Internet Engineering Task Force
INTERNET-DRAFT
draft-ietf-uri-urc-req-00.txt

Ron Daniel Jr.
Los Alamos National Laboratory
Michael Mealling
Georgia Institute of Technology

November 21, 1994

URC Scenarios and Requirements

Status of this draft

This Internet Draft expires May 25, 1995.

Abstract

This draft describes the place of the Uniform Resource Characteristic (URC) service within the overall context of Uniform Resource Identification on the Internet. It presents several scenarios illustrating how the URC service might be used. From these usage scenarios, we derive a set of requirements that any proposed URC services must meet.

Contents

1 Introduction 3

2 User Scenarios 3

2.1 URN to URL resolution 4

2.2 Meta-data for its own sake 5

2.3 Ensuring the veracity of the resource 6

2.4 Ensuring the veracity of the URC 7

2.5 Bibliographic Search 8

2.6 Filtering by Seals of Approval 10

3 Provider Scenarios 11

3.1 Publishing a new resource 11

3.2 Publishing a new version of a resource 12

3.3 Providing an additional location for a resource 12

3.3.1Mirroring of free information 13

3.3.2Mirroring of information that is for sale 13

3.3.3Mirroring on a regular basis 14

3.4 Removing a location for a resource 14

3.5 Establishing a new publishing authority 15

3.6 Dealing with the demise of a publisher 15

4 Requirements 16

4.1 Requirements on the URC 16

4.2 Requirements on the URC Service 17

5 Characteristics 19

1 Introduction

As Joe Jackson says, ``You can't get what you want 'till you know what you want''. This applies to software development just as much as it applies to affairs of the heart. In order for the URI working group to design an architecture that does what we want, we need to know what we want it to do. This paper presents a wide range of scenarios for how we would like the URC service to operate. From those scenarios, we derive requirements for the functionality and encoding of Uniform Resource Characteristics (URCs) within the overall architecture of Uniform Resource Identification.

The URI architecture is concerned with resources and how they will be used in applications. Resources are the objects, services, and information that applications will make use of. In order to be used, applications must have means for discovering, identifying, and retrieving resources. Resouces are named by a URN (Uniform Resource Name), and are retrieved by means of a URL (Uniform Resource Locator). Describing the resource for purposes of discovery, as well as making the binding between a resource's name and its location(s) is the role of the URC (Uniform Resource Characteristic). The URI architecture is described in other working drafts of the URI working group of the Internet Engineering Task Force, particularly in [1]. With the URI architecture in mind, we can say that:

> The purpose or function of a URC is to provide a vehicle or structure for the representation of URIs and their associated meta-information.

The next few sections present concrete examples of how we foresee URCs being used. We also describe the operation of the service in which URCs reside - the URC service. From those concrete scenarios, we derive a set of requirements on the functionality and encoding of URCs. Any proposals for the URC service will be expected to show how they meet the requirements set forth in this paper, or to point out the error of our ways.

2 User Scenarios

This section of the paper presents several scenarios of how users might interact with the URC service. In these scenarios we have attempted to show how the system will be used, without specifying how the system will accomplish its tasks.

2.1 URN to URL resolution

The fundamental purpose of the URC service is to map URNs to URLs so that a resource can be retrieved if its name is known. We believe that the most frequent operation will be to take a URN and return a (possibly empty) list of URLs where the resource named by the URN can be found. This is the primary use of the service and its speed and fault-tolerance are paramount.

- User provides a URN to the browser by clicking on an anchor or by entering text into a dialog box.

- Browser connects to the URC service and gives it the URN.

- Service returns a (possibly empty) list of locations to the browser. Each location must contain a URL. It may also contain information on Content-Type, Price, Signatures, Version, etc. The list of locations unordered. Note that if a location contains information in addition to the the URL, ordering may be used to associate the additional information with a particualr URL, but no importance should be placed on one URL appearing before another in the list of locations sent back to the browser. The means by which the URC service determines this list is outside the scope of this scenario.

- The browser uses user-configurable preferences to order the list. For example, a user might prefer HTML to PostScript to text. One user might prefer locations that carried signature information, another might not care. Most would prefer the cheapest version of a resource, and the most recent version. Estimated network distance is another means for ordering the selections. If multiple locations tie, the browser randomizes them in the list to prevent overload of any one server.

