypeExpression: string
  DefaultValueExpression:

Operation FromXmlObject
  Visibility: 1 - PROTECTED

Parameter xmlSerializationBase
  Kind: 0 - IN
  TypeExpression: Xml.SerializationBase
  DefaultValueExpression:

Operation IsAction
  Visibility: 0 - PUBLIC

Parameter bool
  Kind: 3 - RETURN
  TypeExpression: bool
  DefaultValueExpression:

Operation IsTraversable
  Visibility: 0 - PUBLIC

Parameter bool
  Kind: 3 - RETURN
  TypeExpression: bool
  DefaultValueExpression:

Operation Key
  Visibility: 0 - PUBLIC

Parameter IKey
  Kind: 3 - RETURN
  TypeExpression: IKey
  DefaultValueExpression:

Operation Link
Visibility: 1 - PROTECTED

Parameter xmlRepresentation
  Kind: 0 - IN
  TypeExpression: XmlElement
  DefaultValueExpression:

Parameter xmlConversionType
  Kind: 0 - IN
  TypeExpression: Type
  DefaultValueExpression:

Parameter elementName
  Kind: 0 - IN
  TypeExpression: string
  DefaultValueExpression:

Operation Link
  Visibility: 0 - PUBLIC

Parameter xmlRepresentation
  Kind: 0 - IN
  TypeExpression: XmlElement
  DefaultValueExpression:

Operation Link
  Visibility: 3 - PACKAGE

Parameter xmlLink
  Kind: 0 - IN
  TypeExpression: XmlLink
  DefaultValueExpression:

Operation Link
  Visibility: 0 - PUBLIC

Parameter linkType
  Kind: 0 - IN
  TypeExpression: LinkType
  DefaultValueExpression:

Parameter linkCategory
  Kind: 0 - IN
  TypeExpression: string
  DefaultValueExpression:

Parameter destinationNodeClass
  Kind: 0 - IN
  TypeExpression: Type
  DefaultValueExpression:
Parameter key
Kind: 0 - IN
TypeExpression: IKey
DefaultValueExpression:

Parameter isAction
Kind: 0 - IN
TypeExpression: bool
DefaultValueExpression:

Operation LinkCategory
Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

Operation LinkSource
Visibility: 0 - PUBLIC

Parameter value
Kind: 0 - IN
TypeExpression: Uri
DefaultValueExpression:

Operation LinkSource
Visibility: 0 - PUBLIC

Parameter Uri
Kind: 3 - RETURN
TypeExpression: Uri
DefaultValueExpression:

Operation LinkType
Visibility: 0 - PUBLIC

Parameter LinkType
Kind: 3 - RETURN
TypeExpression: LinkType
DefaultValueExpression:

Operation ProviderData
Visibility: 0 - PUBLIC

Parameter value
Kind: 0 - IN
TypeExpression: XmlDocument
DefaultValueExpression:

Operation ProviderData
Visibility: 0 - PUBLIC

Parameter XmlDocument
Kind: 3 - RETURN
TypeExpression: XmlDocument
DefaultValueExpression:

Operation SourceNodeClass
Visibility: 0 - PUBLIC

Parameter Type
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

Operation ToXmlObject
Visibility: 1 - PROTECTED

Parameter xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

Operation ToXmlObject
Visibility: 3 - PACKAGE

Parameter XmlLink
Kind: 3 - RETURN
TypeExpression: XmlLink
DefaultValueExpression:

Operation XmlConversionType
Visibility: 1 - PROTECTED

Parameter Type
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

Class LinkCollection
Hypermedia::Microsoft::BusinessFramework::Hypermedia::LinkCollection
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Represents a collection of \(<\text{see cref="Link"}>/\) objects.

**Operation Add**
Visibility: 0 - PUBLIC

**Parameter** `int`
Kind: 3 - RETURN
TypeExpression: int
DefaultValueExpression:

**Parameter** `link`
Kind: 0 - IN
TypeExpression: Link
DefaultValueExpression:

**Operation Clear**
Visibility: 0 - PUBLIC

**Operation Clone**
Visibility: 0 - PUBLIC

**Parameter** `object`
Kind: 3 - RETURN
TypeExpression: object
DefaultValueExpression:

**Operation Contains**
Visibility: 0 - PUBLIC

**Parameter** `bool`
Kind: 3 - RETURN
TypeExpression: bool
DefaultValueExpression:

**Parameter** `link`
Kind: 0 - IN
TypeExpression: Link
DefaultValueExpression:

**Operation IndexOf**
Visibility: 0 - PUBLIC

**Parameter** `int`
Kind: 3 - RETURN
TypeExpression: int
DefaultValueExpression:

**Parameter** link
Kind: 0 - IN
TypeExpression: Link
DefaultValueExpression:

**Operation Insert**
Visibility: 0 - PUBLIC

**Parameter** index
Kind: 0 - IN
TypeExpression: int
DefaultValueExpression:

**Parameter** link
Kind: 0 - IN
TypeExpression: Link
DefaultValueExpression:

**Operation LinkCollection**
Visibility: 0 - PUBLIC

**Operation LinkCollection**
Visibility: 0 - PUBLIC

**Parameter** links
Kind: 0 - IN
TypeExpression: LinkCollection
DefaultValueExpression:

**Operation LinkCollection**
Visibility: 3 - PACKAGE

**Parameter** xmlLinks
Kind: 0 - IN
TypeExpression: XmlLink[]
DefaultValueExpression:

**Operation Remove**
Visibility: 0 - PUBLIC

**Parameter** link
Kind: 0 - IN
TypeExpression: Link
DefaultValueExpression:
Operation RemoveAt
Visibility: 0 - PUBLIC

Parameter index
Kind: 0 - IN
TypeExpression: int
DefaultValueExpression:

Operation ToXmlObject
Visibility: 3 - PACKAGE

Parameter XmlLink[]
Kind: 3 - RETURN
TypeExpression: XmlLink[]
DefaultValueExpression:

Enumeration LinkType
Hypermedia::Microsoft::BusinessFramework::Hypermedia::LinkType
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Specifies the scope of a <see cref="Link"/>.

Enumeration Literal Class
value: 1 << 0

Enumeration Literal Instance
value: 1 << 1

Enumeration Literal InstanceSpecific
value: 1 << 2

Class PropertyIdentifier
Hypermedia::Microsoft::BusinessFramework::Hypermedia::PropertyIdentifier
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Identifies a property on a node class.

Attribute FullName
**Attribute Name**
- TypeExpression: string
- DefaultValueExpression:

**Attribute NodeClassName**
- TypeExpression: string
- DefaultValueExpression:

**Attribute PropertyType**
- TypeExpression: Type
- DefaultValueExpression:

**Operation Clone**
- Visibility: 0 - PUBLIC

- Parameter object
  - Kind: 3 - RETURN
  - TypeExpression: object
  - DefaultValueExpression:

**Operation Equals**
- Visibility: 0 - PUBLIC

- Parameter bool
  - Kind: 3 - RETURN
  - TypeExpression: bool
  - DefaultValueExpression:

- Parameter obj
  - Kind: 0 - IN
  - TypeExpression: object
  - DefaultValueExpression:

**Operation FullName**
- Visibility: 0 - PUBLIC

- Parameter string
  - Kind: 3 - RETURN
  - TypeExpression: string
  - DefaultValueExpression:

**Operation GetHashCode**
- Visibility: 0 - PUBLIC

- Parameter int
  - Kind: 3 - RETURN
Operation Name
Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

Operation NodeClassName
Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

Operation operator !=
Visibility: 0 - PUBLIC

Parameter bool
Kind: 3 - RETURN
TypeExpression: bool
DefaultValueExpression:

Parameter left
Kind: 0 - IN
TypeExpression: PropertyIdentifier
DefaultValueExpression:

Parameter right
Kind: 0 - IN
TypeExpression: PropertyIdentifier
DefaultValueExpression:

Operation operator ==
Visibility: 0 - PUBLIC

Parameter bool
Kind: 3 - RETURN
TypeExpression: bool
DefaultValueExpression:

Parameter left
Kind: 0 - IN
TypeExpression: PropertyIdentifier
DefaultValueExpression:
Parameter right
Kind: 0 - IN
TypeExpression: PropertyIdentifier
DefaultValueExpression:

Operation PropertyIdentifier
Visibility: 0 - PUBLIC

Parameter nodeClassName
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

Parameter name
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

Operation PropertyIdentifier
Visibility: 0 - PUBLIC

Parameter propertyIdentifier
Kind: 0 - IN
TypeExpression: PropertyIdentifier
DefaultValueExpression:

Operation PropertyIdentifier
Visibility: 3 - PACKAGE

Parameter xmlProperty
Kind: 0 - IN
TypeExpression: Xml.PropertyIdentifier
DefaultValueExpression:

Operation PropertyType
Visibility: 0 - PUBLIC

Parameter Type
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

Operation ToXmlObject
Visibility: 3 - PACKAGE

Parameter Xml.PropertyIdentifier
Kind: 3 - RETURN
TypeExpression: XmlNode
Default:ValueExpression:

**Class PropertyIdentifierList**

Hypermedia: Microsoft::BusinessFramework::Hypermedia::PropertyIdentifierList
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Represents an ordered list of unique <see cref="PropertyIdentifier"/> objects.

