



3.6 STRUCTURE OF A DO-IRP SERVICE.

A DO-IRP service is, in general, a collection of multiple service sites, each hosting a full replica of the identifiers of the service. Each service site, in general, consists of multiple servers, each with a particular subset of the identifiers of the service (typically determined by a hashing algorithm). The DO-IRP protocol specifies the procedure for a client to determine which site within the service and which server within the site should be contacted (see section 7.1).

4 DATA MODEL

Each identifier may have a set of elements assigned to it, which are collectively known as the identifier's "Identifier Record". There is no inherent limit to the number of elements in each identifier record. Records can optionally contain an element which is a signature certifying the contents of the record; additionally, if requested by a client, messages used to transmit records shall be signed for security. Each such element has a specified "type" which is represented by a unique identifier of its own. DO-IRP servers maintain these Identifier Records and typically return an instance of each such requested Identifier Record in response to a request for resolution of that designated identifier. More tailored requests can result in specific elements of an identifier record being shown in a given returned identifier record.

4.1 IDENTIFIER RECORD

An identifier record thus consists of one or more elements, each of which contains multiple fields including its type, a value associated with that type, and a unique index number that distinguishes that element from other elements in the record, especially those with the same type. The specific type shown in an element may define the syntax, structure, possible semantics, or any other necessary descriptive characteristics of the element's value field. Each element also contains a set of administrative information such as a timestamp, Time to Live ("TTL") and certain associated permissions. Pre-defined types are discussed in section 4.3.

Figure 4.1 below shows the identifier "35.1234/abc" with a sample record with three elements with indexes 1,2 and 3; The element 1 is shown in detail below.

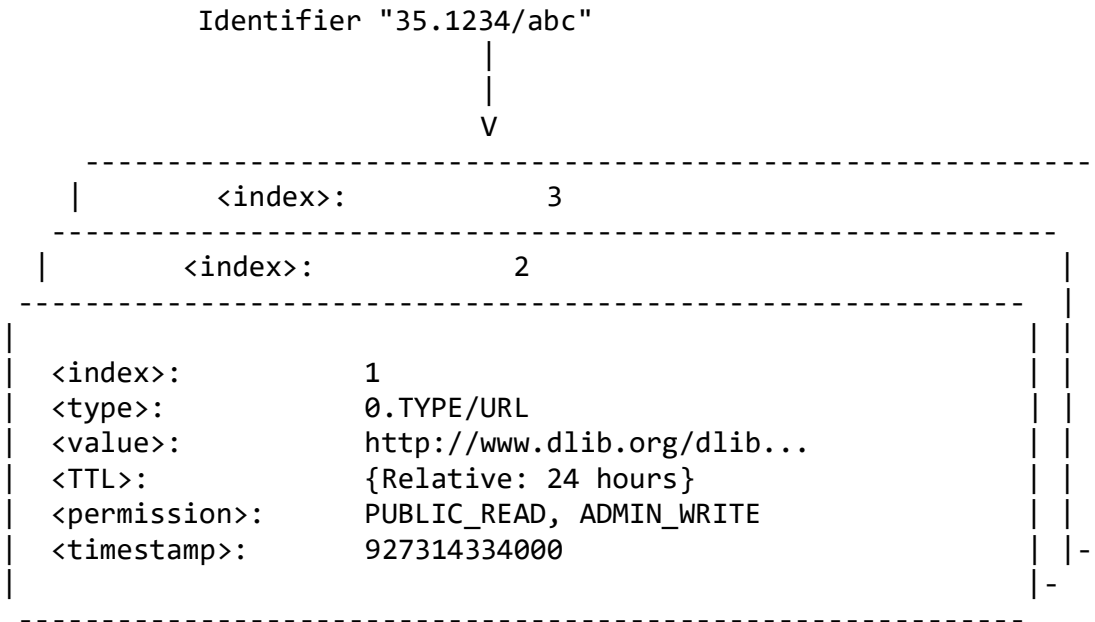


Figure 4.1: Identifier "35.1234/abc" and its set of three elements

Figure 4.1 shows a single element whose index is set to 1. The type for this element is URL [5], which is represented in the system by the resolvable identifier 0.TYPE/URL. In general, a type is represented as a unique identifier and will either be a system type or a user created type. The URL data as stated in the <value> field is "http://www.dlib.org/dlib... ". The TTL (time to live) entry suggests that the record should be cached no more than 24 hours before the source of the information should be consulted again. The <permission> field grants anyone permission to read; but only the administrator has permission to update that particular element.

Thus, an element may be thought of as group of fields which are defined below, along with the standard binary encoding of an element used by the DO-IRP protocol.

<index>

An unsigned 32-bit integer that uniquely differentiates an element from all the other elements in the identifier record. The index 0 is reserved. For maximum compatibility with existing implementations, indexes greater than or equal to 2^{31} (which would represent negative values if interpreted as signed integers) should not be used.

<type>

A UTF8-string that specifies the identifier for the type of the value field in the element. Note that throughout this document, a UTF8-string consists of a 4-byte unsigned integer followed by an UTF-8 encoded character string. The integer specifies the number of octets in the character string.

