UNIVERSITY OF WATERLOO
Software Engineering

# Analysis of Resource Identification Implementations for use in Visa's Web Services Projects

Visa International
Foster City, California

**Prepared by**
Matthew Stephan
Student ID: xxxxxxxx
Userid: mdstepha@uwaterloo.ca
3A Software Engineering
September 1, 2005

Mathew Stephan
X
X
X

September 1, 2005

Bill Wilson, Program Director
Software Engineering Department,
University of Waterloo,
Waterloo, Ontario N2L 3G1

Dear Mr. Wilson:

This report, entitled "Analysis of Resource Identification Implementations for use in Visa Web Services Projects", is the third work report that I have written. The information presented in the report is based upon my recently completed 3A work term with Visa International.

Visa International is responsible for all of the Visa credit card transactions that occur in every region in the world. As part of the technology department, specifically the research and development component, I was often working on ways that Visa can remain an industry leader by utilizing relatively new technologies, such as the Internet. One specific task that I was assigned was to assist the team working on Web Services by comparing methods for resource identification.

The enclosed report encapsulates the analysis that I was performing throughout my work term. It contains a comparison of the currently used method of resource identification and a newer method that is currently in development. The report uses similar criteria for comparison that I was using during my tenure at Visa International.

I would like to thank my team members, Gabe Wachob and Mike Lindelsee, for the assistance they provided me regarding Web Services and resource identification. I also wish to thank Frankie Stephan for proofreading my report. I hereby confirm that I received no help, other than what is mentioned above, in writing this report. I also confirm that this report has not been previously submitted for academic credit at this or any other academic institution.

Sincerely,

Matthew Stephan
Student ID: xxxxxxxx

## Executive Summary

This report performs a critical analysis of two resource-identification methods that can be used by Visa International to create a Web Service platform, should they choose to develop one. The scope of this report will extend to concerns that Visa International will need to be conscience of when choosing an approach to resource identification.

The report begins by explaining the difference between the current standard in resource identification, Hypertext Transfer Protocol Uniform Resource Identifiers (HTTP URI), and a newly developed approach, entitled Extended Resource Identifiers (XRI). Basically, XRIs add a level of abstraction to URIs, and thus provide more functionality as well as complexity. The report also contains a description of Web Services and resource identification at a high level.

One of the first areas examined is the issue of persistent identification. HTTP URIs utilize Uniform Resource Names (URN) in order to deal with this, while XRIs are able to solve it inherently. Next the difficulty of implementing each of the two methods is analyzed. It is shown that that implementing HTTP URIs would take notably less time and money to implement than XRIs. Afterwards, the problem of delegation identification is investigated. HTTP URIs perform delegation only until the authority component of the URI. XRIs, on the other hand, allow delegation of the entire resource identifier. The issue of sharing a resource over multiple systems is then considered. It is shown that XRIs deal with this by the use of cross references that allow literal strings to be shared and used across multiple systems. HTTP URIs have no equivalent at this point.

It is concluded that XRIs provide significant flexibility and versatility to resource identification in a web service project, but also are more costly to implement. Unless time and cost is a serious issue when or if Visa International decides to implement a web service platform, it is recommended that they should utilize XRIs. Otherwise, they should use HTTP URIs.

## Table of Contents

## List of Figures

## List of Tables

## 1 Introduction

Humans have an inherent fear of that which is unknown to them. This characteristic can be seen in the way that people tend to be frightened by new technologies, such as the Internet. This fear, coupled with the need to protect one's personal information, has led to notable resistance in the area of online-payment transactions. As this inertia eventually dies down, however, it is in the best interest of payment-oriented companies, such as Visa International, to find ways to utilize these new innovations in technology. One area of investigation and research that Visa International is pursuing is the area of Web Services. A web service can be defined as "any service that is available over the internet, uses a standardized extended markup language (XML) messaging system, and is not tied to any one operating system or programming language."[1] This area has become increasing popular recently due to the way it allows for interoperability between different systems without time-consuming integration efforts. Since Visa International already has a number of different systems that deal with inter-company communication, it is logical that they are pursing research in the field of Web Services.

A crucial aspect of Web Services is resource identification, that is, the ability to ascertain which component of a web service system is performing a specific action. An example of this is the need to identify the recipient or the sender of an XML message. The most commonly used approach to resource identification is the use of a Uniform Resource Identifier (URI). Prevalent examples of this are Hypertext Transfer Protocol (HTTP) URIs, such as http://www.uwaterloo.ca/, which are utilized in Internet browsers. A relatively new form of resource identification being developed is the Extensible Resource Identifier (XRI). This method adds a level of abstraction to URIs by allowing for a more extensive syntax and provides significantly more flexibility. Both of these resource-identification implementations will be discussed further in this report.

