U.S. Department of Justice (DOJ)

Universal Lexical Exchange (ULEX) 1.0

User Guide

Revision 1

Change History

Rev	Date	Author	Description of Revision
1	4/2/2008	Jack Wallace	Initial version

Acknowledgments

This document was developed through a collaborative effort sponsored by the U.S. Department of Justice (DOJ) Law Enforcement Information Sharing Program (LEISP) and written by the Georgia Tech Research Institute (GTRI), as a product of a project funded by the Office of Justice Programs (OJP), U.S. Department of Justice,

The following individuals have attended our project meetings, assisted on this research effort, provided input, and reviewed this report:

Debby Park - GTRI
Benjamin Shrom - GTRI
Jack Wallace - GTRI
John Wandelt - GTRI

Jeremy Warren, CTO - DOJ / LEISP
Boris Shur, Chief Data Architect – DOJ / LEISP
Sudhi Umarji - DOJ / LEISP / Trusted Federal

Table of Contents

1 ULEX Conceptual Model

1.1 ULEX Overview

ULEX provides a flexible framework used for the creation of schemas for information sharing, both for publishing information and for system-to-system federated searches. ULEX is used by **programs** to develop standard information sharing schemas for use by their constituent **communities** or agencies. For the purposes of this document, standards developed at the program level are referred to as "ULEX-based standards".

ULEX specifies a set of schemas that establish consistent definitions supporting publication, search, and retrieval, at a structural level. ULEX data elements and structures are organized into components that support the underlying actions required to accomplish the publication, search, and retrieval operations fundamental to information sharing. All ULEX-based standards augment the organizational elements provided by ULEX with well-defined, community-specific structures that allow a common understanding of the data to be shared within the community.

ULEX utilizes a generic paradigm for information sharing called the **data item**. A data item is whatever the source considers a logical unit of information. For example, to an incident based reporting system, a logical unit of information is an incident report that may contain activities, people, places, and things. To a logistics system, a logical unit of information may be details about a single piece of hardware, along with its maintenance history and location. The data item concept provides a single, generic container that can be used to encapsulate different types of data needed by various communities. The data item can be thought of as a collection of structured entities, attributes of these entities, relationships between these entities and unstructured textual information.

Since no set of schemas can support all communities due to varying needs, ULEX supports multiple **levels of understanding**. Each ULEX-based standard defines its own base level of data that all implementations understand, referred to as the **digest**, plus a well-defined extension mechanism that allows constituent communities to plug in additional content, referred to as **structured payload**, without impacting anyone's understanding of the base level. Since all implementations for a given ULEX-based standard understand the base level, communities only have to develop an application once in order to be able to share information with all other implementations based on the same ULEX-based standard. Where additional content is required for a selected community, modules can be written to deal specifically with information in the structured payload rather than rewriting the implementation for a completely different exchange.

Each ULEX-based standard defines a high level set of commonly understood, structured **base objects** which may represent real-world objects such as person, location, or vehicle or more abstract objects such as reports or documents.

1.2 ULEX Information Environment

In this specification the term **data source** is used to refer to a system that is operated by an organization that publishes information to a data repository. The data repository that receives and

ingests the published information is referred to as a **data consumer**. The data source publishes periodically as determined by operations guidelines. A **service provider** is the system or application that provides access to the data. When two organizations mutually allow for two-way sharing of information, with each party retaining ownership and possession of its information they are partners in information sharing and each of the systems is referred to as a **partner system.**

In the ULEX model, queries are performed between partner systems. When a user forms a query by describing the information he is looking for, he gets back a list of data items which match that query.

The term ULEX PD (Publication and Discovery) applies to the interface between data sources and data consumers, while the term ULEX SR (Search and Retrieval) applies to the interface between partner systems. It is possible that in the future a given system is both a data source and a partner system, though no such ULEX-compliant system exists today.

In addition, it must be noted that we have used the term **data source** to refer to a system that is operated by an organization that publishes information to a data repository. In general, unless explicitly stated otherwise, the assumption is that the data source "owns" the information that it publishes. It is indeed also possible for the data repository to republish information that it has previously received from another source. When a data repository republishes information that it does not own, it is acting as an aggregator and to distinguish this case from one in which a data source only publishes what it owns, we use the term **data submitter** to refer to aggregators. We use the term **data owner** to refer to the original owner of the information. A service provider may include data from more than one data owner.