**Operation Add**
Visibility: 0 - PUBLIC

**Parameter** int
Kind: 3 - RETURN
TypeExpression: int
Default:ValueExpression:

**Parameter** property
Kind: 0 - IN
TypeExpression: PropertyIdentifier
Default:ValueExpression:

**Operation Clear**
Visibility: 0 - PUBLIC

**Operation Clone**
Visibility: 0 - PUBLIC

**Parameter** object
Kind: 3 - RETURN
TypeExpression: object
Default:ValueExpression:

**Operation Insert**
Visibility: 0 - PUBLIC

**Parameter** index
Kind: 0 - IN
TypeExpression: int
Default:ValueExpression:

**Parameter** property
Kind: 0 - IN
TypeExpression: PropertyIdentifier
Default:ValueExpression:
**Operation PropertyIdenfierList**  
Visibility: 0 - PUBLIC

**Operation PropertyIdenfierList**  
Visibility: 0 - PUBLIC

**Parameter properties**  
Kind: 0 - IN  
TypeExpression: ReadOnlyPropertyIdenfierList  
DefaultValueExpression:

**Operation PropertyIdenfierList**  
Visibility: 3 - PACKAGE

**Parameter xmlProperties**  
Kind: 0 - IN  
TypeExpression: XmlPropertyIdenfier[]  
DefaultValueExpression:

**Operation Remove**  
Visibility: 0 - PUBLIC

**Parameter property**  
Kind: 0 - IN  
TypeExpression: PropertyIdenfier  
DefaultValueExpression:

**Operation RemoveAt**  
Visibility: 0 - PUBLIC

**Parameter index**  
Kind: 0 - IN  
TypeExpression: int  
DefaultValueExpression:

**Class ProviderInfoRequest**

Hypermedia: Microsoft::BusinessFramework::Hypermedia::ProviderInfoRequest  
Stereotype:  
Visibility: 0 - PUBLIC  
Multiplicity:  
Persistence: 0 - PERSISTENT

Documentation: Represents a request for provider information.
**Attribute ElementName**
TypeExpression: string
DefaultValueExpression:

**Attribute GetSupportedHypermediaInfo**
TypeExpression: bool
DefaultValueExpression:

**Attribute XmlConversionType**
TypeExpression: Type
DefaultValueExpression:

**Operation ElementName**
Visibility: 0 - PUBLIC

Parameter string
  Kind: 3 - RETURN
  TypeExpression: string
  DefaultValueExpression:

**Operation FromXmlObject**
Visibility: 1 - PROTECTED

Parameter XmlSerializationBase
  Kind: 0 - IN
  TypeExpression: Xml.SerializationBase
  DefaultValueExpression:

**Operation GetSupportedHypermediaInfo**
Visibility: 0 - PUBLIC

Parameter bool
  Kind: 3 - RETURN
  TypeExpression: bool
  DefaultValueExpression:

**Operation ProviderInfoRequest**
Visibility: 0 - PUBLIC

**Operation ProviderInfoRequest**
Visibility: 0 - PUBLIC

Parameter getSupportedHypermediaInfo
  Kind: 0 - IN
  TypeExpression: bool
  DefaultValueExpression:
**Operation ProviderInfoRequest**
Visibility: 0 - PUBLIC

**Parameter** xmlRepresentation
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

**Operation ProviderInfoRequest**
Visibility: 1 - PROTECTED

**Parameter** xmlRepresentation
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

**Parameter** xmlConversionType
Kind: 0 - IN
TypeExpression: Type
DefaultValueExpression:

**Parameter** elementName
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

**Operation ToXmlObject**
Visibility: 1 - PROTECTED

**Parameter** xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

**Operation XmlConversionType**
Visibility: 1 - PROTECTED

**Parameter** Type
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

Class ProviderInfoResponse
Hypermedia: Microsoft::BusinessFramework::Hypermedia::ProviderInfoResponse
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Represents the response to a request for provider information.

Attribute ElementName
  TypeExpression: string
  DefaultValueExpression:

Attribute SupportedNodeClasses
  TypeExpression: SupportedNodeClassCollection
  DefaultValueExpression:

Attribute Version
  TypeExpression: string
  DefaultValueExpression:

Attribute XmlConversionType
  TypeExpression: Type
  DefaultValueExpression:

Operation ElementName
  Visibility: 0 - PUBLIC

Parameter string
  Kind: 3 - RETURN
  TypeExpression: string
  DefaultValueExpression:

Operation FromXmlObject
  Visibility: 1 - PROTECTED

Parameter xmlSerializationBase
  Kind: 0 - IN
  TypeExpression: Xml.SerializationBase
  DefaultValueExpression:

Operation ProviderInfoResponse
  Visibility: 0 - PUBLIC

Parameter responseCode
  Kind: 0 - IN
  TypeExpression: ResponseCode
  DefaultValueExpression:

Parameter version
  Kind: 0 - IN
  TypeExpression: string
  DefaultValueExpression:
Operation ProviderInfoResponse
Visibility: 0 - PUBLIC

Parameter xmlRepresentation
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

Operation ProviderInfoResponse
Visibility: 1 - PROTECTED

Parameter xmlRepresentation
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

Parameter xmlConversionType
Kind: 0 - IN
TypeExpression: Type
DefaultValueExpression:

Parameter elementName
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

Operation SupportedNodeClasses
Visibility: 0 - PUBLIC

Parameter SupportedNodeClassCollection
Kind: 3 - RETURN
TypeExpression: SupportedNodeClassCollection
DefaultValueExpression:

Operation ToXmlObject
Visibility: 1 - PROTECTED

Parameter xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

Operation Version
Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

Operation XmlConversionType
Visibility: 1 - PROTECTED

Parameter Type
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

Class ProviderListingResponse
Hypermedia: Microsoft.BusinessFramework::Hypermedia::ProviderListingResponse
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Represents a response to a request to retrieve hypermedia.

Attribute ElementName
TypeExpression: string
DefaultValueExpression:

Attribute Providers
TypeExpression: Uri[]
DefaultValueExpression:

Attribute XmlConversionType
TypeExpression: Type
DefaultValueExpression:

Operation ElementName
Visibility: 0 - PUBLIC

Parameter string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

Operation FromXmlObject
Visibility: 1 - PROTECTED

Parameter xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:
**Operation ProviderListingResponse**

Visibility: 0 - PUBLIC

**Parameter** responseCode

Kind: 0 - IN
TypeExpression: ResponseCode
DefaultValueExpression:

**Parameter** providers

Kind: 0 - IN
TypeExpression: Uri[]
DefaultValueExpression:

**Operation ProviderListingResponse**

Visibility: 0 - PUBLIC

**Parameter** xmlRepresentation

Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

**Operation ProviderListingResponse**

Visibility: 1 - PROTECTED

**Parameter** xmlRepresentation

Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

**Parameter** xmlConversionType

Kind: 0 - IN
TypeExpression: Type
DefaultValueExpression:

**Parameter** elementName

Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

**Operation Providers**

Visibility: 0 - PUBLIC

**Parameter** Uri[]

Kind: 3 - RETURN
TypeExpression: Uri[]
DefaultValueExpression:

**Operation ToXmlObject**
Visibility: 1 - PROTECTED

**Parameter xmlSerializationBase**

Kind: 0 - IN  
TypeExpression: Xml.SerializationBase  
DefaultValueExpression:

**Operation XmlConversionType**

Visibility: 1 - PROTECTED

**Parameter Type**

Kind: 3 - RETURN  
TypeExpression: Type  
DefaultValueExpression:

---

**Class ReadOnlyPropertyIdentifierList**

Hypermedia: Microsoft.BusinessFramework::Hypermedia: ReadOnlyPropertyIdentifierList  
Stereotype:  
Visibility: 0 - PUBLIC  
Multiplicity:  
Persistence: 0 - PERSISTENT  
Documentation: A read only version of an `<see cref="PropertyIdentifierList"/>`.