The <type> field is used to specify the identifier of the type that defines the syntax, format, and possible semantics of the data in the element's <value> field. The identifier for a type



can be specified by the creator of the element as desired, but this standard recommends that it be an identifier resolvable using DO-IRP to make it globally resolvable and accessible. See section 9 on Type Identifiers Consideration for more details.

The UTF8-string in the <type> field shall not end with the "." character. However, for clarity, the "." character may appear in the end of the <type> prefix used in an identifier query. This prefix string is used to query for all elements under a common type hierarchy. For example, one may query for all elements under the type hierarchy "a.b" (e.g., elements of <type> "a.b.x", "a.b.y" and "a.b.z") by setting the <type> parameter of the query to "a.b.". Note here that the <type> prefix in the query must specifically end with the "." character. Details of the query operation can be found in the protocol specification in section 7.2.

<value>

A sequence of octets (preceded by its length in a 4-byte unsigned integer). The syntax, format, and semantics of these octets are determined by the <type> field.

<permission>

An eight-bit bit-mask for access control of the element. Access control is defined in terms of read and write permissions, applicable to either the general public or administrator(s). DO-IRP servers must evaluate each element's permissions before fulfilling any client request to make sure that only properly authenticated and authorized clients receive the element. Each element can have its permission field specified as any combination of the following bits:

- PUBLIC_WRITE (0x01)
permission that allows anyone to modify or delete the element.
- PUBLIC_READ (0x02)
permission that allows anyone to read the element.
- ADMIN_WRITE (0x04)
permission that allows any authenticated administrator with Modify_Element / Delete_Element privileges (see 4.3.1) to update or delete the element
- ADMIN_READ (0x08)
permission that allows any authenticated administrator with "Authorized_Read" privilege (see section 4.3.1 below) to read the element.

It is normally intended that only administrators shall be granted write permissions (in addition to read permissions). An element with no PUBLIC_READ nor ADMIN_READ permission will not be accessible via DO-IRP by anyone except by the DO-IRP server administrator, once set. Such an element may be used, for example, to store secret keys for authentication purposes. An element with neither PUBLIC_WRITE nor ADMIN_WRITE permission makes the element immutable and it cannot be deleted via DO-IRP. The most common permissions are "1110" (admin read and write, public read) and "1100" (admin read and write, no public access).

The administrator for a given identifier must specify the permission for each element. It is recommended that implementations use PUBLIC_READ, ADMIN_READ and ADMIN_WRITE as



the default permission for each element.
The more significant bits of the permission field are reserved.

<TTL>

An octet followed by a 4-byte integer that specifies the Time-To-Live of the element. It is used to describe how long the element should be cached by clients before the source of the information (i.e., the corresponding DO-IRP server) should again be consulted. A zero value for a TTL indicates that the element should only be used for the transaction in progress and should not be cached. Any non-zero TTL is defined in terms of a TTL type (specified in the first octet), followed by the TTL value (the 32-bit integer that follows the TTL type). The TTL type indicates whether the TTL value is relative (octet 0x00) or absolute (octet 0x01). An absolute TTL value is an unsigned integer which defines the time to live as an expiration in terms of seconds since 00:00:00 UTC, January 1st 1970. A relative TTL specifies the time to live in terms of the number of seconds elapsed since the element was obtained by the client from any DO-IRP server. A relative TTL which would be negative if interpreted as a signed integer indicates that the element should not be cached.

<timestamp>

A 4-byte unsigned integer that records the last time the element was updated at the server. The field contains elapsed time since 00:00:00 UTC, January 1970 in seconds.

<references>

This field is deprecated and should be populated with four zero octets 0x00, 0x00, 0x00, 0x00. Since older implementations may have used this field, clients may need to process it in which case the structure of the field is a 4-byte integer followed by list of “references” each composed of a string. The initial integer specifies the number of references in the list. Each reference in the list is a UTF8-string followed by a 4-byte integer.

4.2 IDENTIFIER RECORD CONSIDERATIONS

By default, DO-IRP servers return all the elements with public-read permission in response to any resolution request. It is possible for a client to ask for a subset of those elements with a specific type (e.g., all URLs stored in the identifier record). The client may also ask for a specific element based on a specific index.

Each element can be uniquely referenced by the combination of the identifier and its element index. Care must be taken when changing the index as it may break an existing reference to the element. For example, suppose the record of identifier X/Y has an element whose index is 1. That element may be referred to as 1:X/Y. If the administrator changes the index from 1 to 2, the reference to 1:X/Y will become obsolete. To remain accurate, any reference to the element will have to change to 2:X/Y.



Elements within an identifier record may or may not have contiguous index numbers. Nor should it be assumed that the indexes will start with 1. An administrator may assign an element with any index as long as each index is a unique integer within the record.

An element may be “privatized” or “disabled” by clearing the “public-read” bit on its <permission> field. This limits read-access to the administrator only. The “privatized” element can then be used to keep any historical data (on behalf of the administrator) without exposing it to public. Such an approach may also be used to keep any obsolete identifier or prefix from being reused accidentally. This is also used for the HS_SECKEY type; see section 4.3.7.