- Once the list of locations has been sorted, the browser attempts to retrieve the resource from the first location. If that fails, the next location is tried. This continues until one of the following is true:

  -- The browser successfully retrieves the resource

  -- The list is exhausted

  -- The user tells the browser to cancel the retrieval

- The browser displays the resource to the user, perhaps with the aid of an external viewer.

Note that the list of termination conditions given above is not complete. The URC service will undoubtably make extensive use of caching for speed. If the list was obtained from a cache, the possibility exists that the resolution failed because of the cache being stale. A reasonable fix is to allow the user to configure the browser to retry the query, this time getting authoritative information to fill the list of locations.

This scenario provides us with several requirements for the URC and the URC service. First, we must be able to locate a URC from a URN. We must be able to transport a URC without errors using normal Internet protocols. The URC must be parseable by a computer. It may have a hierarchical structure, and we should be able to rearrange elements of the URC within the same hierarchical level. The URC must be able to contain a wide variety of information. Furthermore, we must be able to distinguish queries answered over cached information from those answered over authoritative information.

2.2 Meta-data for its own sake

The scenario above assumed that the user really and truly wanted to retrieve the resource on the other end of the anchor. However, sometimes the user will not be sure if they want to get the resource. This is already the case in WWW browsing where users will sometimes decide, by inferring size and speed from the URL, not to access particular resources. As the WWW starts to encompass resources that will require payment for their access, users will want to know just what they about to get themselves into.

- o User is browsing and comes across a moderately interesting link.
- o User does a right-mouse-button over the link, presenting a pop-up menu.
- o User selects ``More info...'' from the pop-up and releases the mouse button.
- o Browser fetches the URC for the resource and displays it in a nicely formatted dialog box.
- o The user decides that they don't mind paying $1 for the resource, and selects ``OK'' in the dialog.
- o The browser fetches the resource and displays it to the user.

To support this scenario, the service must be able to provide an entire URC, not just the list of locations that it returned in the

previous scenario. Second, URCs must have a printable representation that can be understood and transcribed by humans. This does not mean that all elements must be easily understood, or even that we have to transmit URCs in a readable form. It merely means that, in addition to any other representation, fields must have have a printed representation that does not intentionally obfuscate the URC, barring the presence of encryption.

2.3 Ensuring the veracity of the resource

An important concern voiced over the URI mailing list and in discussions with different communities of users has been how to ensure the veracity of a resource. This concern has been raised on both the user and provider side. Users want to make sure that they are getting the real resource, especially if they are paying for it. Providers want to make sure that they are not haunted by bogus versions of a resource. To ensure the veracity of a resource, the location information provided by the URC service could carry a digital signature of the resource.

- o The user starts to retrieve a resource according to the first scenario.
- o As the browser is going through its list of locations, it notes if the current location has signature information. The rest of this scenario assumes that we successfully retrieve a resource which has signature info.
- o When the browser retrieves the resource, it displays it to the user.
- o In the background, the browser verifies the signature on the information. To do this it retrieves the appropriate public key of the publisher through a secure, ubiquitous public key service. The public key is used to decrypt the signature from the location object. It is compared with the MD-5 hash of the resource.
- o If the signature does not check out, the browser alerts the user.
- o If the user goes on to another resource before the signature computation is complete, it is discarded.

This assumes that signatures are computed over the contents of a complete file. Some resources, such as search services, can not be treated in such a fashion. One possibility would be for the URC to contain the signature of a constant header the service provides with its results. The header would contain a public key used to verify a

signature of the search results appended to the search results.

This scenario imposes the requirement that it be possible to establish an unbroken chain of authentication from a URN through the URC to the resource. Multiple signatures schemes should be supported to allow different cost/security tradeoffs to be made.

2.4 Ensuring the veracity of the URC

Resources are not the only information that can be tampered with. The URC service will provide a tempting target for attack. It needs to be secured against determined attacks and the information it provides needs to be verifiable. However, security does not come for free, and we should not impose that cost on all accesses. Therefore it is not appropriate to make the URC server compute a digital signature for every query response it generates.

One approach would be for the server to keep two pre-computed signatures for each of its URCs. The first is a signature over the entire URC, the second is only computed over the location information it would return in response to a standard URN resolution query.

- User configures the browser to verify URC information.

- The user clicks on a link

- The browser sends a URN resolution request to the URC service. The request has a flag set so that the URC server will provide digital signature information.