**Operation Clone**

Visibility: 0 - PUBLIC

**Parameter object**

Kind: 3 - RETURN  
TypeExpression: object  
DefaultValueExpression:

**Operation Contains**

Visibility: 0 - PUBLIC

**Parameter bool**

Kind: 3 - RETURN  
TypeExpression: bool  
DefaultValueExpression:

**Parameter property**

Kind: 0 - IN  
TypeExpression: PropertyIdentifier  
DefaultValueExpression:

**Operation IndexOf**

Visibility: 0 - PUBLIC
Parameter int
Kind: 3 - RETURN
TypeExpression: int
DefaultValueExpression:

Parameter property
Kind: 0 - IN
TypeExpression: PropertyIdentifier
DefaultValueExpression:

Operation ReadOnlyPropertyIdentifierList
Visibility: 1 - PROTECTED

Operation ReadOnlyPropertyIdentifierList
Visibility: 0 - PUBLIC

Parameter properties
Kind: 0 - IN
TypeExpression: ReadOnlyPropertyIdentifierList
DefaultValueExpression:

Operation ReadOnlyPropertyIdentifierList
Visibility: 3 - PACKAGE

Parameter xmlProperties
Kind: 0 - IN
TypeExpression: Xml.PropertyIdentifier[]
DefaultValueExpression:

Operation ToXmlObject
Visibility: 3 - PACKAGE

Parameter Xml.PropertyIdentifier[]
Kind: 3 - RETURN
TypeExpression: Xml.PropertyIdentifier[]
DefaultValueExpression:

Class Response
Hypermedia::Microsoft::BusinessFramework::Hypermedia::Response
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: The base class for all responses that are made from <see cref="IHypermediaProvider"/> and the hypermedia service.
**Attribute DiagnosticInfo**
TypeExpression: DiagnosticInfo
DefaultValueExpression:

**Attribute ElementName**
TypeExpression: string
DefaultValueExpression:

**Attribute ResponseCode**
TypeExpression: ResponseCode
DefaultValueExpression:

**Attribute XmlConversionType**
TypeExpression: Type
DefaultValueExpression:

**Operation DiagnosticInfo**
Visibility: 0 - PUBLIC

**Parameter** value
Kind: 0 - IN
TypeExpression: DiagnosticInfo
DefaultValueExpression:

**Operation DiagnosticInfo**
Visibility: 0 - PUBLIC

**Parameter** DiagnosticInfo
Kind: 3 - RETURN
TypeExpression: DiagnosticInfo
DefaultValueExpression:

**Operation ElementName**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation FromXmlObject**
Visibility: 1 - PROTECTED

**Parameter** xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:
**Operation Response**  
Visibility: 0 - PUBLIC

**Parameter** responseCode  
Kind: 0 - IN  
TypeExpression: ResponseCode  
DefaultValueExpression:

**Operation Response**  
Visibility: 0 - PUBLIC

**Parameter** xmlRepresentation  
Kind: 0 - IN  
TypeExpression: XmlElement  
DefaultValueExpression:

**Operation Response**  
Visibility: 1 - PROTECTED

**Parameter** xmlRepresentation  
Kind: 0 - IN  
TypeExpression:XmlElement  
DefaultValueExpression:

**Parameter** xmlConversionType  
Kind: 0 - IN  
TypeExpression: Type  
DefaultValueExpression:

**Parameter** elementName  
Kind: 0 - IN  
TypeExpression: string  
DefaultValueExpression:

**Operation ResponseCode**  
Visibility: 0 - PUBLIC

**Parameter** value  
Kind: 0 - IN  
TypeExpression: ResponseCode  
DefaultValueExpression:

**Operation ResponseCode**  
Visibility: 0 - PUBLIC

**Parameter** ResponseCode  
Kind: 3 - RETURN  
TypeExpression: ResponseCode
DefaultValueExpression:

**Operation ToXmlObject**
Visibility: 1 - PROTECTED

Parameter xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

**Operation XmlConversionType**
Visibility: 1 - PROTECTED

Parameter Type
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

**Enumeration ResponseCode**
Hypermedia::Microsoft::BusinessFramework::Hypermedia::ResponseCode
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Indicates the success/failure of a request to an <see cref="IHypermediaProvider"/>.

**Enumeration Literal Success**
value:

**Enumeration Literal SuccessWithInfo**
value:

**Enumeration Literal UnknownFailure**
value:

**Enumeration Literal InvalidProviderInfoVersion**
value:

**Class SerializationBase**
Hypermedia::Microsoft::BusinessFramework::Hypermedia::SerializationBase
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity: 
Persistence: 0 - PERSISTENT

Documentation: The base class for all requests and responses that are made to/from <see cref="IHypermediaProvider"/> and the hypermedia service.

**Attribute ElementName**

TypeExpression: string
DefaultValueExpression:

**Attribute XmlConversionType**

TypeExpression: Type
DefaultValueExpression:

**Operation DeserializeKey**

Visibility: 3 - PACKAGE

**Parameter** iKey

Kind: 3 - RETURN
TypeExpression: iKey
DefaultValueExpression:

**Parameter** serializedKey

Kind: 0 - IN
TypeExpression: XElement
DefaultValueExpression:

**Operation ElementName**

Visibility: 0 - PUBLIC

**Parameter** string

Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation FromXmlObject**

Visibility: 1 - PROTECTED

**Parameter** xmlSerializationBase

Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

**Operation GetXmlNodeClass**

Visibility: 3 - PACKAGE

**Parameter** XmlNodeClass

Kind: 3 - RETURN
TypeExpression: XmlNodeClass
DefaultValueExpression:
Parameter nodeClass
  Kind: 0 - IN
  TypeExpression: Type
  DefaultValueExpression:

Operation LinkTypeHasSingleValue
  Visibility: 3 - PACKAGE

Parameter bool
  Kind: 3 - RETURN
  TypeExpression: bool
  DefaultValueExpression:

Parameter linkType
  Kind: 0 - IN
  TypeExpression: LinkType
  DefaultValueExpression:

Operation LinkTypeToXmlLinkType
  Visibility: 3 - PACKAGE

Parameter XmlLinkType
  Kind: 3 - RETURN
  TypeExpression: XmlLinkType
  DefaultValueExpression:

Parameter linkType
  Kind: 0 - IN
  TypeExpression: LinkType
  DefaultValueExpression:

Operation SerializationBase
  Visibility: 1 - PROTECTED

Operation SerializationBase
  Visibility: 1 - PROTECTED

Parameter xmlRepresentation
  Kind: 0 - IN
  TypeExpression: XmlElement
  DefaultValueExpression:

Parameter xmlConversionType
  Kind: 0 - IN
  TypeExpression: Type
  DefaultValueExpression:

Parameter elementName
  Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

Operation SerializeKey
Visibility: 3 - PACKAGE

Parameter XmlElement
Kind: 3 - RETURN
TypeExpression: XmlElement
DefaultValueExpression:

Parameter key
Kind: 0 - IN
TypeExpression: IKey
DefaultValueExpression:

Operation ToXml
Visibility: 0 - PUBLIC

Parameter XmlElement
Kind: 3 - RETURN
TypeExpression: XmlElement
DefaultValueExpression:

Operation ToXml
Visibility: 0 - PUBLIC

Parameter XmlElement
Kind: 3 - RETURN
TypeExpression: XmlElement
DefaultValueExpression:

Parameter parentNode
Kind: 0 - IN
TypeExpression: XmlNode
DefaultValueExpression:

Operation ToXmlObject
Visibility: 1 - PROTECTED

Parameter xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

Operation XmlConversionType
Visibility: 1 - PROTECTED
Parameter Type
   Kind: 3 - RETURN
   TypeExpression: Type
   DefaultValueExpression:

Operation XmlLinkTypesToLinkType
   Visibility: 3 - PACKAGE

Parameter LinkType
   Kind: 3 - RETURN
   TypeExpression: LinkType
   DefaultValueExpression:

Parameter xmlLinkType
   Kind: 0 - IN
   TypeExpression: XmlLinkType
   DefaultValueExpression:

Class XmlSerializerFactory
   Hypermedia: Microsoft::BusinessFramework::Hypermedia::SerializationBase::XmlSerializerFa
   Stereotype: Class
   Visibility: 2 - PRIVATE
   Multiplicity:
   Persistence: 0 - PERSISTENT

Documentation: The XmlSerializerFactory is used to create <see cref="XmlSerializer"/> objects
for the specified <see cref="Type"/>.

Attribute Serializer
   TypeExpression: XmlSerializer
   DefaultValueExpression:

Attribute Type
   TypeExpression: Type
   DefaultValueExpression:

Operation Create
   Visibility: 0 - PUBLIC

Operation Serializer
   Visibility: 0 - PUBLIC

Parameter XmlSerializer
   Kind: 3 - RETURN
   TypeExpression: XmlSerializer
   DefaultValueExpression:
**Operation Type**
Visibility: 0 - PUBLIC

**Parameter Type**
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

**Operation XmlSerializerFactory**
Visibility: 0 - PUBLIC

**Parameter type**
Kind: 0 - IN
TypeExpression: Type
DefaultValueExpression:

**Enumeration SerializationType**
Hypermedia:Microsoft:BusinessFramework::Hypermedia:SerializationType
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Identifies a serialization method.

**Enumeration Literal SoapFormatter**
value:

**Enumeration Literal TypeConverter**
value:

**Class SupportedNodeClass**
Hypermedia:Microsoft:BusinessFramework::Hypermedia::SupportedNodeClass
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Identifies what types of hypermedia an <see cref="IHypermediaProvider"/> supports for a node class.

**Attribute LinkCategory**
TypeExpression: string
DefaultValueExpression:

**Attribute LinkType**
TypeExpression: LinkType
DefaultValueExpression:

**Attribute NodeClass**
TypeExpression: Type
DefaultValueExpression:

**Operation Equals**
Visibility: 0 - PUBLIC

**Parameter bool**
Kind: 3 - RETURN
TypeExpression: bool
DefaultValueExpression:

**Parameter obj**
Kind: 0 - IN
TypeExpression: object
DefaultValueExpression:

**Operation GetHashCode**
Visibility: 0 - PUBLIC

**Parameter int**
Kind: 3 - RETURN
TypeExpression: int
DefaultValueExpression:

**Operation LinkCategory**
Visibility: 0 - PUBLIC

**Parameter string**
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation LinkType**
Visibility: 0 - PUBLIC

**Parameter LinkType**
Kind: 3 - RETURN
TypeExpression: LinkType
DefaultValueExpression:

**Operation NodeClass**
Parameter Type
   Kind: 3 - RETURN
   TypeExpression: Type
   DefaultValueExpression:

Operation SupportedNodeClass
   Visibility: 0 - PUBLIC

Parameter nodeClass
   Kind: 0 - IN
   TypeExpression: Type
   DefaultValueExpression:

Parameter linkType
   Kind: 0 - IN
   TypeExpression: LinkType
   DefaultValueExpression:

Parameter linkCategory
   Kind: 0 - IN
   TypeExpression: string
   DefaultValueExpression:

Operation SupportedNodeClass
   Visibility: 3 - PACKAGE

Parameter xmlSupportedNodeClass
   Kind: 0 - IN
   TypeExpression: XmlSupportedNodeClass
   DefaultValueExpression:

Operation ToXmlObject
   Visibility: 3 - PACKAGE

Parameter XmlSupportedNodeClass
   Kind: 3 - RETURN
   TypeExpression: XmlSupportedNodeClass
   DefaultValueExpression:

Class SupportedNodeClassCollection
   Hypermedia: Microsoft::BusinessFramework::Hypermedia::SupportedNodeClassCollection
   Stereotype: 
   Visibility: 0 - PUBLIC
   Multiplicity: 
   Persistence: 0 - PERSISTENT
Documentation: Represents a collection of <see cref="SupportedNodeClass"/> objects.

**Operation Add**
Visibility: 0 - PUBLIC

**Parameter** int
Kind: 3 - RETURN
TypeExpression: int
DefaultValueExpression:

**Parameter** supportedNodeClass
Kind: 0 - IN
TypeExpression: SupportedNodeClass
DefaultValueExpression:

**Operation Clear**
Visibility: 0 - PUBLIC

**Operation Clone**
Visibility: 0 - PUBLIC

**Parameter** object
Kind: 3 - RETURN
TypeExpression: object
DefaultValueExpression:

**Operation Contains**
Visibility: 0 - PUBLIC

**Parameter** bool
Kind: 3 - RETURN
TypeExpression: bool
DefaultValueExpression:

**Parameter** supportedNodeClass
Kind: 0 - IN
TypeExpression: SupportedNodeClass
DefaultValueExpression:

**Operation IndexOf**
Visibility: 0 - PUBLIC

**Parameter** int
Kind: 3 - RETURN
TypeExpression: int
DefaultValueExpression:

**Parameter** supportedNodeClass
Kind: 0 - IN
Operation Insert
Visibility: 0 - PUBLIC

Parameter index
Kind: 0 - IN
TypeExpression: int
DefaultValueExpression:

Parameter supportedNodeClass
Kind: 0 - IN
TypeExpression: SupportedNodeClass
DefaultValueExpression:

Operation Remove
Visibility: 0 - PUBLIC

Parameter supportedNodeClass
Kind: 0 - IN
TypeExpression: SupportedNodeClass
DefaultValueExpression:

Operation RemoveAt
Visibility: 0 - PUBLIC

Parameter index
Kind: 0 - IN
TypeExpression: int
DefaultValueExpression:

Operation SupportedNodeClassCollection
Visibility: 0 - PUBLIC

Operation SupportedNodeClassCollection
Visibility: 0 - PUBLIC

Parameter supportedNodeClasses
Kind: 0 - IN
TypeExpression: SupportedNodeClassCollection
DefaultValueExpression:

Operation SupportedNodeClassCollection
Visibility: 3 - PACKAGE

Parameter xmISupportedNodeClasses
Kind: 0 - IN
TypeExpression: XmlSupportedNodeClass[]
DefaultValueExpression:

**Operation ToXmlObject**
Visibility: 3 - PACKAGE

**Parameter** XmlSupportedNodeClass[]
Kind: 3 - RETURN
TypeExpression: XmlSupportedNodeClass[]
DefaultValueExpression:

---

**Class TraverseRequest**
Hypermedia::Microsoft::BusinessFramework::Hypermedia::TraverseRequest
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Represents a request to traverse a `<see cref="Link"/>`.