4.3 PRE-DEFINED ELEMENT TYPES

Type is a required field in an element, although the exact type to be used is open-ended. Some types are defined in this document and have specific server and client processing semantics. DO-IRP implementations must use the defined types whenever applicable. New types can be defined if different processing rules were to apply; in such cases, those new types may be registered by an organization as identifiers for resolution using DO-IRP under a prefix controlled by the organization.

Types defined in this document are assumed to have an implicit prefix of “0.TYPE”, although the prefix is generally omitted when writing about the use of these types.

The following types have defined DO-IRP server and client processing semantics: HS_ADMIN, HS_SITE, HS_SITE.PREFIX, HS_SERV, HS_SERV.PREFIX, HS_PUBKEY, HS_SECKEY, and HS_VLIST.

Additionally, the types HS_ALIAS, HS_CERT, and HS_SIGNATURE, while not necessary for DO-IRP service use, are considered standard system types that cut across multiple applications; and all are registered under the prefix “0.TYPE”.⁶

The following sub-sections provide detailed descriptions of these system types.

4.3.1 IDENTIFIER ADMINISTRATOR: HS_ADMIN

An administrative operation on an identifier record or on the DO-IRP service (e.g., add, delete or modify an element) can only be performed by an administrator that is authenticated and that has adequate privileges.

⁶ Historical note: the term HS in the name of these types comes from the legacy notation that identified these types as Handle System (HS) types.



We defined an “administrator” as an entity represented by an identifier and an element index within that identifier record. This identifier-index is a reference to an element that must contain a public key or secret key to be used when challenging the entity to authenticate itself.

An authenticated administrator is an entity that has demonstrated that it possesses either the private key matching the public key, or the secret key, pointed to by the administrator’s identifier-index element reference.

The privileges or authorizations that an administrator has over an identifier record is specified in one or more elements of type HS_ADMIN within that identifier record. Each HS_ADMIN element can be used to define a set of administrators sharing the same administration privilege. Identifiers with multiple administrators of different privileges may have multiple HS_ADMIN elements. HS_ADMIN elements are used by DO-IRP servers to authenticate administrators before fulfilling any DO-IRP administration request. However, as an implementation option, an identifier record without an HS_ADMIN element can be modified by a qualified DO-IRP server administrator.

An HS_ADMIN element is one whose <type> field is HS_ADMIN and whose <value> field consists of the following entries encoded as binary data:

<AdminPermission>

A 16-bit bit-mask that defines the administration privilege of the administrator or set of administrators identified by the <AdminRef> entry; see below for the permissions.

<AdminRef>

Specifies the record’s administrator. It consists of the administrator’s identifier and element index (as defined above), encoded as a 4-byte length followed by that many UTF-8 encoded bytes, followed by a 4-byte unsigned integer for the index of the element that is referenced in the administrator’s identifier record. The referred element in the administrator’s identifier record must either be an identity authentication element (such as an HS_PUBKEY or HS_SECKEY element) or an HS_VLIST element, which specifies a group of administrators consisting of a list of one or more administrator identification and authentication elements references. (See HS_VLIST paragraph below for more description as well as HS_VLIST in section 4.2.8 for its detailed specification). An index value of 0 implies that the reference is to all elements in the administrator’s identifier record; this means all HS_SECKEY or HS_PUBKEY elements in that record can be used.

The <AdminRef> entry may refer to an administrator with the same identifier the record of which has the HS_ADMIN element. In this case, authentication of the administrator does not rely on any other identifiers and does not require any additional identifier resolution. Alternatively, the <AdminRef> entry may refer to an administrator in a different identifier record requiring additional resolution operations. As a consequence, HS_ADMIN elements from different identifiers may share common administrator(s), even across different prefixes. This feature provides a flexible solution to the task of administrating large collections of identifier records which could range from a few



identifier records in a single prefix to very large numbers of identifier records spanning multiple prefixes.

The HS_VLIST element is used to specify a list of one or more administrators. It is typically used to define a list of administrators that can administer an element. This list is referred to as an administrative group. The use of HS_VLIST allows a single HS_ADMIN element to have as many individual administrators as specified in the HS_VLIST. Each administrator reference is defined in terms of an <index>:<identifier> pair. An administrator group may also contain other administrator groups as its members. This allows administrator groups to be defined in a hierarchical fashion. Care must be taken, however, to avoid cyclic definition of administrators or administrator groups. Multiple levels of administrator groups should be avoided due to their lack of efficiency, but will not be signaled as an error. Client software should be prepared to detect any potential cyclic definition of administrators or <AdminRef> entries that point to non-existent elements and treat them as an error.

An identifier record can have multiple HS_ADMIN elements, each of which references a different administrator and with possibly different permissions. Different administrators can play different roles or be granted different permissions. For example, the identifier record that corresponds to a prefix, such as the one associated with the identifier “0.NA/35.978”, may have two administrators, one of which may only have permission to create new identifiers under the prefix, while the other may have permission to create new derived prefixes (e.g., “35.978.123”).