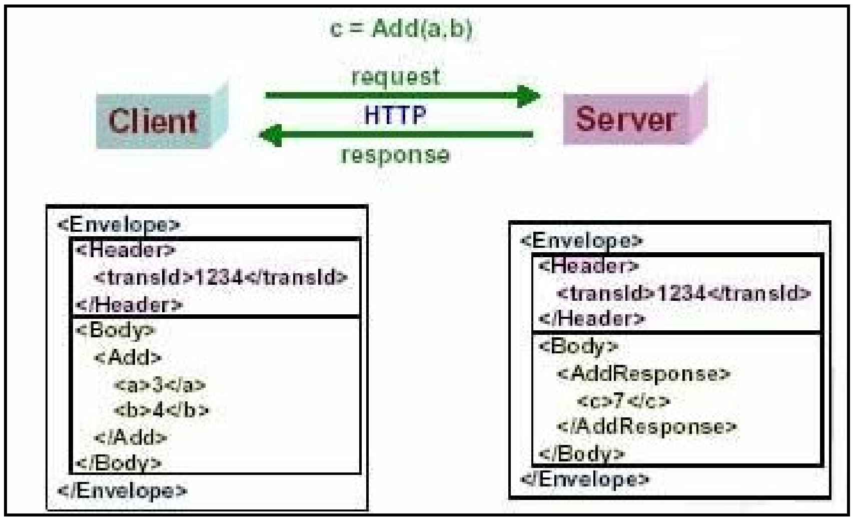
Since resource identification is such a fundamental part of Web Services, it is important that Visa International determines which implementation is best for them. This report assists them in making such a decision by providing an analysis of the two different resource-identification schemes mentioned above. The only constraint that exists when scrutinizing the approaches to resource identification in the context of a Visa International web service is whether or not it is able to satisfy the basic functionality required by a web service platform. However, Visa International is such a large and complex organization that the needs of their web-service project would be notably more complex. Therefore, the justification for choosing a resource identification method will need to have criteria that fit the needs of such a web-service project. The criteria that are going to be used to determine the ideal resource-identification method in the context of a Visa International Web Service implementation include persistent identification, difficulty of implementation, delegation identification, and shared identification. Some of these criterions are mentioned in the Introduction to XRI document [2].

The intended audience of this report is any Visa International (or other organization) group who is looking into developing a Web Services project and has yet to decide what type of resource-identification scheme should be used. In order to fully appreciate the contents of this report, a basic understanding of Web Services, XML, and URIs is required. Sufficient information will be provided on resource identification, URIs, and XRIs in order to facilitate the reader's understanding of the analysis provided. The scope of this report will only touch upon issues that Visa International will need to consider in choosing a resource-identification method if they decide to develop a web-service platform.

This report will briefly discuss the role of resource identification in Web Services, elaborate on URIs and XRIs, perform quantitative and qualitative analysis on the two schemes with respect to Visa International's needs, and, lastly, draw conclusions and make recommendations that will assist Visa International in making a decision.

# 2 Resource Identification in Web Services

As mentioned previously, one of the main allures of Web Services is the way it facilitates interoperability between varying components over the internet [1]. More specifically, those individual systems can be completely different than the other systems on a web service network and still be able to communicate to one another. This implies that there must be a uniform scheme of messages and way of identifying a message sender and recipients. XML provides a relatively simple and very effective scheme for having system-independent messages. As seen in figure 2.1, the client and the server are communicating through the use of XML messages. The XML message in the left has the client making a request to add the numbers 2 and 3 together to the server through the use of XML elements. The server then performs the needed calculation and returns the results through the XML message on the right. The important thing to note from this figure is that it does not matter what systems the client and the server are because they are using a uniform method of communicating. Both have their own ways of dealing with incoming messages.



```
<Envelope>
  <Header>
    <transId>1234</transId>
  </Header>
  <Body>
    <Add>
      <a>3</a>
      <b>4</b>
    </Add>
  </Body>
</Envelope>
```

```
<Envelope>
  <Header>
    <transId>1234</transId>
  </Header>
  <Body>
    <AddResponse>
      <c>7</c>
    </AddResponse>
  </Body>
</Envelope>
```

**Figure 2.1 Simple Web Services Example. Provided by [3]**

The issue of uniform resource identification, such as identifying the client and the server in the example just discussed, does not yield such a clear-cut solution. For a web service platform, the resource identifiers must be able to identify message participants but should also allow identification of any other resources in the platform, including other resources or subcomponents that may need to be referred to.  The resource-identification method needs to provide a precise and unambiguous form to identify all components of a web-service platform.  Everything on the platform must be able to be uniquely identified. While URIs satisfy this, there are many other issues that have arisen that may not be able to be dealt with by URIs.  For a complex web-service system, such as one that would need to be created at Visa International, there are many more aspects to consider.   Issues such as persistence, implementation difficulty, delegation, and others need to be considered, and are done so in this report.