**Attribute ElementName**
TypeExpression: string
DefaultValueExpression:

**Attribute Key**
TypeExpression: IKey
DefaultValueExpression:

**Attribute Link**
TypeExpression: Link
DefaultValueExpression:

**Attribute XmlConversionType**
TypeExpression: Type
DefaultValueExpression:

**Operation ElementName**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation FromXmlObject**
Visibility: 1 - PROTECTED
Parameter xmlSerializationBase
   Kind: 0 - IN
   TypeExpression: Xml.SerializationBase
   DefaultValueExpression:

Operation Key
   Visibility: 0 - PUBLIC

Parameter iKey
   Kind: 3 - RETURN
   TypeExpression: iKey
   DefaultValueExpression:

Operation Link
   Visibility: 0 - PUBLIC

Parameter Link
   Kind: 3 - RETURN
   TypeExpression: Link
   DefaultValueExpression:

Operation ToXObject
   Visibility: 1 - PROTECTED

Parameter xmlSerializationBase
   Kind: 0 - IN
   TypeExpression: Xml.SerializationBase
   DefaultValueExpression:

Operation TraverseRequest
   Visibility: 0 - PUBLIC

Parameter link
   Kind: 0 - IN
   TypeExpression: Link
   DefaultValueExpression:

Parameter key
   Kind: 0 - IN
   TypeExpression: iKey
   DefaultValueExpression:

Operation TraverseRequest
   Visibility: 0 - PUBLIC

Parameter xmlRepresentation
   Kind: 0 - IN
Operation TraverseRequest
Visibility: 1 - PROTECTED

Parameter xmlRepresentation
Kind: 0 - IN
TypeExpression: XmlElement
DefaultValueExpression:

Parameter xmlConversionType
Kind: 0 - IN
TypeExpression: Type
DefaultValueExpression:

Parameter elementName
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

Operation XmlConversionType
Visibility: 1 - PROTECTED

Parameter Type
Kind: 3 - RETURN
TypeExpression: Type
DefaultValueExpression:

Class TraverseResponse
Hypermedia::Microsoft::BusinessFramework::Hypermedia::TraverseResponse
Stereotype:
Visibility: 0 - PUBLIC
Multiplicity:
Persistence: 0 - PERSISTENT

Documentation: Represents the response to a request to traverse a &lt;see cref="Link"&gt;.

Attribute DestinationNode
TypeExpression: object
DefaultValueExpression:

Attribute DisplayMessage
TypeExpression: string
DefaultValueExpression:

Attribute ElementName
TypeExpression: string
DefaultValueExpression:

**Attribute SerializationType**
TypeExpression: SerializationType
DefaultValueExpression:

**Attribute XmlConversionType**
TypeExpression: Type
DefaultValueExpression:

**Operation DestinationNode**
Visibility: 0 - PUBLIC

**Parameter** object
Kind: 3 - RETURN
TypeExpression: object
DefaultValueExpression:

**Operation DisplayMessage**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation ElementName**
Visibility: 0 - PUBLIC

**Parameter** string
Kind: 3 - RETURN
TypeExpression: string
DefaultValueExpression:

**Operation FromXmlObject**
Visibility: 1 - PROTECTED

**Parameter** xm serializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

**Operation SerializationType**
Visibility: 0 - PUBLIC

**Parameter** SerializationType
Kind: 3 - RETURN
TypeExpression: SerializationType
DefaultValueExpression:

Operation ToXmlObject
Visibility: 1 - PROTECTED

Parameter xmlSerializationBase
Kind: 0 - IN
TypeExpression: Xml.SerializationBase
DefaultValueExpression:

Operation TraverseResponse
Visibility: 0 - PUBLIC

Parameter responseCode
Kind: 0 - IN
TypeExpression: ResponseCode
DefaultValueExpression:

Parameter displayMessage
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:

Operation TraverseResponse
Visibility: 0 - PUBLIC

Parameter responseCode
Kind: 0 - IN
TypeExpression: ResponseCode
DefaultValueExpression:

Parameter destinationNode
Kind: 0 - IN
TypeExpression: object
DefaultValueExpression:

Parameter serializationType
Kind: 0 - IN
TypeExpression: SerializationType
DefaultValueExpression:

Parameter displayMessage
Kind: 0 - IN
TypeExpression: string
DefaultValueExpression:
Parameter xmlRepresentation
   Kind: 0 - IN
   TypeExpression: XmlElement
   DefaultValueExpression:

Operation TraverseResponse
Visibility: 1 - PROTECTED

Parameter xmlRepresentation
   Kind: 0 - IN
   TypeExpression: XmlElement
   DefaultValueExpression:

Parameter xmlConversionType
   Kind: 0 - IN
   TypeExpression: Type
   DefaultValueExpression:

Parameter elementName
   Kind: 0 - IN
   TypeExpression: string
   DefaultValueExpression:

Operation XmlConversionType
Visibility: 1 - PROTECTED

Parameter Type
   Kind: 3 - RETURN
   TypeExpression: Type
   DefaultValueExpression:

Package References
   Hypermedia::References
   Stereotype: modelReferences
Appendix F

Metamodel Hypermedia Provider Design

Pages 1-17
Metamodel Hypermedia Provider Design

Subsystem Overview and Design

1. Introduction

Problem Statement

Applications built upon the .NET Business Platform define Entities to represent the business objects that they work with in their problem domain. Examples of these Entities are Customer, Order, Vendor, Sales Person, Employee, Timecard, etc. These Entities are interconnected through different associations. For instance, an Order has both a Customer and a Sales Person associated with it. Since these associations exist in the problem domain of the application, they are associations that the end user typically understands. Thus it would be beneficial to allow the end user to navigate via these associations.

The information that defines these associations is captured in the metamodel (object model) of the applications being developed. In the .NET Business Framework this metadata is available via the Metadata Subsystem. Also, the Hypermedia Subsystem exists as a means of propagating this information up to the applications that wish to display it.

There must be a layer between the Metadata Subsystem and the Hypermedia Subsystem that converts the metadata into a format that the Hypermedia Subsystem understands. To provide this information to the Hypermedia Subsystem a hypermedia provider must be created, which supplies the information in the form of links. This layer is known as the Metamodel Hypermedia Provider.

Document Scope

This document is an overview of the Metamodel Hypermedia Provider as a deliverable of the Iceberg project.

This document considers only the hypermedia being provided by the Metamodel Hypermedia Provider; no other hypermedia providers will be discussed.

For more information about the Iceberg project see the Iceberg Software Requirements and the Glider Website. For more information on the Hypermedia Subsystem see the Hypermedia Subsystem Design.

Subsystem Description

High Level Overview

The Metamodel Hypermedia Provider is responsible for analyzing the association information contained in the Metadata Subsystem, and providing hypermedia to the HypermediaService to represent these associations. This allows clients of the HypermediaService to retrieve this hypermedia, and make it available to the end user, which, in turn, allows the end user to navigate the metamodel.

One of the benefits to the end user of navigating the metamodel is that they are able to view information related to the Entity instance they are currently viewing. For instance, if the end user is viewing Customer Aaron Fitz, and they navigate the association to Orders, they would only see the Orders for Aaron Fitz. This means that the hypermedia that the Metamodel Hypermedia Provider provides, is node based (i.e. the hypermedia depends on which instance of the Entity they are viewing).

The Hypermedia Subsystem Design states that there are two types of hypermedia that apply to instances. These types are referred to as Instance hypermedia and Instance Specific hypermedia. Instance hypermedia is hypermedia which applies to (is available for) any instance of a given class. Instance Specific hypermedia is hypermedia which applies to (is available for) only a subset of the instances of a given class. In the case of the Metamodel Hypermedia Provider, all of the hypermedia it provides falls into the first group – Instance hypermedia. Since the association between two Entities exists regardless of which instance of the Entity is in question, the hypermedia based on that association applies to each instance of the Entity. However, this does not mean that the query that is the destination of the hypermedia will contain any results. For example, the association, and hence the hypermedia, between Customers and Orders exists for each Customer, but it is conceivable that a Customer exists for which there are no Orders.

Definitions

Node
A class instance. For the Metamodel provider, a source node is an instance of an Entity class held by the client. A destination node is an Explorer Query based on the destination Entity Class from a UML association in the object model.

Restricting Property
A key property of the destination class of an association. This property is used to restrict queries based on the source class.

Provider Information Version
A value that identifies a specific version of the object model from which the Metamodel Provider's hypermedia is generated.

For additional terminology definitions, see the Bedrock Glossary and the Hypermedia Subsystem Design.

References
Entity Security Policy (Dynamic Views) Subsystem Design
Glacier Website
Hypermedia Subsystem Design
Inferno Software Requirements
Query Subsystem Design
Metadata Subsystem Design

2. Use Cases, Scenarios and Requirements

Actors

Hypermedia Service
The HypermediaService is the primary class used by clients of the Hypermedia Subsystem to retrieve hypermedia.