- The browser receives the list of locations as in the first scenario. In addition it receives a digital signature of that information which has been encrypted with the private key of the URC server.

- The browser retrieves the public key of the server, and uses it to verify the URC information.

- If there is no problem, the browser continues as before to retrieve the resource. If there is a problem the browser alerts the user, who should alert the administrator of the URC server.

If a general query is issued, the URCs for all matching resources are returned in their entirety. The browser then has to verify each of the URCs in turn. Validating general queries will be an expensive process, but it is the user's machine paying most of the cost.

This scenario requires that the URC have a consistent external representation that is suitable for the computation of digital signatures. That representation must be network friendly to the extent that it will be transmitted without any changes over standard Internet protocols. Furthermore, it must be possible to separate the portion of a URC being signed from the portion carrying the signature.

2.5 Bibliographic Search

In addition to locations, the URC provides a convenient place to store bibliographic information such as author, title, subject, date of publication, etc. Also, since publishers are assumed to be arranged in a hierarchy, it should be possible to find every publisher affiliated with the URC service. Combining these two properties opens up the possibility of bibliographic searches across the whole of the web. Exactly how this should work is not so obvious. A naive approach would be:

- o User enters author, title, and/or subject information into a form
- o Browser passes the query to the URC service.
- o Within the URC service, each node is consulted with the query, the results are collected and passed back to the browser.
- o The browser presents the search results to the user.

Of course, the scenario above is unrealistic. If every bibliographic search of every user consults every URC server, the service as a whole will soon grind to a halt. The obvious alternative is for some sites to come forward and carry the burden of these searches, similar to the current situation with Archie. Some sites will do this out of the goodness of their heart, while others may charge a fee for their services. The usage scenario is now:

- o User connects to a URC search site
- o Browser puts up the form from that site
- o User fills it in and hits ``submit''
- o The URC search site handles the query over its database and returns the result to the browser.
- o The browser displays the results to the user.

The database for handling the search is updated regularly by harvesting the network.

- Search server starts a depth-first search of the tree of publishers.

- Search server queries the current URC server for all URCs that are new or have been modified since the last time the search server visited. Those URCs are put into the database of the search service.

- Search server asks the URC server for all changes in publication hierarchy since last visit.

- Search server continues depth-first search using the new topology

The URC service will grow beyond the capabilities of all but the most dedicated sites (OCLC, Library of Congress, etc.) to keep a comprehensive index. The natural course is for search servers to only keep a portion of the URCs that exist. Exactly how they choose the subset to retain will be a decision that varies from one search service to another.

Several requirements for the URC service spring from considering searching. First, we must be able to connect to a variety of URC servers instead of having only one URC server be our gateway to the world. Second, if URN resolution is handled through a query forwarding mechanism, servers will want to distinguish between a simple resolution request (which should be forwarded) and general bibliographic queries which should not be forwarded to most sites. If URN resolution does not require query forwarding then this is not a problem. Third, there will need to be a means for determining how to contact the server administrator so that the administrator will add the central search services to the list of entities that can launch certain queries. Fourth, a publisher's server must keep a complete record of all the sub-publishers authorized by that publisher and be able to provide that list in response to a query. Fifth, we must be able to determine the parent publisher of a publisher, either from the URN or by a special request to the URC server. Sixth, the administrator of a URC server should be able to make incremental modifications to the URCs on that server. Seventh, URCs should carry information on their creation and modification dates so that incremental harvesting is possible.

2.6 Filtering by Seals of Approval

One of the interesting concepts to come out of the Interpedia effort [2] is the concept of SOAPs (Seals Of APproval). SOAPs are capsule reviews of a resource and are implemented using digital signature technology so that they will be extremely difficult to forge. Critics, professional organizations, etc. could use SOAPs to carry quick reviews of resources and to point to more elaborate reviews. For example, the IEEE might receive a request to ``publish'' a resource in one of their electronic journals. The editorial board of the journal lines up the requisite number of reviewers and sends them the URL of the resource. Each of the reviewers sends their review back to the editors, who either turn the author down flat, recommend changes, or accept the resource as it is. If the editorial board accepts it, they form a digital signature of the resource, the quick rating, an optional URN of a full review, etc. all encrypted with the private key of the particular journal.