# 3 Means of Identification

In order to accomplish resource identification in Web Services, a standard scheme needs to be used that is system independent and facilitates identification of all components within a given platform. The most common method used today is the HTTP URI, however, there is currently work being done in the development of XRIs

## 3.1. HTTP URIs

HTTP URIs, most commonly used when referring to a website in an Internet browser, is the combined use of the HTTP protocol and URIs. The first component of an HTTP URI is the identification of the protocol, which is in the form of 'http://'. This is followed by a URI. What many people do not think about is that the address is actually pointing to a specific resource located somewhere on the Internet and that the URI resolves to this resource. [1] URI representation of a resource allows for retrieval of the resource from anywhere within the Internet irregardless of the system requesting the retrieval.[4]

As demonstrated in figure 3.1.1, the URI component is placed after the protocol identification and can be broken into subcomponents. One way of thinking about the URI is that it represents a linear chain of resolve requests. Using the URI in the figure as an example, the resolution steps are as follows; www.usa.visa.com is resolved, personal is resolved, student is resolved, and index.jsp is resolved. The personal resource is actually a resource under the www.usa.visa.com resource. Therefore a personal resource in an URI like www.uwaterloo.ca/personal is different because it is in the www.uwaterloo.ca context rather than the www.usa.visa.com context.

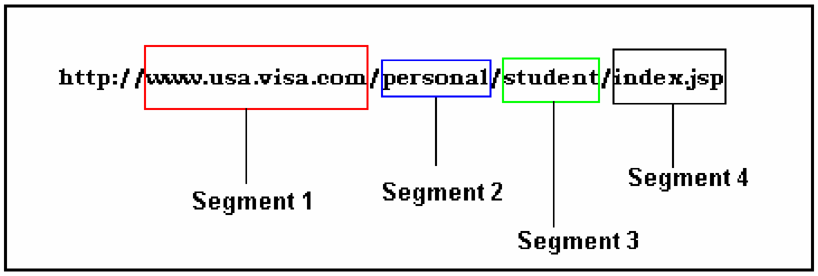
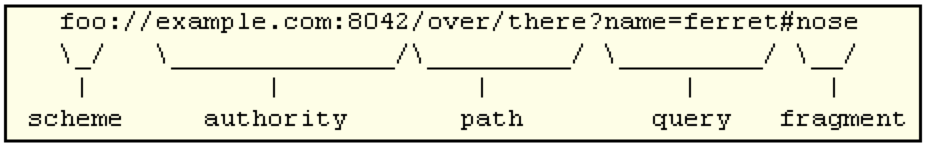


**Figure 3.1.1 Linear resolution of a simple URI's segments.**

A more complex URI is shown in figure 3.1.2 that highlights the different components of a URI. As demonstrated, the initial component is referred to as the authority segment and is, optionally, followed by a path, query, and a fragment. The authority element and path elements are separated by a '/' character, a query is indicated with a '?' character and a fragment is identified by a '#' character. The fragment component allows for a particular part of a resource to be retrieved. The query element allows for basic logic to be implemented into URIs by allowing for a varying value to be placed in the URI. [4]



```
  foo://example.com:8042/over/there?name=ferret#nose
  \_/   \______________/\_________/ \_________/ \__/
   |           |            |            |        |
scheme     authority       path        query   fragment
```

**Figure 3.1.2 Breakdown of a complex URI. Provided by [RFC]**

## 3.2. XRIs

XRIs are a newly formed way of performing resource identification. Developed by the Oasis organization, XRIs add a level of abstraction to URIs in an attempt to allow for more functionality regarding identification and resolution of resources. As exhibited in figure 3.2.1, XRIs are actually built on URIs and Internationalized Resource Identifiers (IRIs), which are URIs that allows for all ASCII characters instead of a set of them. [2]