Beyond its use in the HS_ADMIN, the HS_VLIST could be used in other situations where a grouping of identifiers is needed.

The set of possible permissions for an administrator is defined as follows:

Add_Identifier (0x0001)

This permission, when set in a prefix identifier record, allows an authenticated administrator to create new identifiers under that prefix. This permission is only meaningful when set in an identifier record pertaining to a prefix, as it is a prefix-only permission.

Delete_Identifier (0x0002)

This permission allows an authenticated administrator to delete the identifier record.

Add_Derived_Prefix (0x0004)

This permission, when set in a prefix identifier record, allows an authenticated administrator to create new prefixes derived from the stated prefix identifier. This permission is only meaningful when set in a prefix identifier record and it is a prefix-only permission.

Reserved (0x0008)

This bit is reserved for historical reasons.



Modify_Element (0x0010)

This permission allows an authenticated administrator to modify any elements other than HS_ADMIN elements. HS_ADMIN elements are used to define administrators and are managed by a different set of permissions (as described next).

Delete_Element (0x0020)

This permission allows an authenticated administrator to delete any element other than the HS_ADMIN elements.

Add_Element (0x0040)

This permission allows an authenticated administrator to add elements other than the HS_ADMIN elements.

Modify_Admin (0x0080)

This permission allows an authenticated administrator to modify HS_ADMIN elements.

Remove_Admin (0x0100)

This permission allows an authenticated administrator to remove HS_ADMIN elements.

Add_Admin (0x0200)

This permission allows an authenticated administrator to add new HS_ADMIN elements.

Authorized_Read (0x0400)

This permission grants an authenticated administrator read-access to elements with the ADMIN_READ and without PUBLIC_READ permission. Administrators without this permission will not have access to elements that require authentication for read access.

List_Identifiers (0x0800)

This permission allows an authenticated administrator to list all identifiers under a designated prefix, even if such identifiers are managed in a distributed fashion on multiple servers. Identifiers that are based on prefixes derived from the specified prefix are not included in the listing. This is a prefix-wide setting and must be set on the respective prefix identifier record.

List_Derived_Prefixes (0x1000)

This permission allows the administrator to list all prefixes derived from a designated prefix. If such derived prefixes have in turn their own derived prefixes, those further derivatives are also included in the listing as long as such derived prefix records are on the same DO-IRP service on which this listing operation is performed. This is a prefix-wide setting and must be set on a prefix identifier record.

Administrator permissions are encoded in the <AdminPermission> entry in the <value> field of any HS_ADMIN element. Each permission is encoded as a bit flag. The permission is granted if the flag is set to 1.

HS_ADMIN elements are used by DO-IRP servers to authorize the administrator before fulfilling any administrative requests. The server authenticates a client by checking whether the client has possession of the secret key or the private key that matches the secret key or the public key



corresponding to the identity claimed by the client; the server considers an authenticated client authorized if that identity matches any of the administrator references (i.e., AdminRef). The authentication is carried out via the DO-IRP authentication process, see section 5.2.

HS_ADMIN elements may require authentication for read access (when no PUBLIC_READ permission is configured at element level) in order to prevent public exposure of the data. Additionally, if a secret key is used instead of a public key, the administrator reference that contains the administrator's secret key should not have PUBLIC_READ permission so that the key is not publicly visible.

Modify_Element, Delete_Element, Modify_Admin, and Remove_Admin permissions are not usable if the respective element is not configured with the ADMIN_WRITE or PUBLIC_WRITE permission (see 4.1).

4.3.2 SERVICE SITE INFORMATION: HS_SITE

HS_SITE is used to provide information about a DO-IRP service to inform a DO-IRP clients how to contact it. This information specifies the server's address, its protocol version, its public key. Each prefix identifier should have one or more HS_SITE elements in its prefix identifier record. This is called the service information for the prefix and denotes, among other things, the location of servers where any existing identifier that is based on that prefix can be created, updated, deleted, or resolved. A user that tries to resolve an identifier that does not exist will get a "identifier not found" response from the local service specified in the HS_SITE information. Attempts to resolve an invalid identifier, namely one that does not exist within the designated LIS, will get a "identifier not found" response from that local service.

The service information is managed by the system administrator. It must reflect the configuration of the DO-IRP service for each of the prefixes it manages. An additional layer of indirection is provided by the use of HS_SERV, which is called a service identifier; HS_SERV allows multiple prefixes to reference a single set of HS_SITE elements, as described in section 4.3.4. DO-IRP clients depend on the service information to locate a responsible DO-IRP server before they can send their service requests. The service information can also be used by clients to authenticate any service response from the server.

An HS_SITE element is one whose <type> field is HS_SITE and whose <value> field consists of the following entries encoded as binary data:

<Version>

A 2-byte value that identifies the version number of the data format used in the encoding of the <value> field of the HS_SITE element. It is defined to allow backward compatibility over time. This section of this document defines the HS_SITE encoding with version number 1.