The install Process and Admin Console register hypermedia providers with the HypermediaService. The HypermediaService maintains (persists and caches) a list of all registered hypermedia providers (implementers of the IHypermediaProvider interface). (For a description of how hypermedia providers are registered with the HypermediaService, refer to the Hypermedia Subsystem Design document. Hypermedia providers do not actively participate in their own registration. Consequently, no use case for this activity will be included in this document.)

The HypermediaService is also responsible for fulfilling requests for hypermedia. When requests are made upon the HypermediaService to retrieve hypermedia, the HypermediaService determines which providers can supply hypermedia to fulfill the request. It then retrieves the hypermedia from these providers and aggregates it. The aggregated collection of hypermedia is then returned to the client.

Metadata Subsystem
The Metadata Subsystem actor represents the subsystem that provides interfaces to retrieve a list of queriable Business Entities, a list of queriable properties for a specified Entity,

The Metadata Subsystem can provide a list of all associations defined in the UML model for a specified class. Associations belong to one of four types: Simple, Composition, Inheritance, and AssociationClass. The Metadata Subsystem will return Simple, Composition, Inheritance, and Association Class associations. For a specified class, Metadata will provide associations for not only the specified class, but provide associations inherited from its parent classes as well. For each association, Metadata will supply the associated class name, the association keys (properties of the source class that reference instances of the destination class), and the association type.

Security Subsystem
The Security Subsystem actor represents the subsystem that is responsible for administrating security information. The security subsystem will provide a means of determining the current user's Dynamic Views.

Query Subsystem
The Query Subsystem actor represents the subsystem that the Metamodel Provider uses to create the queries that will result from link traversals.
Metamodel Provider Information Use Cases

The Metamodel Provider Information Use Cases diagram depicts the use cases that the Metamodel Hypermedia Provider must realize in relation to fulfilling requests for provider information.

Use Case Descriptions

Add New Association

Overview:
This use case applies to Phase II. When a new association is added to the Metadata Subsystem, it will fire an event that can be handled by the Metamodel Hypermedia Provider. The Metamodel Provider will persist the time at which this event occurred as the Provider Information Version. (See the Get Provider Info and Get Hypermedia use cases for more details on how the ProviderInfoVersion is used to satisfy requirement 3.3.)

Starting Point:
Information about a new association has been added to the Metadata Subsystem.

Ending Point:
The Metamodel Hypermedia Provider knows about the object model changes, and has modified ProviderInfoVersion accordingly.

Measurable Result:
The ProviderInfoVersion has been updated and persisted to reflect the new object model changes. (The provider will return error status and diagnostic information in its response to hypermedia requests based upon stale object model information.)

Flow of Events:
The Metamodel Hypermedia Provider handles an update event from the Metadata subsystem. The Metamodel Hypermedia Provider persists the time at which this event was fired as the ProviderInfoVersion.

See Also:
The ProviderInfoVersion affects the Get Provider Info and Get Hypermedia use cases and is used to satisfy requirement 3.3.

Requirements:
This use case satisfies requirements 2.2 and 3.3.

Remove Associations
Overview:
This use case applies to Phase II. When an association is removed from the Metadata Subsystem, it will fire an update event that can be handled by the Metamodel Hypermedia Provider. The Metamodel Provider will persist the time at which this event occurred as the Provider Information Version.

Starting Point:
Information about an association has been removed from the Metadata Subsystem.

Ending Point:
The Metamodel Hypermedia Provider knows an object model change has occurred, and has changed its ProviderInfoVersion accordingly.

Measurable Result:
The Metamodel Hypermedia Provider knows about the change and has updated the ProviderInfoVersion to the current time.

Flow of Events:
The Metamodel Hypermedia Provider receives an event from the Metadata Subsystem indicating that an object model change has occurred. The Metamodel Provider persists the time at which the event was handled as the ProviderInfoVersion.

See Also:
Same as Add New Association.

Requirements:
This use case satisfies requirements 2.3 and 3.3.

Get Provider Info
Overview:
The HypermediaService needs to know the provider information (link category, link types, supported node classes, last update time) provided by a given hypermedia provider. In the case of the Metamodel Provider, the list of supported node classes includes all entities with the following associations (the association must also be marked as Hypermedia Navigable):
- Simple associations, regardless of whether the association is navigable from the source class.
- Composition associations.
- Inheritance associations in which the source class is the parent.

Association class associations are not supported. The many-many relationship, which can be modeled with the Association Class association, is not supported directly by Metadata. However, many-many relationships are supported as a pair of 1:n simple associations. Only entities and associations that are explorables are returned in the list of source node classes.

Starting Point:
The HypermediaService makes a request to the Metamodel Hypermedia Provider for its provider information.

Ending Point:
The Metamodel Hypermedia Provider has returned the provider information to the HypermediaService.

Measurable Result:
The provider information supported by the Metamodel Hypermedia Provider has been returned to the HypermediaService.

Flow of Events:
The HypermediaService requests the Metamodel Hypermedia Provider for its provider information. The Metamodel Provider compiles a response for the HypermediaService containing the following information:
- The node classes which are supported. The Metamodel Hypermedia Provider gets a list of entities with associations which are marked as Hypermedia Navigable from the Metadata Subsystem. It then creates a SupportedNodeClass instance for each of these entities and adds it to the list of supported node classes in the provider info response.
Each SupportedNodeClass created by the Metamodel Provider will have a link category of Microsoft.BusinessFramework.Metamodel.
Each SupportedNodeClass created by the Metamodel Provider will have a link type of Hypermedia.LinkType.Instance.

- The Provider Information Version. The HypermediaService will cache (not persist) this version along with the node classes supported by the provider. The HypermediaService will then include the version with subsequent requests for hypermedia. The MetamodelProvider compares the version accompanying the request with its persisted object model version to determine if the client request is based on stale provider information. (For more information on how the ProviderInfoVersion is used to satisfy requirement 3.3, refer to the Get Hypermedia use case.)

Requirements:
This use case satisfies requirement 3.3.
Metamodel Provider Hypermedia Retrieval Use Cases

The Metamodel Provider Registration Use Cases diagram depicts the use cases that the Metamodel Hypermedia Provider must realize in relation to hypermedia retrieval.

Use Case Descriptions

Get Hypermedia

Overview:
The retrieval of hypermedia is part of the IHypermediaProvider interface, which the Metamodel Hypermedia Provider must realize.

- The Metamodel Hypermedia Provider only supports requests for Instance links and will return an empty response with a response code of SuccessWithInfo if it receives a request that does not request Instance hypermedia.
- The Metamodel Hypermedia Provider fulfills hypermedia requests with a collection of links.
- The hypermedia request will be accompanied by the Provider Information Version. The provider will compare the version accompanying the request with its internal version. If the provider determines that the HypermediaService's request is based on stale provider information, it will return an empty response and error information.
- The Metamodel Hypermedia Provider restricts the hypermedia returned based on the user's Dynamic Views. Even though the source class may have a linkable association to another class, a link will not be generated if the user does not have a Dynamic View for the destination class. If the user has more than one Dynamic View for a destination class and the Metamodel Provider cannot determine which view to use, it will return a link for each view.
- The link display name will be derived from the association name and the Dynamic View name (e.g., AssociationName(Viewname)).
Starting Point:
The HypermediaService makes a request for hypermedia.

Ending Point:
A HypermediaResponse containing the requested hypermedia has been returned to the HypermediaService.

Measurable Result:
A HypermediaResponse containing the requested hypermedia has been returned to the HypermediaService.

Flow of Events:
A request is made by the HypermediaService to retrieve Instance links for a given class. The provider uses the Metadata Subsystem to determine which Entities are associated with the one specified in the request. The provider examines each association and determines if the user has a Dynamic View for the association's destination class. (The provider won't supply links to entities for which the user does not have the rights to build a query.) The provider then creates a link for each association for which the user has access to a (destination Entity) Dynamic View, and places association information in the link. If the user has more than one Dynamic View for an associated class, the provider supplies links for each view. The association information is also used to fill the AssociatedProperties member of the link. As they are created, the links are added to the HypermediaResponse.