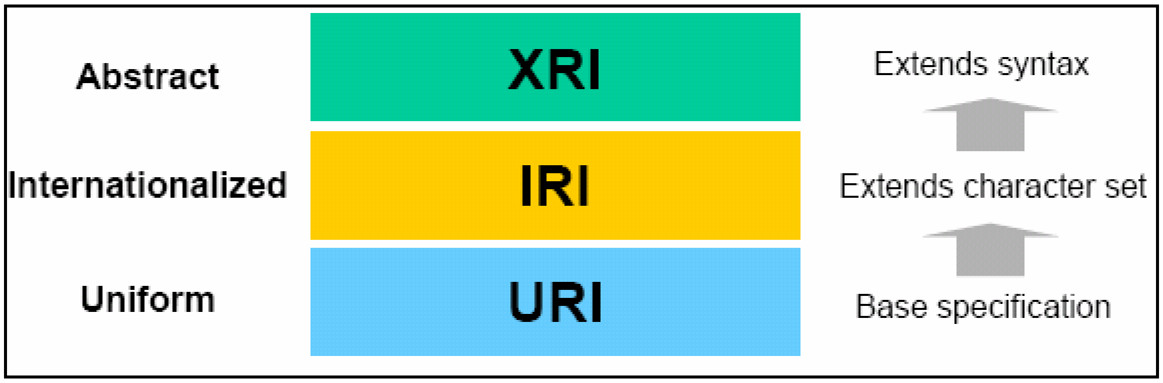


**Figure 3.2.1 The relationship between URIs, IRIs, and XRIs. Provided by [2]**

### 3.2.1. XRI Syntax

The syntax for an XRI has the same overall form as a URI, that is, an authority, followed by a path, followed by a query, and a fragment. XRIs begin by identifying their protocol scheme, which is always ‘xri://’. The authority then must begin with one of the characters listed in Table 3.2.1.1. This character identifies the global context of the authority. So, for example, the @ character is used to indicate that the global context of the authority is an organization or a component of an organization. It is then followed by a number of XRI segments until the first ‘/’ character, which denotes the end of the authority component. XRI segments are strings that are delaminated by the character ‘*’ in a similar manner that the ‘/’ character is used in a URI.

**Table 3.2.1.1 Global Context Symbols for XRI. Taken from [5]**

| Symbol Character | Authority Character | Establishes Global Context For |
|---|---|---|
| = | Person | Identifiers for whom the authority is an individual person. |
| @ | Organization | Identifiers for whom the authority is an organization or a resource in an organizational context. |
| + | General Public | Identifiers for whom the authority is the general public, i.e., that represent generic “dictionary” concepts for which there is no specific authority. (In the English language, for example, these would be the generic nouns.) |
| $ | Standards body | Identifiers for whom the authority is a specification from a standards body, for example, other XRI specifications. |

Following the XRI authority component, is the path component, which is in fact identical to a URI path except that the individual path elements can contain either a ‘*’ or a ‘!’ character. The ‘*’ character is used to identify a resource that is not persistent and “may be reassigned by an identifier authority to represent a different resource at some future date.”[5] The ‘!’ character, on the other hand, is used to identify a resource that is persistent and will not be reassigned. The query and fragment components of XRIs are identical to URI components [5]

Figure 3.2.1.1 shows an example of an XRI. The authority element is composed of the ‘@’ global context symbol meaning we are dealing with an organization, the organization named ‘organization’, and a component of the organization called ‘component’. Therefore, the authority of the remaining elements of the resource identifier is the component belonging to the organization. Note that the first star is omitted, as per the syntax, in order to promote readability. The first path segment, ‘names’ begins with a ‘!’ character meaning that it is persistent and will never be reassigned. Because the second path segment is surrounded by parenthesis it is what is known as a cross reference and is interpreted as a literal string.[5] This will be elaborated on later in the report when the issue of shared identification is discussed. After analyzing this XRI, it likely refers to a resource entitled ‘first.name’ in the names resource of the component belonging to the organization resource.

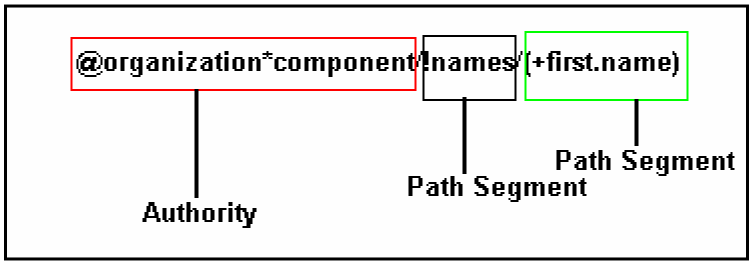
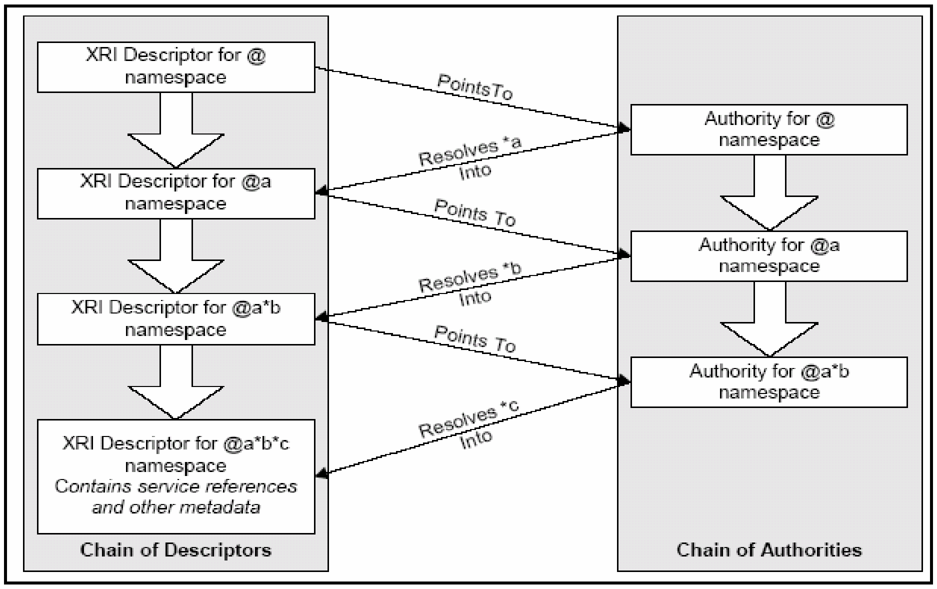