Note that this version number is not the same as the DO-IRP protocol version number used in request and response messages.

<ProtocolVersion>

A 2-byte integer value that identifies the highest DO-IRP protocol version understood by servers of the site. The higher byte of the value identifies the major version and the lower byte the minor version. Details of the message format for DO-IRP protocol version 3.0 are specified in section 6.2. Although the terminology used is the same, this field does not denote the protocol version of this DO-IRP specification. Rather, it denotes the protocol version as specified by the prefix administrator for the implementation of the local service that supports the identifier record. Local services which support the DO-IRP protocol version 3.0 specified in this document should be described with HS_SITE elements specifying <ProtocolVersion> 3.0.

<SerialNumber>

A 2-byte integer value that should be increased by administrators each time the HS_SITE element gets changed. For uses currently envisioned, it is acceptable for the 16-bit serial number to wrap around. It is used in the DO-IRP protocol to synchronize the HS_SITE elements between client and server. In general, clients are expected to synchronize HS_SITE elements when they expire from cache, which they do by re-resolving the identifier.

<PrimaryMask>

An 8-bit mask that identifies the primary site(s) of the DO-IRP service. The first bit of the octet is the <PrimarySite> bit. It indicates whether the HS_SITE element is a primary site. A primary site is the one that supports administrative operations for its identifiers. The second bit of the octet is the <MultiPrimary> bit. It indicates whether the service has multiple primary sites. A <MultiPrimary> entry with zero value indicates that the service has a single primary site and all administration has to be done at that site. A non-zero <MultiPrimary> entry indicates that the service has multiple primary sites. Each primary site may be used to administer identifiers managed under the service.

<HashOption>

An octet that identifies the hash option used by the service site to distribute identifiers among its multiple servers. This is not meaningful for a typical service site with a single server. Valid options include HASH_BY_PREFIX (0x00), HASH_BY_SUFFIX (0x01), or HASH_BY_IDENTIFIER (0x02). These options indicate whether the hash operation should only be applied to the prefix portion of the identifier, or only the suffix portion of the identifier, or the entire identifier, respectively. See section 7.1 for the hashing algorithm used by each service site to distribute identifiers among its servers.

<HashFilter>

An UTF8-string entry reserved for future use.

<AttributeList>

Attributes used to add literal explanations of the service site. An attribute is a <type>:<value> pair. The entry consists of a 4-byte integer followed by a list of UTF8-string pairs. The integer



indicates the number of attributes (UTF8-string pairs) that follow. The most common <type> is “desc”, for which the <value> should contain a description of the organization hosting the service site. Other attributes may be defined to help distinguish the service sites from each other. Resolvers may use the attribute type “alt_addr” as an indication that a site’s server (or “alt_addr.<ServerID>” for one of a site’s multiple servers) is reachable via an alternate IP address. This can be used to indicate a site with dual-stack reachability over IPv4 and IPv6. Resolvers may use the attribute types “domain” and “path” (or “domain.<ServerID>” and “path.<ServerID>” for use with HTTP(S) tunneling as described in section 6.1.2.3.

<NumOfServer>

A 4-byte integer that defines the number of servers in the service site. The entry is followed by a list of <ServerRecord>s. Each <ServerRecord> defines a server that is part of the service site.

Each <ServerRecord> consists of the following data fields:

```
<ServerRecord> ::= <ServerID>
                   <Address>
                   <PublicKeyRecord>
                   <ServiceInterface>
```

where each field is defined as follows:

<ServerID>

A 4-byte unsigned integer that uniquely identifies a server process under the service site. <ServerID>s do not have to begin with 1 and they don’t have to be consecutive numbers. They are used to distinguish servers under a service site from each other. Note that there can be multiple servers residing on any given computer, each with a different <ServerID>.

<Address>

The 16-byte IPv6 address of the server. Any IPv4 address should be presented as either ::xxxx:xxxx or ::FFFF:xxxx:xxxx (where xxxx:xxxx can be any 4-byte IPv4 address).

<PublicKeyRecord>

A 4-byte integer followed by a byte-array that contains the server’s public key (server authentication key). The integer specifies the size of the byte-array. For key types described in this specification, the byte-array (for the public key) consists of a number of parts: a UTF8-string that describes the key type, a two-byte option field reserved for future use, and a key-type-dependent number of length-prefixed byte-arrays that describe the public key itself. The key types in current use are “DSA_PUB_KEY,” where there are four byte-arrays after the two-byte option field for the four DSA parameters q, p, g, and y; and “RSA_PUB_KEY”, where after the two-byte option field are two byte-arrays for the exponent and modulus, followed by an empty byte-array (four zero bytes). Other key types may be useful to consider in the future.

The public key in the <PublicKeyRecord> can be used to authenticate any service response from the DO-IRP server.