Users could use SOAPs to augment bibliographic searches. For example, a new physics grad student might ask to see all the abstracts of all the resources dealing with (string theory AND quantum chromodynamics) which had been reviewed by the American Physical Society and received a rating of 9 or above.

Such queries do not necessarily need to proceed in the same fashion as the general bibliographic search described in the earlier section. Instead, SOAPs may well become the valuable intellectual property of professional organizations. It may be that if you wish to do searches on things with the SOAP of the APS, you have to connect to their server to do it, presumably paying them for the privilege. Given this money-making potential, it is doubtful that many professional organizations will allow authors to include a SOAP in the default URC for their resource unless the author pays for the privilege.

o User connects to the server of a reviewing organization, or to a server that has licensed the right to use the SOAPs of particular reviewing organizations.

- o Browser displays the search form of that organization. This will be a typical bibliographic search form augmented with special features for the SOAPs issued by the organization.

- o User fills out the form and submits it.

- o The server does the search and returns the results to the browser.

- o The browser displays the results of the search to the user.

This scenario imposes two requirements on the URC. First, it must be possible to extend the URC by adding arbitrary elements. In the case above, the SOAP is the new element. Second, it must be possible to ignore elements that you do not understand. For this to be possible it must be possible to determine where any particular element ends, even if you know nothing about the structure of information inside the element. Note that there is an interaction between ignorability and having a consistent representation for the purposes of digital signatures. Digital signatures are computed over the external representation, which can include experimental elements. Ignorability is a feature of the conversion from the external to the internal representation, where if we do not understand an element we are free to discard it while we are parsing the URC.

3 Provider Scenarios

3.1 Publishing a new resource

This is one of the fundamental operations for resource providers. Consequently it needs to be as simple and as bulletproof as possible.

Consider the processes of preparing and testing a new resource. Any anchors in the resource must be expressed as URNs, not URLs. However, the resource will typically undergo considerable change while it is being developed, so it is not appropriate for a community larger than the developers to be able to resolve the URNs to URLs. To meet this need, the authors request "development URNs" from the naming authority. Development URNs will have very minimal URCs. Typically they will contain zero or one URL and some access control information. These URNs are hidden from all but the developers and server administrator. Using the development URNs, the author(s) prepare and test the new resource. Note that many development URNs will never make it to the status of full URNs.

When the author(s) are ready to publish the resource to the world, they will modify the access control information in the development URC to allow wider access. They may also augment the URC with author, title, publication date, etc. The amount of information needed in the URC of a published resource will vary from one publisher to another and can be enforced by the publisher's URC software. If the resource is to be verifiable, the signature of the resource will be put into the URC at this time. Once all the material for the URC has been provided, a signature can be computed over it as well.

Once the URC information has been put onto the local URC server, it will be propagated to any other servers around the globe that can play the role of default server for that publisher.

This first publishing scenario imposes several requirements. A publisher must be able to provide developmental URNs and to shepherd the corresponding URC through the development process. Minimal URCs should be easy to generate by hand, and the URC must be incrementally modifiable. Note that a logical consequence of this is that we will want to maintain versioning information for the URC, not just the resource it describes. However, that is not foreseen to be part of a minimal URC. While in development, harvester queries should not get these URCs. Finally, note that we leave open the possibility for multiple URC servers to provide default information for the works of a particular publisher. Any proposed specifications will need to show how they meet requirements for consistency among such cooperating servers. At a minimum, a URC server for a publisher should know all the other URC servers for that publisher and be able to update their records.

3.2 Publishing a new version of a resource

When it is time to revise a resource the authors request a development version of the URC info. This version will have restricted access so that ordinary users only see the older version while the authors and URC server administrator can see the new version. Once the new version of the resource is ready, the modifications in the URC are made publicly accessible. Locations for the new version are established, and the locations for the old version should gradually go away.

This scenario requires that the developers be able to augment their resolution queries with version information so that they will be able to access the new version while the rest of the world continues to receive the old version. It also requires that access control mechanisms have a fine enough granularity in the URC to allow such a discrimination to be made.

3.3 Providing an additional location for a resource

One of the main benefits we are looking for from the URC service it the ability to have multiple locations for a resource. How are these additional locations to be established? There will be several ways this might happen, the appropriate model will depend on financial considerations more than technical ones. We will consider three cases out of many possible ones. The first is simple mirroring of free information. The second is a mirror of a small publisher's information that is sold. The third is a contractual arrangement between sites.