**Figure 3.2.1.1 Example of an XRI**

### 3.2.2. XRI Resolution

The last aspect of XRI that must be explained is authority resolution. As shown in the previous example, an authority can contain many segments. XRI resolves the authority component in a very unique way. Each segment of the authority resolves to a XRI descriptor (XRID), which is a specific form of an XML document. Each segment’s XRID contains information about where to resolve to next as well as other information.[6] Figure 3.2.2.1 shows the resolution process that is performed to resolve the authority @a*b*c. First the XRID contained at the ‘@’ location is resolved and directs the resolution to the ‘a’ segment. That segment resolves to an XRID that contains the resolution for the ‘*b’ segment and the process continues. The entire authority is resolved in this linear manner until the path character, '/', is reached and the desired resource can be accessed.


XRI Descriptor for @ namespace
XRI Descriptor for @a namespace
XRI Descriptor for @a*b namespace
XRI Descriptor for @a*b*c namespace
Contains service references and other metadata
Chain of Descriptors
PointsTo
Resolves *a Into
Points To
Resolves *b Into
Points To
Resolves *c Into
Authority for @ namespace
Authority for @a namespace
Authority for @a*b namespace
Chain of Authorities
**Figure 3.2.2.1 Resolution of @a*b*c. Taken from [6]**

# 4 Analysis: HTTP URIs versus XRIs

The only constraint that Visa International would need to consider for a resource-identification method in a Web Services project is that it facilitates the previously discussed requirements for resource identification inherent in a web service platform. URIs are already used in the industry and have proven that they meet this requirement. XRIs are built directly on URIs, thus, it can then be deduced that XRIs also meet this requirement. [2] However, if Visa International wanted to choose the best resource-identification approach for their use then they would have to consider other criteria.

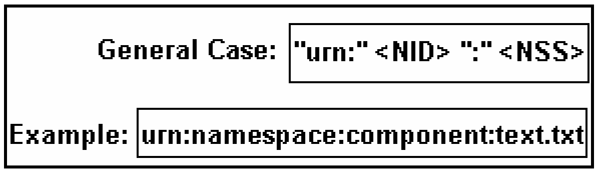
## 4.1. Persistent Identification

One issue that currently exists regarding resources is the problem of a moved or deleted resource that is still being refereed to by an identifier. For example, let us say that there is a text document that is identified by the identifier http://www.usa.visa.com/textfiles/text.txt. If the authority of this resource decided to change the ‘textfiles’ resource to ‘textfile’ then the identifier would become a “broken link”, that is, it would identify nothing. If Visa International was to pursue a web-service project, then they should be concerned about the resource-identification method’s ability to deal with this problem.

### 4.1.1. HTTP URIs

This issue is addressed through the creation of a Uniform Resource Name (URN). A URN is a “persistent, location-independent, resource identifier.”[7] It works by having a resource assigned a permanent URN and allows URI references to that URN. By definition, a URN is a URI. [7] In order to enforce integrity it utilizes the concept of namespaces to ensure that there is no collision between two or more URNs.

Figure 4.1.1.1 illustrates both the general case for URN and an example of a URN. The general case begins with the protocol identification “urn:”. It is then followed by a namespace identifier (NID), a “:” character, and the Namespace specific string (NSS). The managing of namespaces is what ensures that persistence is enforced. The specific example has a namespace of “namespace”, a child of that namespace entitled “component”, and a namespace specific string entitled “text.txt”. Semantically, this URN represents a persistent resource of namespace’s component entitled text.txt. Therefore, one could be directed to the HTTP URI of text.txt by informing them that the persistent resource’s URN name is “urn:namespace:component:text.txt”.