<ServiceInterface>

Consists of the following data fields:

```
<ServiceInterface> ::= <InterfaceCounter>
                        * [ <ServiceType>
                           <TransportProtocol>
                           <PortNumber> ]
```

A 4-byte integer followed by an array of triplets consisting of <ServiceType, TransportProtocol, PortNumber>. The 4-byte integer specifies the number of triplets. Each triplet lists a service interface provided by the server. For each triplet, the <ServiceType> is an octet (as a bit mask) that specifies whether the interface is for administration (0x01), resolution (0x02), or both (0x03). The <TransportProtocol> is also an octet (as a bit mask) that specifies the protocol. Possible protocols include UDP (0x00), TCP (0x01), HTTP (0x02), and HTTPS (0x03). The <PortNumber> is a 4-byte unsigned integer that specifies the port number used by the interface. The conventional port number used by DO-IRP servers is 2641 for UDP and TCP, and 8000 for HTTP.

Each server within a service site is responsible for a subset of identifiers managed by the DO-IRP service. Clients can find the responsible server by performing a common hash-operation. See section 7.1.

4.3.3 SERVICE SITE INFORMATION: HS_SITE.PREFIX

The HS_SITE.PREFIX has the exact same format as the HS_SITE type described above. Like HS_SITE elements, HS_SITE.PREFIX elements are used to describe service sites of a DO-IRP service. HS_SITE.PREFIX elements may be assigned to prefix identifiers to designate that derived prefixes may be resolved at the specified service. A prefix identifier associated with a set of HS_SITE.PREFIX elements indicates that derived prefixes of the prefix, if any, may be managed by the service described by the HS_SITE.PREFIX elements.

HS_SITE.PREFIX and HS_SERV.PREFIX elements are not generally used directly by clients, but by a server which receives client requests about prefix identifiers, in order to determine whether to respond with a RC_PREFIX_REFERRAL response instead of a RC_ID_NOT_FOUND response; see section 7.4.

4.3.4 SERVICE IDENTIFIER: HS_SERV

Any DO-IRP service can be defined in terms of one or more HS_SITE elements. These HS_SITE elements may be assigned directly to the relevant prefix identifier, or an additional level of indirection may be introduced through the use of an HS_SERV element in the prefix identifier record. The value of the HS_SERV element contains the service identifier, meaning an identifier



whose record contains the HS_SITE elements defining the DO-IRP service. This way, the HS_SITE elements can be maintained in a separate record

Use of service identifiers allows sharing of service information among multiple prefixes. It also allows changes to service configuration (e.g., adding a new site) to be made in one place rather than in every prefix identifier involved.

Although not typical, a prefix identifier may have multiple HS_SITE and multiple HS_SERV elements. In such a case the service information for the prefix should be considered as the concatenation of the HS_SITE elements in the prefix identifier record, together with the service information from all of the HS_SERV elements.

The use of service identifiers raises several special considerations. Multiple levels of service identifier redirection should be avoided due to their lack of efficiency, but are not signaled as an error. Looped reference of service identifiers or HS_SERV elements that refer to non-existent service identifiers should be caught and error conditions passed back to the user.

4.3.5 SERVICE IDENTIFIER: HS_SERV.PREFIX

HS_SERV.PREFIX serves the same role with respect to HS_SITE.PREFIX as HS_SERV serves with respect to HS_SITE. If a prefix identifier has an HS_SERV.PREFIX element, the data of that element is an identifier, whose corresponding record describes, via one or more HS_SITE.PREFIX elements and/or recursively via HS_SERV.PREFIX elements, the service information at which clients may be able to resolve derived prefixes of the original prefix.

4.3.6 IDENTITY: HS_PUBKEY

An element of type HS_PUBKEY stores a public key. The element can be used as an administrative identity, for referring to in HS_ADMIN or HS_VLIST elements for authorization, or for identifying the administrative identity in DO-IRP authentication. A reference to the element for use as an administrative identity is as a pair of the identifier and the index of the element within the identifier record; in this document this is often written with a colon as <index>:<identifier>.

The <value> of the element is a binary encoding of the public key which for key types considered in this specification is as follows. First, there is a UTF8-string that describes the key type; then, a two-byte option field reserved for future use; and finally, a key-type-dependent number of length-prefixed byte-arrays that contains the public key itself. The key types in current use are “DSA_PUB_KEY”, where there are four byte-arrays after the two-byte option field for the four DSA parameters q, p, g, and y; and “RSA_PUB_KEY”, where after the two-byte option field are two byte-arrays for the exponent and modulus, followed by an empty byte-array (four zero bytes).

4.3.7 IDENTITY: HS_SECKEY

An element of type HS_SECKEY is used to store a secret key. The element can be used as an administrative identity, for referring to in HS_ADMIN or HS_VLIST elements for authorization, or for identifying the administrative identity in DO-IRP authentication.

The <value> of the element is the secret key. In order to protect the secret key, the <permission> of the element should be set to forbid PUBLIC_READ. This is the only common usage of any <permission> other than PUBLIC_READ, ADMIN_READ, and ADMIN_WRITE.

Use of public keys is recommended over use of secret keys for DO-IRP authentication.