3.3.1 Mirroring of free information

A researcher in Australia comes across a collection of interesting technical reports on a server in Sweden. He wishes to mirror those reports as a service to the research community in Australia. He contacts the administrator of the archive in Sweden, who also happens to be the author of the reports of interest. She gives her permission for a mirror to be established. He pulls over the reports and sets them up on his HTTP server. Now that they have URLs, he sends a register_new_url message to the URC service. Since the Swedish researcher has provided a digital signature for the URCs of all the reports, a new location can not just be blindly entered. The URC service forwards the request to the Swedish researcher. She checks out the new URLs to make sure that they are faithful versions of her reports, then signs the register_new_url message with her private key before sending it back to the URC service. The service verifies the authentication information, sees that it is good, adds the new location to the URC of each report and recalculates the signature information. Now when users attempt to resolve the URN, they can fetch it from either Australia or Sweden. As a matter of courtesy, the Australian researcher periodically informs the Swedish researcher about how many times her reports have been accessed.

This scenario does not impose any requirements on the URC service beyond those already described for modifying the URC information.

3.3.2 Mirroring of information that is for sale

An experimental film maker in Germany has been selling avant-garde videos over the WWW. A film distributor in Canada contacts her to see if they can serve these up to the North American market in exchange for a cut of the action. The film maker says ``sure''. The Canadian distributor puts copies of the videos onto their video server and attempts to register the new location with the URC service. The service forwards the request to the film maker, who authorizes it by signing the request with her private key. Periodically the Canadians send the film maker a check to cover the royalties the film maker collects from every download from the Canadian server. As part of the contract between the two sites, the film maker can access the logs on the distributor's server to make sure that she is being paid for all the copies it provides.

This scenario does not impose many requirements on the URC service. Note that the access logs are an obvious point of attack for an unscrupulous mirror site administrator. It would be nice if there was a means for ensuring their veracity, however, that is an HTTP (or equivalent) server issue, not a URC server issue.

3.3.3 Mirroring on a regular basis

Some large sites may set up cooperative mirroring agreements. For example, Los Alamos National Laboratory might make arrangements with CERN to provide mirrors of each others work. When either of these sites publishes a new resource, it sends a message to the other. The second site fetches the resource and puts it on their server. It then issues a register_new_url message to the URC service. It is forwarded to the publisher of the resource, where it is automatically approved without human intervention.

This scenario requires that the URC server's access controls be capable of registering multiple users - not a big problem. It also adds the concept of a list of sites to notify when new material is published. However, that requirement could be eliminated in favor of a polling model as already discussed in the information harvesting scenario.

3.4 Removing a location for a resource

One of the other strong motivations for the URC service is to allow administrators of collections of information to rearrange their collections without breaking pages across the globe. Moving resources can be accomplished in two steps - establishing the new location then deleting the original location. Deletion is also necessary when we wish to remove a resource for any reason.

- Administrator of a resource location sends a delete_location message to the URC server. This will typically require authentication that is provided through digital signature means.

- The URC service authenticates the request. If the issuer of the request has permission, the URC is searched for the specified location. If found, it is removed and a new signature for the URC is computed.

- If there are other URC servers providing default information for the particular publisher, they are notified as well so that they may also modify their databases.

This scenario requires that we be able to select portions of a URC and delete them. Access control mechanisms should operate on a fine enough grain that the administrator of one location could delete their location from the URC, but could not delete other locations.

3.5 Establishing a new publishing authority

Publishers are arranged in a hierarchy where new publishers can be added as children of existing ones.

- o Billy Bob Riker, Harley biker, decides to publish his doggerel to the world. He contacts his friendly neighborhood web publisher who, in exchange for a modest amount of cash, establishes ``HogDog Press'' as a publisher by issuing the following request to the URC service, signed with the private key of the publisher:

- o Register_new_publisher(parent_publisher, name_of_new_publisher, signature_of_request)

- o Since Billy Bob's nomadic life style is a little hard on disk drives, he contracts with a third party to provide storage, HTTP service, and URC service. These are private business dealings between the two parties and do not especially concern us.

- o Billy Bob's prose is published to the world using the operations described earlier.