General Case: "urn:" <NID> ":" <NSS>

Example: urn:namespace:component:text.txt

**Figure 4.1.1.1 URN Syntax and URN Example. General Case provided by [7]**

## 4.1.2. XRIs

XRIs provide all the functionality of URNs except that they also allow identification of non-persistent identifiers. [2] That is, rather than needing URIs and URNs to deal with both persistent and non-persistent identifiers, XRIs handle both by employing the concept of an abstract layer. Using the characters mentioned previously, the “!” and “*” characters, XRIs can resolve different elements according to whether or not they are persistent.

Figure 4.1.2.1 demonstrates a resource that is being refereed to by both a persistent XRI and a non-persistent XRI. For the non-persistent XRI, any of the individual components can be renamed assuming that the XRID associated with that component is modified. In this example, since the XRIs refer to the same resource, they are known as synonyms of one another.

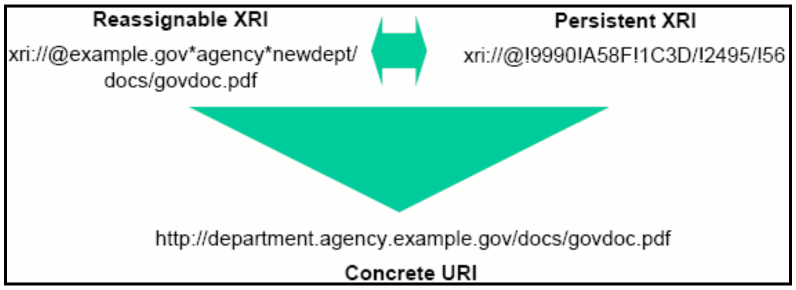

Reassignable XRI
xri://@example.gov*agency*newdept/docs/govdoc.pdf
Persistent XRI
xri://@!9990!A58F!1C3D/!2495/!56
http://department.agency.example.gov/docs/govdoc.pdf
Concrete URI

**Figure 4.1.2.1 Persistent and Non-Persistent XRI resolving to a URI. From [6]**

From a numerical perspective, a XRI's ability to be either persistent or modifiable implies that a web-service platform would be able to store half as many resource identifier entries and improve performance time. That is, rather than storing URNs and URIs and having the system map the URIs to the URNs, the platform can store only the persistent XRIs and provide those to the constituents of the platform. Thus, time is saved by not having to do the URN to URI lookup and space usage is cut in half by eliminating the need to store two different resource identifiers per persistent resource.

## 4.2. Difficulty of Implementation

Since the HTTP URIs method of resource identification is the most commonly used one today, it is evident that using HTTP URIs is the easiest approach to implement. Developers are already familiar with it and other systems at Visa International already utilize it. So it then becomes a question of how much more difficult and time consuming it is would be for a Visa International web-service project to accomplish resource identification via XRIs.

Focusing on developers, it is necessary to consider the time it would take them to learn and understand the eccentricities of XRIs. For the developers currently working at Visa International who have been looking into XRIs, it took them roughly two weeks of reading through documents and simulating practical examples to acquire a solid grasp of the fundamentals. Translating this to cost, it implies that a cost of two weeks worth of salaries times the number of developers working would need to be incurred in order to have the developers learn XRIs.

Regarding interoperability, the majority of systems at Visa International right now are utilizing URIs. Since XRIs are built on top of URIs, the amount of work required to get the pre-existing systems operable on a platform utilizing URIs would be minimal. [2] However, if the systems are going to be upgraded to use XRIs then this would require more effort. The XRI Syntax document [5] contains a thorough description of the process required to convert an XRI to a URI and vice versa. It illustrates a number of different transformation steps that allow for conversion. The Visa International developers experimenting with XRIs found that it took nearly half an hour to convert an URI to a XRI, including testing and verification. Using this as a guideline, it would take half an hour times the number of URIs in a specific system to transform a single system. Thus, the cost for a single system would be developers salaries per hour times half the number of URIs in the system.

## 4.3. Delegation Identification

The concept of being able to delegate resources to specific components is an important aspect of Web Services. Delegation is referring to the ability to indicate what component of a web platform is responsible for a particular resource. Since Visa International's web-service project would likely be one of notable complexity it would need to have an effective way of handling delegation.

### 4.3.1. HTTPS URIs

Currently, HTTP URIs provide complete delegation of resources for the entire authority component of the URI. In fact, this is what has made HTTP URIs so successful up to this point. [4] Using an existing name system, such as Domain Name System (DNS), HTTP URIs lookup each component of the authority segment, separated by the "." character, until the authority is complete resolved.