4.3.8 VALUE LIST: HS_VLIST

HS_VLIST allows referencing a list of elements in other identifiers, and allows the referenced elements to be updated without requiring an update of the referring identifier record. An HS_VLIST element is one whose <type> is HS_VLIST and whose <value> consists of a 4-byte unsigned integer followed by a list of references to other elements. The integer specifies the number of references in the list. The references may refer to elements under the same identifier or elements from other identifiers. Each reference is encoded as an UTF8-string followed by a 4-byte unsigned integer that identifies the referenced identifier and its index.

HS_VLIST elements may be used to define administrator groups for identifiers. Each administrator (as defined in 4.3.1) in the HS_VLIST includes is a member of the administrator group. Each element reference is defined in terms of an <index>:<identifier> pair. An administrator group may also contain other administrator groups as its members. This allows administrator groups to be defined in a hierarchical fashion. Care must be taken, however, to avoid a cyclic definition of administrators or administrator groups. Multiple levels of administrator groups should be avoided due to their lack of efficiency, but will not be signaled as an error. Client software should be prepared to detect any potential cyclic definition of administrators that refer to non-existent elements and treat them as errors.

4.3.9 ALIASES: HS_ALIAS

It is possible that the set of elements (i.e., type-value pairs) of a digital object may be included in other digital objects and thus multiple identifiers could be associated with the same information represented in digital form, thus creating, in effect, multiple different digital objects all with the same information represented in digital form. DO-IRP does not specify a specific mechanism for identifier records to reference all such alternatives, but various DO-IRP implementations and their users should use the pre-defined HS_ALIAS type in such cases.



An HS_ALIAS element is one whose <type> field is HS_ALIAS and whose <value> field contains a reference to another identifier. An identifier whose record contains an HS_ALIAS element is an alias to the identifier referenced in the HS_ALIAS element. An alias record should not have any additional elements other than HS_ALIAS or HS_ADMIN (for administration) elements. This is necessary to prevent any inconsistency between an identifier and its aliases.

During an identifier resolution, a client may get back an HS_ALIAS element. This indicates that the identifier in question is an alias identifier. The client may then retry the query against the identifier specified in the HS_ALIAS element until final results are obtained.

The use of HS_ALIAS introduces a number of special considerations. For example, multiple levels of aliases should be avoided for the sake of efficiency, but are not signaled as an error. Alias loops and aliases that point to non-existent identifiers should be caught and error conditions passed back to the user.

4.3.10 CRYPTOGRAPHIC CLAIMS: HS_CERT

HS_CERT and HS_SIGNATURE elements are used for offline signatures of identifier records. However, if an implementer chooses to do so, he/she could develop interoperable client and server software without making use of HS_CERT and HS_SIGNATURE. The DO-IRP protocol (see sections 6.2.3 and 6.2.4) allows a client to authenticate responses received from a server by requesting that the server include a request digest and sign the response. Offline signatures complement this by including in an identifier record an HS_SIGNATURE element, which is a claim that certain elements are present, signed by an identified entity; and HS_CERT elements, which form a chain of trust which can be used to determine whether the entity signing an HS_SIGNATURE element has the authority to state which elements should be in the identifier record. By means of HS_SIGNATURE and HS_CERT elements, a client can validate the contents of identifier records, regardless of the trustworthiness of the communication channel by which it obtains them.

DO-IRP servers can use the HS_CERT chain of trust as a parallel authorization mechanism for the creation and update of identifier records. This is particularly useful when a DO-IRP service consists of multiple sites with separate administrative and organizational control; the sites need to mirror an identical collection of records, but in principle do not trust each other. The sites can choose to mirror only records that have a valid HS_SIGNATURE according to a specified chain of trust. Even for direct administration by clients, a DO-IRP server could require that not only the client have correct permissions for the operation according to HS_ADMIN elements, but also that the resulting records have a valid HS_SIGNATURE element.

Both HS_CERT and HS_SIGNATURE elements are serialized as JWT claims sets in a JWS structure (RFC 7515, RFC 7519). They differ in the expected claims, interpretation, and validation.



The HS_CERT element claims set will have the following claims:

- "iss": the issuer of the certificate, as an element reference (<index>:<identifier> pair) or identifier
- "sub": the subject of the certificate, as an element reference (<index>:<identifier> pair) or identifier
- "exp", "nbf", "iat": expiration, not-before, and issued-at dates as numeric seconds since the epoch
- "publicKey": the public key of the subject, in JWK format (RFC 7517) [6]
- "perms": a list of permission objects, where each permission object has the form
 { "handle": "<identifier>", "perm": "<permission>" }
with "<permission>" one of "everything", "thisHandle", "derivedPrefixes", or "handlesUnderThisPrefix"
- "chain": an optional list of element references (<index>:<identifier> pairs) or identifiers, used to build a chain of trust for validating the certificate issuer (in the absence of an explicit chain, the chain can still be built implicitly, as will be discussed)

An HS_CERT indicates an assertion by one entity (the issuer) that a second entity (the subject) is authorized to sign certain identifier records with its public key. It is typically the case that the issuer and subject reference HS_PUBKEY elements in identifier records; however, the "publicKey" contained in the HS_CERT itself is actually used in validating claims signed by the subject. (It is conventional for the HS_PUBKEY element to be identical to the "publicKey".)