3.6 Dealing with the demise of a publisher

Poor Billy Bob. The market for Harley doggerel was not enough to cover his yearly storage fees and his service provider is about to evict his bits. While the parent publisher will never again register an entity as ``HogDog Press'' no one is paying the service provider for the machine resources, so it is time to remove the URLs from the URCs for Billy Bob's resources.

Accomplishing this in the presence of digital signatures could be a tricky question. Billy will have signed the URC elements with HogDog's private key, and he is not about to go along willingly with the eviction proceedings. Of course, he is not the administrator of the HTTP and URC servers. The administrator of the servers simply clobbers the old URC and replaces it with one that contains a ``no longer available'' element. It can't be signed with Billy Bob's key, but so what? Well, it leads to a form of denial of service attack.

Once the new URC is in place, the resources are deleted from the HTTP server.

This scenario requires us to pay considerable attention to who will sign URCs and URC components, how URCs might be nested in other URCs, etc.

4 Requirements

In each of the scenarios above we listed any new requirements that would be placed on the URC and the URC service. This section collects those requirements into 2 categories: requirements on the URC, and requirements on the URC service. Any proposed specifications for the URC and URC service will need to demonstrate how they meet all the requirements, or demonstrate how the requirements are unnecessary or in error.

4.1 Requirements on the URC

Machine Consumption A URC must be parsable by a computer.

Consistent External Representation In order for digital signatures of the URC information to work, and to simplify the requirement for parsability, the URC must have a consistent external representation.

Transport Friendliness Related to the consistency of the external representation, it must be possible to transport a URC unmodified in the common Internet protocols, such as TCP, SMTP, FTP, Telnet, etc.

Human Readability A URC must have a printed representation that is suitable for printing on paper, as well as suitable for entry by means of being typed by a user on a keyboard. Digital signature information need not apply to such items given the high probability of trivial differences.

Some meta-information items are meant for humans only while others are only meant to be machine consumable. One requirement should not preclude the other from being encoded.

Simplicity It must be simple for humans to generate correct minimal URCs that do not carry any signature information.

Rearrangeability It must be possible to reorder elements in a URC without changing their semantics. Note that elements in this case may mean compound entities. The compound entities (such as information related to a particular location) should be rearrangeable, while the information inside the entity may need to have its order preserved.

Generality In the most basic sense, a URC must be able to contain ANY conceivable type of meta-information or URI. Therefore it must be possible to add new types of elements to the URC without breaking previous applications. Any restrictions on the representational

capability of a URC will be the target of intense scrutiny.

Structure In accommodate the encapsulation of objects that are currently unforeseen, have a self-describing structure. We interpret this as meaning that elements in a URC must be tagged with a descriptive label and it must be possible to determine their extent, even in the presence of nesting.

Ignorability Related to the previous 2 requirements, an application encountering an unknown tag must be able to ignore it without error. This implies that it must be easy to tell where an element stops, even if you don't know anything about its internal structure. Also note that nesting unknown elements must still be handled correctly.

Searchable It must be possible to select a URC based on a search of its components. It must be possible to select which components will be searched and which will not be searched.

Subsettable It must be possible to form a new URC from some of the components of another URC.

Seperable It must be possible to separate a signature in a URC from the information it signs.

Incrementally Modifiable It must be possible to add (or delete) elements to (or from) a URC in an incremental fashion. It must be possible to provide elements in a URC for tracking the changes in a URC.

Versioning It must be possible for a URC to track version changes in the resource(s) it describes. This is related to, but distinct from, the requirement that a URC be able to track version changes in a URC.

Caching Caching should be possible for any URC regardless of whether or not any of its specific elements are not cacheable. Further, it must be possible to determine if a query has been answered from cached or authoritative URC information.

Grandfathering Current meta-information schemes should be allowed to work within the URC structure, where this will not conflict with the other requirements.

4.2 Requirements on the URC Service

The previous section discussed requirements on the encoding and functionality of single URCs. This section presents the requirements we have derived for collections of URCs sitting on a URC server, and

that server's communication with applications.

Resolution It must be possible for a URN to be resolved into a URC.

A URC is meant to be the format that URNs and URLs are transported in, therefore a given URN or URL may be resolved into a URC. Nothing within a URC should cause it to not be the solution to a URN or URL resolution.

Query Language It must be possible for simple resolution queries to be augmented with information on the version of a resource desired, and an indication of whether signature information should be supplied.