Alternate Flow of Events:
The Metamodel Hypermedia Provider receives a hypermedia request from the HypermediaService for a node class that it does not support. A HypermediaResponse with no links and a response code of UnknownFailure is returned to the HypermediaService.

Alternate Flow of Events:
The HypermediaService makes a request which does not request Instance links. A HypermediaResponse with no links and a response code of SuccessWithInfo is returned to the HypermediaService.

Alternate Flow of Events:
The HypermediaService makes a request based on a stale Provider Information Version. A HypermediaResponse with no links and a response code of InvalidProviderInfoVersion is returned to the HypermediaService. This serves to notify the HypermediaService of the need to re-request provider information before proceeding to ensure its requests are consistent with the current object model.

Alternate Flow of Events:
If source class is the parent in one or more inheritance associations, a single "ShowDetails" link will be added to the response. The client displays a "ShowDetails" link based on the existence of this link, indicating the option to "zoom"/"drill down" for more specific information (additional columns) available when the source node is viewed as its most-derived class.

Requirements:
This use case satisfies requirement 2.1, 3.3, and 4.3-4.6.

Traverse Link

Overview:
Links which were retrieved from the Metamodel Hypermedia Provider can be traversed by the Metamodel Hypermedia Provider. Traversal of a link will return a traversal result containing the destination node of the link. In the case of the Metamodel Hypermedia Provider the destination node will always be a Query.

Starting Point:
The Metamodel Hypermedia Provider receives a request to traverse a link.

Ending Point:
The link has been traversed and a traversal result has been returned.

Measurable Result:
A traversal result has been returned.

Flow of Events:
The Metamodel Hypermedia Provider receives a request to traverse a link. It examines the information that is contained in the link to determine which Dynamic View to use. A Query is then created based off of that Dynamic View. A restriction is added to the Query to limit the results to those that pertain to the association. For example, if the association is between the Customers and Orders classes, the Query on Orders would be limited to the set of Orders for that Customer instance. The appropriate properties are added to the Query's SelectedProperties. A traversal result is created, and the Query is added to it. The traversal result is then returned.

Requirements:
This use case addresses requirements 4.1-4.6.
3. Constraints (optional)
None documented at this time.

4. Decomposition

Metamodel Hypermedia Provider Architecture

Overview
The Metamodel Hypermedia Provider Architecture diagram below gives an overview of the classes that comprise the Metamodel Hypermedia Provider.

The Metamodel Hypermedia Provider Detailed Design diagram shows a detailed view of the classes that comprise the Metamodel Hypermedia Provider. This includes internal helper classes used by the Metamodel Hypermedia Provider.
Dependencies on other Subsystems

The Metamodel Hypermedia Provider has the following dependencies on other subsystems and namespaces.
Hypermedia
The Hypermedia namespace contains the basic hypermedia classes (e.g., Link), as well as classes that serialize and deserialize the XmlDocument parameters that accompany IHypermediaProvider method calls. These classes include:
- Link — represents a hyperlink.
- ProviderInfoRequest, HypermediaRequest, TraverseRequest — deserialize the XmlDocument parameters accompanying IHypermediaProvider requests and exposes their elements as properties.
- ProviderInfoResponse, HypermediaResponse, TraverseResponse — serialize results of IHypermediaProvider methods as XmlDocuments.

MetaData
The Resources namespace contains the Resources subsystem that provides the Metamodel Provider with functionality for extracting association information from the object model. The Metamodel Provider uses this information to generate hyperlinks.
- Association — provides association information for a single object model entity.
- Associations — provides information for a single UML association.
- GetHyperNavigableEntities — provides information about those Entities from the object model that have linkable associations.

Query
The Query namespace contains the Query Subsystem that provides the functionality to build the query objects that are the destination nodes for Metamodel Provider link traversal requests. The classes include:
- Query — represents an explorer query.
- Restriction — represents a query restriction that generates a where clause.
- Enumerations and classes that define restriction operators and token types.

Security Subsystem
The Security namespace provides the functionality (the DynamicView class) to retrieve the current user's DynamicViews for a specified entity. The Metamodel Provider uses these views to restrict the set of hypermedia it returns.

See Also:
Classes

The Metamodel Hypermedia Provider object model is fairly simple, consisting of the following classes:

**MetamodelProviderWebService**

The MetamodelProviderWebService implements the IHypermediaProvider interface as web methods. The MetamodelProviderWebService contains an instance of the MetamodelProvider class. The MetamodelWebService's web method implementations of IHypermediaProvider delegate to the IHypermediaProvider methods of the contained MetamodelProvider instance.

The MetamodelProviderWebService service is consumed by the HypermediaService. The HypermediaService aggregates results generated by the MetamodelProviderWebService with results generated by other providers.


**MetamodelProvider**

MetamodelProvider is the class that implements the IHypermediaProvider interface and the actual functionality for the Metamodel Hypermedia Provider. The MetamodelProvider class fulfills requests for hypermedia for the Microsoft Business Framework Metamodel link category (i.e., its methods only return results for this category).

**RootInfo**

The RootInfo class is used to determine what the possible Root Entities of a given Entity are.

**MetamodelLinkProviderData**

MetamodelLinkProviderData manages the serialization and deserialization of Metamodel-provider-specific information to/from the ProviderData XmlDocument member of the Link class. The Metamodel Provider uses ProviderData to roundtrip security and Query information. When a link is created in a call to GetHypermedia, information regarding the Dynamic View for which the link was created and other necessary information to create the proper TraverseInfo are added to ProviderData. This information is then available to the provider when TraverseLink is called.

Outstanding Issues:

1. Isn't serialization of Entity keys a common operation, not limited to the Metamodel provider? If so, shouldn't the functionality be defined outside of the Metamodel provider's namespace?

   **Status:** Yes. The EntityKeySerializer class, currently defined in the Metamodel provider's namespace, will eventually be replaced with shared functionality, once this has been defined.

   **Resolution:** The ClassKey (EntityKey's base class) along with the KeyUtil class implement the Key serialization functionality.

5. Responsibility Mapping

Use Case Realizations

Get Provider Info
This sequence diagram realizes the Get Provider Info use case. It demonstrates how a client request for the types of hypermedia supported is fulfilled.
Get Hypermedia

- This sequence diagram realizes the Get Hypermedia use case. It describes how client requests for hypermedia are fulfilled through the generation of links.
Traverse Link

- The sequence diagram realizes the Traverse Link use case. This use case describes how client requests for link traversal are satisfied through the building of a query.

6. Subsystem Interface

The Metamodel Provider is accessed through the IHypermediaProvider interface. This interface is defined in the Hypermedia subsystem.
7. Design Overview

Mapping Restrictions to Key Properties

The destination node returned for all Metamodel Provider link traversals is an Explorer Query. The restrictions on these queries are built by setting key properties equal to specific values (analogous to setting the key field on a table equal to a specific value in an SQL where clause). For a multi-segment key, the restriction contains one equal statement per segment. The values used in these restrictions come from the instance values of the source node class' key properties, and are passed to the provider's TraverseLink method via an EntityKey. The relationship between the association keys and the restrictions generated are illustrated in the following examples.

- Simple Associations

  ![Diagram of Simple Associations]

  - Simple 1:1 Association
    - Navigable (in the UML sense) from Source Class (source class is Alpha)
      - Restriction (Beta.Key.ID1 == Alpha.betaKey.ID1) & (Beta.Key.ID2 == Alpha.betaKey.ID2)
      - Not UML Navigable from Source Class (source class is Beta)
        - Restriction (Alpha.betaKey.ID1 == Beta.Key.ID1) & (Alpha.betaKey.ID2 == Beta.Key.ID2)
  - Simple n:1 Association
    - UML Navigable from Source Class (source class is Omega)
      - Restriction (Omicron.Key.ID1 == Omega.omicronKey.ID1) & (Omicron.Key.ID2 == Omega.omicronKey.ID2)
    - Not UML Navigable from Source Class (source class is Omicron)
      - Restriction (Omicron.omicronKey.ID1 == Omicron.Key.ID1) & (Omicron.omicronKey.ID2 == Omicron.Key.ID2)

- Composition Associations

  ![Diagram of Composition Associations]

  - 1:1 Composition Association
    - Source Class is Sigma (the Client/Aggregate Class)
      - Restriction: (Sigma.Key.ID1 == Sigma.Key.ID1) & (Sigma.Key.ID2 == Sigma.Key.ID2)
      - Query. SelectedProperties will contain (only) Tau properties. (i.e., our query should only contain the columns aggregated from Tau.)
    - Source Class is Tau (the Supplier/Contained Class)
      - Restriction: (Sigma.Key.ID1 == Sigma.Key.ID1) & (Sigma.Key.ID2 == Sigma.Key.ID2)
Query.SelectedProperties will contain (only) Sigma properties. (i.e., our query should only contain the columns from Sigma.)