Figure 4.3.1.1. illustrates a HTTP URI that has multiple segments in an authority being looked up. As demonstrated, lookup is performed from right to left. Firstly, the "com" DNS name server is reached and "visa" is looked up.  Once found, the Visa DNS name server is resolved and "usa" is looked up.  This continues till the beginning of the authority is reached.  As a result, DNS and HTTP URIs facilitate delegation, but only for the authority segment of a URI.

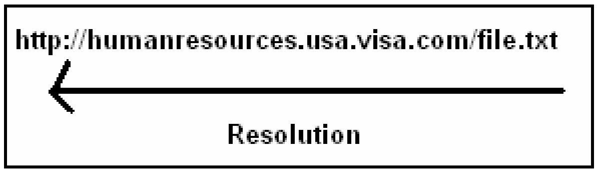


**Figure 4.3.1.1 DNS and HTTP URI resolution**

### 4.3.2. XRIs

XRIs provide the same functionality of DNS and HTTP URIs, but accomplish it in a different fashion.  Also, they allow for resolution and delegation past the authority component, that is, XRIs provide delegation for path components. [2]

XRIs are resolved from left to right and point to a specific XRID.  Using the previous example as a basis, the XRI equivalent is @visa*usa*humanresources/file.txt. The XRI would be resolved in the manner discussed earlier in this report.  At this point, the XRI and the URI are equivalent.  Let us say, however, that the authority wants to delegate the file to a specific resource under their authority, such as their San Francisco office.  Using the XRI @visa*usa*humanresources/*sanfrancisco*office/file.txt will accomplish this.  The path component, *sanfrancisco*office, provides delegation of the file to the San Francisco office.

This is especially useful from a web services perspective because it allows significantly more flexibility in the way that resources can be handled within a given platform.  By allowing delegation past the authority component, platform designers now have the option of choosing where and how resources are stored.  Quantitatively, DNS and HTTP URIs can lead to crowding of an authority's namespace with too many resources.   This can cause problems such as namespace collisions and added complexity for system developers.   The added delegation provided by XRIs significantly reduces the number of single resources that need to be stored in a specific namespace.

## 4.4. Shared Identification

A significant problem that exists in many web service platforms is the issue of being able to deal with resources that are in different contexts but need to be integrated and able to work together.  As put in the XRI Introduction document, "As long as a resource is only identified in a local context, then each additional system with which it must be shared requires an additional mapping between the two contexts." [2] Mathematically, if system one and system two have a resource that both need to identify, then it requires two mappings; system one to system two and system two to system one. Add another system, system three, then there would need to be a mapping of system one to system two, system one to system three, system two to system one, system three to system one, system two to system three, and system three to system two.  As can be deduced, there is a significant number of mappings required in order to share a resource between multiple systems if it is only identified locally in each system.

XRIs attempt to combat this issue via the use of cross references.  A cross reference, syntactical noted by text inside of parentheses, allow "…identifiers to be shared across contexts." [5]  The text inside of the parentheses is interpreted as a literal string and is uniform across all systems that need that identifier.  Each system can then interpret that literal text as they see fit.  An example of this would be a social security number.  Let us say we have three systems; System 1, identified by the XRI @government*departments*IRS; System 2, identified by the XRI @government*departments*immigration; and System 3, identified by the XRI @government*departments*DMV.  If we were searching for information from all three systems with a specific social security number as the identifier, an ideal solution would be to have a reusable identifier that is uniform across all three contexts.  By appending !(123-45-6789) to the end of each of the aforementioned system XRIs, we can get all of the resources we want based on that number. The “!” character was used because a social security number is an example of something that is persistent.

Figure 4.4.1 illustrates the amount of mappings required in a three-system example that uses HTTP URIs and a three-system example that uses XRIs with cross references. In the later, each system needs only one mapping in order to be able to share a resource with each other and utilize data based on that resource.   As seen in the diagram, the amount of mappings required can become quite significant, on the order of n to the power of two. Thus, the ability of XRIs to reduce the number of mappings is quite important to web-service platform developers.