Security Since the URC service will be providing information of significant value, it will be a tempting target for attack. It must be possible to secure the service without imposing performance penalties on unauthenticated access to free, unencrypted, information.

Authentication Chain One aspect of the general requirement for security is that it must be possible to establish a chain of authentication from the URN, through the URC, to the resource retrieved through a URL.

Access Control Another aspect of security is that it must be possible to control (at least) read and write access to portions of a URC.

Maintenance The URI architecture is intended to be long-lived. The service must not prevent the maintenance of the resources and their meta-information.

Synchronization If multiple servers are equally authoritative for a publisher, they must work with each other to keep their URCs in sync within a reasonable time delay. This delay is certainly less than a week, but can be more than an hour.

Development URC servers must support the development of new resources by issuing and handling "development" URNs and URCs.

Choice A user must be able to connect to a range of URC servers, and not have to do all interaction through one server that is a gateway to the world.

Scalability In order for the system to scale to large numbers of users, queries on one URC server should not automatically be forwarded in such a fashion that they will hit a large number of other servers around the globe.

Administrative Contact There must be a standard means for contacting the administrator(s) of a server.

Hierarchical Operations A publisher will have the rights to register sub-publishers.

The publisher must keep a list of the sub-publishers it has created.

That list should be available as the result of an appropriate query to a server that speaks for the publisher.

It must be possible to determine the parent publisher of a sub-publisher.

5 Characteristics

Several important characteristics of URCs come about as a result of fulfilling the above requirements. Some of these characteristics are a result of requirements on URNs and URLs that make up some of the elements of a URC:

Time To Live Since a URC may contain transient information such as timestamps, access privileges, etc. it can not be guaranteed to have a Time To Live greater than 0. Any subset of a URC is free to specify its own TTL but this still does not affect the whole URC. While this does not preclude the user from attempting to trust a URC for a longer amount of time it should not be something to depend on.

Character Sets Since the encapsulation and scalability requirements force the inclusion of alternate character sets, some common scheme must be found that accommodates all character sets in a way that fulfills the transport friendly encoding requirement. This precludes any restrictions on allowable character sets.

Data Naming Fulfilling the grandfathering requirement will make it nearly impossible to specify the numerous ways extremely similar pieces of information can be represented. Thus one consideration should be a central authority that makes suggestions as to the consolidation of the names used to identify specific pieces of meta-information.

Member Element Control By allowing any piece meta-information to be included within a URC the number of globally understood elements will be small if not non-existent. Therefore some entity must have some control over some set of very concretely specified

member elements. The specification of that entity should be done in an encoding specification and is outside the scope of this list of functional requirements.

Multiple Signatures To accomodate the requirements of insertion and deletion in the presence of digital signatures, we anticipate that URCs will be signed using the private key of a server. The server's public key would be signed by the private key of the publisher. Entities with the authority to modify URC elements will have to have their keys signed by the server.

References

[1] Sollins, K. and Masinter, L., Functional Requirements for Uniform Resource Names, <URL:ftp://cnri.reston.va.us/internet-drafts/draft-ietf-uri-urn-req-01.txt>

[2] Rhine, J., Interpedia Homepage, <URL:http://www.hmc.edu/interpedia/index.html>

Glossary

Default URC The URC that is provided by the publisher of a resource.

Default URC server The URC server(s) that can provide the default URCs for a publisher.

Local URC server The URC server that a user's browser is configured to connect to as a first resort.

Development URN A URN used while developing a resource. It starts with very tight access controls so that only the resource developers and the server administrator can see the URC information and resolve the URN to a URL. The access controls can be eased later.

value-added URC server A server that provides more than just the default information on a resource. Servers run by professional organizations that provide SOAPs are one example, servers that keep full-text indices or n-grams of text in order to offer greater search capabilities are another.

SOAP Seal Of APproval - A capsule review of a resource which uses cyptographic techniques to provide guarantees on the source of the review and its authenticity.

Contact Information

Ron Daniel Jr.
MS B287
Los Alamos National Laboratory
Los Alamos, NM, USA 87545
voice: (505) 665-0139
fax: (505) 665-4939
rdaniel@lanl.gov

Michael Mealling
Office of Information Technology, Network Services
Georgia Institute of Technology
Atlanta, GA, USA 30332-0730
voice: (404) 894-1712
fax: (404) 894 9548
michael.mealling@oit.gatech.edu

This Internet Draft expires May 25, 1995.