- 1-n Composition Associations
  - Source Class is Gamma (the Client/Aggregate Class)
    - Restriction: (Gamma.Key.ID1 == Gamma.Key.ID1) && (Gamma.Key.ID2 == Gamma.Key.ID2) && (Gamma.Key.ID3 == Gamma.Key.ID3)
    - Query.SelectedProperties will contain (only) Delta properties. (i.e., our query should only contain the columns aggregated from Delta.)
  - Source Class is Delta (the Supplier/Contained Class)
    - Restriction: (Gamma.Key.ID1 == Gamma.Key.ID1) && (Gamma.Key.ID2 == Gamma.Key.ID2) && (Gamma.Key.ID3 == Gamma.Key.ID3)
    - Query.SelectedProperties will contain (only) Gamma properties. (i.e., our query should only contain the columns from Gamma.)

- Association Class Association
  The association class association UML relationship, illustrated in the diagram below, is not directly supported by the Metamodel provider.

```
+EpsilonKey <<key>> +PhiKey
Epsilon

Phi

EpsilonPhi
```

The many-to-many relationship, which the association class represents, is supported by the Metamodel provider through a pair of simple associations, as shown below.

```
1
+etaKey <<key>> +nuKey
1

Nu

NuKey

Nu.ID

EtaNu
```

- Source class is Eta
  - Destination Class: EtaNu
  - Restriction: (EtaNu.etaKey.ID1 == Eta.Key.ID1) && (EtaNu.etaKey.ID2 == Eta.Key.ID2)
- Source class is Nu
  - Destination Class: EtaNu
  - Restriction: (EtaNu.nuKey.ID1 == Nu.Key.ID)
- Source class is EtaNu
  - There are two associations from EtaNu and therefore two possible hyperlinks, one to Eta the other to Nu. We will examine the restrictions for the link to destination Eta in this example.
  - Restriction: (Eta.Key.ID1 == EtaNu.etaKey.ID1) && (Eta.Key.ID2 == EtaNu.etaKey.ID2)

- Inheritance Association
  - Source Class is parent
    - A show "ShowDetails" link is generated to indicate that there exists additional properties/detail.
  - Source Class is child
    - No link. The metamodel provider does not provide links to navigate inheritance associations to more abstract classes (this procedure would simply filter out properties).
What is claimed is:

1. A system for supplying links between objects comprising:
   a link service receiving a link request from a client, the
   link request identifying a source object, the link service
   aggregating links from link providers for which the
   source object is a source of the links and providing the
   aggregated links to the client.

2. The system of claim 1 wherein the link request identifies
   an instance of the source object and wherein the link
   service aggregates the links based on the instance of the
   source object.

3. The system of claim 1 wherein the link request identifies
   the class of the source object and wherein the link
   service aggregates the links based on the class of the source
   object.

4. The system of claim 1 and further comprising:
   a link provider operably coupled to the link service, the
   link service requesting links having the source object as
   the source of the links from the link provider.

5. The system of claim 4 and further comprising:
   a provider register, operably coupled to the link service,
   storing register information corresponding to the
   provider, the register information being indicative of links
   provided by the link provider.

6. The system of claim 5 wherein the register information is
   indicative of source objects for which the link provider
   provides links.

7. The system of claim 6 wherein the register information is
   indicative of classes of objects for which the link provider
   provides links.

8. The system of claim 6 wherein the register information is
   indicative of instances of objects for which the link
   provider provides links.

9. The system of claim 6 and further comprising:
   a plurality of link providers, the provider register storing
   register information for each of the plurality of link
   providers.

10. The system of claim 9 wherein the link service accesses
    the provider register to identify link providers from
    which to request links, based on the identified source object.

11. The system of claim 10 wherein the link service
    is configured to receive a traversal request from the client,
    the traversal request identifying a link to traverse.

12. The system of claim 11 wherein the link service
    is configured to determine which link provider provided
    the link and to request that link provider to traverse the link,
    the link provider returning a traversal result.

13. The system of claim 12 wherein the link represents an
    association with a destination object and wherein the
    traversal result comprises the destination object.

14. The system of claim 12 wherein the link represents an
    action and wherein the traversal result comprises an
    indication that the link provider has taken the action.

15. The system of claim 12 wherein the link belongs to a
    link category and wherein the traversal result comprises
    a destination object belonging to the link category.

16. The system of claim 15 wherein the link category is
    defined by a link provider.

17. The system of claim 12 wherein the link service
    returns the traversal result to the client.

18. The system of claim 17 and further comprising: a
    handler operably coupled to the client and configured to
    handle the traversal result.

19. The system of claim 5 wherein the link service is
    configured to receive a registration request for an additional
    link provider.

20. The system of claim 19 wherein the link service
    queries the additional link provider for register information
    corresponding to the additional link provider and stores the
    register information in the provider register.

21. A method of maintaining links between objects comprising:
    receiving a link request at a link service from a client, the
    link request identifying an object that is a source object
    in links being requested;

    requesting links from a plurality of providers, based on
    the identified object;

    aggregating links from the providers, at the link service;
    and

    returning the aggregated links to the client.

22. The method of claim 21 wherein the link request
    identifies an instance of the source object and wherein
    requesting comprises:

    requesting links based on the instance of the source object.

23. The method of claim 22 wherein the link request
    identifies a specific instance of the source object and
    wherein requesting comprises:

    requesting links based on the specific instance of the
    source object.

24. The method of claim 21 wherein the link request
    identifies a class of the source object and wherein requesting
    comprises:

    requesting links based on the class of the source object.

25. The system of claim 21 wherein requesting comprises:
    accessing a provider register storing register information
    corresponding to the providers, the register information
    being indicative of links provided by the providers.

26. The method of claim 21 and further comprising:
    receiving a traversal request from the client, the traversal
    request identifying a link to traverse.

27. The method of claim 26 and further comprising:
    determining which link provider provided the link; and
    requesting that link provider to traverse the link, the link
    provider returning a traversal result.

28. The method of claim 27 wherein the link represents a
    destination object and further comprising:

    returning the destination object to the client.

29. The method of claim 27 wherein the link represents an
    action and further comprising:

    returning an indication that the link provider has taken the
    action to the client.

30. The method of claim 21 and further comprising:
    receiving a registration request for an additional link
    provider.
31. The method of claim 30 and further comprising:
querying the additional link provider for register information corresponding to the additional link provider; and

storing the register information.

32. A method of maintaining hypermedia links, comprising:

receiving, at a hypermedia service, a registration request for a link provider that provides links between objects;
querying the link provider for register information indicative of source nodes for which the link provider provides links; and

storing the register information.

33. The method of claim 32 wherein querying comprises:
querying the link provider for object classes for which the link provider provides links.

34. The method of claim 32 wherein querying comprises:
querying the link provider for instances of objects for which the link provider provides links.

35. A method of traversing links between objects the method comprising:
receiving, at a hypermedia service, a traversal request from a client, the traversal request identifying a link to traverse;
determining which of a plurality of link providers provided the link; and
requesting that link provider to traverse the link, the link provider returning a traversal result.

36. The method of claim 35 wherein the link represents a destination object and further comprising:
returning the destination object to the client.

37. The method of claim 35 wherein the link represents an action and further comprising:
returning an indication that the link provider has taken the action to the client.

38. The method of claim 35 and further comprising:
returning the traversal result to the client; and
the client returning the traversal result to a handler that handles the traversal result.