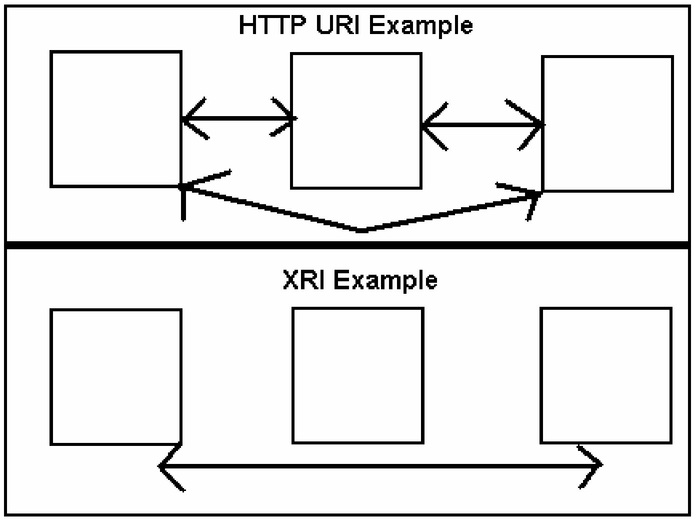


**Figure 4.4.1 Mappings required for a Shared Resource**

## 5 Conclusions

Although HTTP URIs are the current standard in resource identification, it is clear that there are many imperfections that exist. XRIs attempt to combat these shortcomings by adding a level of abstraction, but, by doing so, also add to the complexity.

Both resource-identification methods are able to deal with resource-location persistence and identification of delegation, all of which are prevalent issues in Web Services. It should be noted that the HTTP URI solution for delegation (DNS lookup) is somewhat limited in comparison to the XRI solution because delegation cannot occur beyond the authority component of the URI.

One area where HTTP URIs fall short is the issue of shared identification. XRIs provide a very efficient way of dealing with this issue via the use of cross references, while nothing has yet to be developed for HTTP URIs. However, HTTP URIs do have the advantage of being notably simpler than XRIs. As indicated in the report, the time and cost of implementing HTTP URIs in a web service platform would be significantly less than implanting XRIS in web service platform.

## 6 Recommendations

Although HTTP URIs take significantly less time and cost to implement, the benefits provided by XRIs are far too great to pass up. If Visa International was to develop a web-service platform, the complexity of such a platform would likely entail the use of the features provided by XRIs. As touched upon in the report, the cost will be higher by doing this, but the result will be a much more flexible and versatile web service. By using HTTP URIs, it is almost inevitable that Visa International would come across the same problems that are currently plaguing most of the web service platforms in place today.

The ability of persistent identification provided by XRIs would allow for the platform to have consistent references and significantly reduce the amount of invalid links that are reached. Using HTTP URIS to do this instead of XRIs would mean storing twice as much data. Since a platform developed by Visa International would likely have significant traffic over small periods of time, it is imperative that the links are always valid and XRIs make this easier.

Since a Visa International platform would need to interact with a significant amount of other components, the XRI feature of shared identification would be extremely useful. The cross-platform identification ability would seriously minimize the amount of mappings required to successfully deal with a shared or persistent resource.

The only instance where HTTP URIs may be appropriate is if cost and time are serious factors in Visa International's decision to develop a web service platform or if they wanted to develop a very simplistic prototype. However, as mentioned in the report, if Visa International later on decides to switch to XRIs, then this would yield a cost of notable time and money.

## References

[1] E. Cerami, *Web Services Essentials*, O'Reilly Publishing, February 2002.

[2] D. Reed and D. McAlpin, “An Introduction to XRIs”, March 14 2005; http://www.oasis-open.org/committees/download.php/11857/xri-intro-V2.0-wd-04.pdf (last accessed September 1, 2005)

[3] A.Asaduzzaman, “Building XML Web Services with PHP NuSOAP”, February 6 2003; http://www.devarticles.com/index2.php?option=content&task=view&id=414&pop=1&page=0&hide_js=1 (last accessed September 1, 2005)

[4] T. Berners-Lee et al., *Uniform Resource Identifier (URI): Generic Syntax*, IETF RFC 3986, January 2005; http://www.gbiv.com/protocols/uri/rfc/rfc3986.html (last accessed September 1, 2005)

[5] D. Reed and D. McAlpin, “Extensible Resource Identifier (XRI) Syntax V2.0”, March 14 2005; http://www.oasis-open.org/committees/download.php/11852/xri-syntax-v2.0-cd-01.pdf (last accessed September 1, 2005)

[6] G. Wachob, “Extensible Resource Identifier (XRI) Resolution V2.0”, March 14 2005; http://www.oasis-open.org/committees/download.php/11853/xri-resolution-V2.0-cd-01.pdf (last accessed September 1, 2005)

[7] R. Moats, *URN Syntax*, IETF RFC 2141, May 1997; http://www.ietf.org/rfc/rfc2141.txt (last accessed September 1, 2005)

## Acknowledgements

I would like to take this opportunity to thank Mike Lindelsee and Gabe Wachob of Visa International for providing me with significant information on web services and resource identification which allowed me to create this report and perform the analysis within.

I would also like to thank Frankie Stephan for proofreading this report.