The "perms" claim indicates for which identifier records the subject's signature should be considered authorized, as a subset of the identifier records for which the issuer's signature is authorized:

- { "perm": "everything" } indicates authorization over the same handle records as the issuer
- { "handle": "<identifier>", "perm": "thisHandle" } indicates authorization over <identifier>, and, if <identifier> is a prefix identifier, also identifiers with that prefix
- { "handle": "<prefixIdentifier>", "perm": "derivedPrefixes" } indicates authorization over prefix identifiers derived from <prefixIdentifier>, as well as identifiers with any prefix derived from <prefixIdentifier>
- { "handle": "<prefixIdentifier>", "perm": "handlesUnderThisPrefix" } indicates authorization over identifiers with the prefix whose prefix identifier is <prefixIdentifier>

In no case should the subject be considered to be given authorizations not available to the issuer.



The building of a chain of trust using HS_CERT elements is discussed below.

4.3.11 CRYPTOGRAPHIC SIGNATURES: HS_SIGNATURE

The HS_SIGNATURE element claims set will have the following claims:

- "iss": the issuer of the certificate, as an element reference (<index>:<identifier> pair) or identifier
- "sub": the subject of the certificate, an identifier (the identifier of the record signed)
- "exp", "nbf", "iat": expiration, not-before, and issued-at dates as numeric seconds since the epoch
- "digests": a structure indicating the hashed values of the elements of the signed identifier record. It is an object with two properties: "alg", a string indicating the digest algorithm (e.g. "SHA-256" [7]); and "digests", a list of objects. Each of those objects has two properties: "index", a number indicating the index of an identifier record element, and "digest", a string with a Base64-encoding of the digest (or hash) of the element, as described below.
- "chain": an optional list of element references (<index>:<identifier> pairs) or identifiers, used to build a chain of trust for validating the signature issuer (in the absence of an explicit chain, the chain can still be built implicitly, as will be discussed).

An HS_SIGNATURE indicates an assertion about the elements contained in an identifier record. The issuer is the entity making the claim; the subject is the identifier; the digests indicate the digests (according to a hash algorithm like SHA-256) of the elements. The digests are of the byte-arrays of the elements as encoded for the DO-IRP protocol, but omitting the first eight bytes, which are the index and the timestamp (this is necessary for the client to generate digests of new elements, as the timestamp is generally server-generated).

In order to validate HS_SIGNATURE or HS_CERT claims, a client must build a chain of trust: a sequence of elements where the issuer of one element is the subject of the next element. The first element may be either an HS_SIGNATURE or HS_CERT; all subsequent elements will be HS_CERT. The sequence ends with a "self-signed certificate" where the issuer and subject are the same. A client can use the following algorithm to build the chain. At each point, the client must maintain the result so far (a sequence of elements) and the coming chain (a sequence of element references or identifiers). Initially, the result is the singleton element to be validated, and the coming chain is empty.

- 1) If the coming chain is empty, replace the coming chain with the "chain" claim from the last element of the result so far; if that is empty or missing, replace the coming chain with a



singleton identifier corresponding to the identifier of the issuer of the last element of the result so far (with any index removed).

- 2) Take the first reference from the coming chain, now guaranteed to be non-empty. If it has an index, resolve the referenced element to get the next element of the result so far. If not, resolve the entire identifier record for all HS_CERT elements, and from all the HS_CERT elements where the subject of that element is the issuer of the last element of the result so far, choose one which has the latest "iat" (issued-at) time, and that is the next element of the result so far.
- 3) Remove the just-used first element of the coming chain, and repeat, until either reaching a self-signed certificate, or the result so far is too long according to some configuration (for example 50 elements) which is an error condition.

Once the chain of trust is built, it can be validated according to the following conditions:

- 1) Each element must be a validly-signed JWS with the public key given in the next element of the chain (or, in the case of the self-signed one, with its own key);
- 2) The current time must be between the "nbf" (not-before) and "exp" (expiration) times of the claims set;
- 3) The "root" self-signed certificate at the end of the chain of trust is trusted according to client configuration.

Then a particular HS_SIGNATURE can be further validated against an identifier record as follows:

- 1) The subject must match the identifier of the record.
- 2) That identifier must be included in the "perms" claims of each HS_CERT in the built chain.
- 3) The digests of the HS_SIGNATURE must exactly match some subset of the elements of the identifier record.

Finally, an identifier record is considered validly signed if HS_SIGNATURE elements are valid according to the above, and all elements except HS_SIGNATURE elements and elements (like HS_SECKEY elements) not publicly readable are included in the digests of at least one HS_SIGNATURE element.

The specification assumes the existence of the specific identifier 0.0/0.0, which resolves to a record which is used as a default root of trust for the system, in the absence of more specific client configuration. By default, clients can trust a self-signed certificate where the identifier of the issuer is 0.0/0.0 and the public key is in an HS_PUBKEY element of the record of 0.0/0.0. Clients which are validating must not in general trust a new resolution of the 0.0/0.0 record, but should maintain the