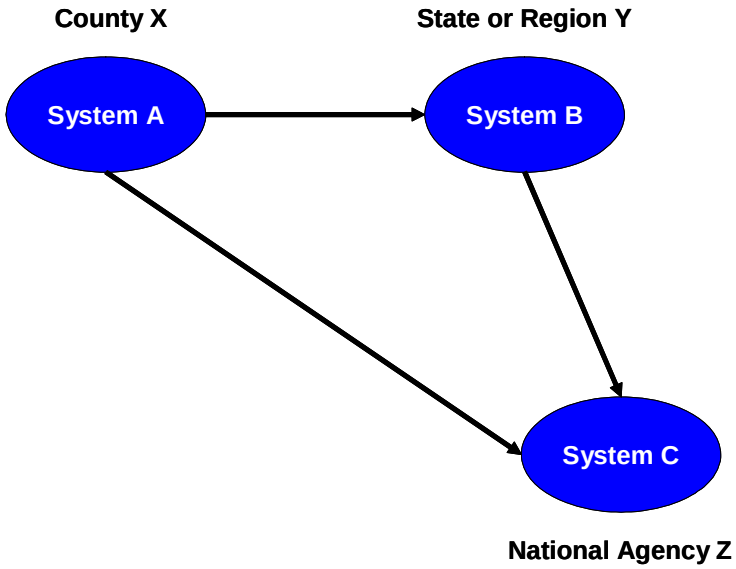

Figure 1. Data Submitters vs. Data Owners

In Figure 1, system B may be a data submitter to system C for information owned by system A. For a particular piece of information that was published to C, through such a route, A is the data owner and B is the data submitter. In addition, system A may also submit directly to C in which case, it is both data owner and data submitter for information published in that operation. See Appendix C - Data Owner/Submitter and Message Origin/Destination for additional details.

1.3 ULEX Supported Operations

Two basic categories of operations are defined by separate ULEX schemas which are supported by shared schemas that provide common definitions of ULEX structures.

ULEX Publication and Discovery (ULEX PD) is for publishing and updating data from a source to a consumer.

ULEX Search and Retrieval (ULEX SR) is for system-to-system federated searches and allows result drill-down to obtain more detailed information.

1.3.1 ULEX PD

ULEX PD supports the action of publishing sharable information to a data repository. ULEX PD provides the structures required to represent the data and metadata associated with publishing data. The data repository receives and records the published information, generally, for the purpose of analysis, making it easier, faster, and less expensive to share data. In publishing data, a source system submits one or multiple data items in a single, one-way action to a data consumer; ULEX does not define an acknowledgment action.

1.3.2 ULEX SR

ULEX SR represents a number of requests and corresponding responses that support data search and retrieval. ULEX SR supports requests that fall into two broad categories, data requests and service provider requests.

ULEX data requests provide the mechanisms for retrieving data from partner systems and include four specific requests: text search, structured search, data item request, and attachment request.

ULEX service provider requests ask for system information about partner systems including capabilities, data owners, and availability.

1.3.2.1 Data Requests

In ULEX, search and retrieval of data is a multi-step process. A search request uses information that broadly identifies the target being sought and returns possible candidates for the user to examine further. In general, a search query is followed by a data item or attachment request. For instance, a search request for information about a person might be issued using a first and last name. Usually this is not enough information to specifically identify a single person. As a result of the search, the queried systems respond with information related to any person having the specified first and last name. The intent is that the user can narrow the search by reviewing the

search response and then request more detailed information on a specific data item or a limited set of data items.

ULEX SR provides two search requests. A **structured search request** looks for specific elements with certain values, such as a first name of "John" and a last name of "Smith." A **text search request** is used to search for a value in any context, that is, no element name is specified. Depending on the implementation, the text search could be performed on unstructured data, such as a report, or on structured data, such as a name or narrative element. For example, a text search for the phrase "John Smith" on unstructured data might find a match in a report. A text search on structured data might find "John Smith" in a structured field defined specifically for a person's full name.

The response from either a structured search or a text search can contain information about data items and/or attachments, such as mug shots and documents, associated with the search parameters. A **data item request** is used to get details on a specific data item. An **attachment request** asks for one or more Attachments described in a search response. Either a data item or an attachment request is a follow-up query to a search request and uses the unique identifier supplied by the search response to identify a single target in a single partner system.

A search response may be sparsely populated and data items may contain only elements that are commonly used to help discriminate one data item from another. A data item response should contain all available details and contextual information associated with the requested data items.

1.3.2.2 Service Provider Requests

For service provider requests, ULEX defines requests and corresponding responses that ask for specific types of information. For instance, one request allows a data consumer to ask for information about a service provider's data owners. The response contains the list of data owners available, including whether the data owner supports structured searches, unstructured searches or both. There are additional requests and responses defined to determine the capabilities and the availability of the service provider. These requests can be issued alone, preceding or following any other ULEX request.

1.4 Key Concepts

There are a number of key concepts that underlie the design of ULEX. To understand how to use ULEX it is essential to understand these fundamental building blocks.

1.4.1 Message

The **message** is the basic structure that wraps ULEX requests and responses. In reality, ULEX defines a number of different message structures for different purposes, but all message structures contain metadata, and usually additional information specific to the request or response they support. For example, a Publish Message includes one or more data items plus attachments and message metadata. Search response messages are similar but may also include warning or error messages, identifiers that can be used to "drill down" for additional details, and contain information about attachments without the actual attachments. Other messages are defined for retrieval of specific data items and attachments, for querying data, and requesting system information from a service provider. The details of the various messages and other

ULEX structures are discussed later in the document. The Publish Message in the diagram below is shown as an example and provides a basis for the discussion of other key ULEX concepts.

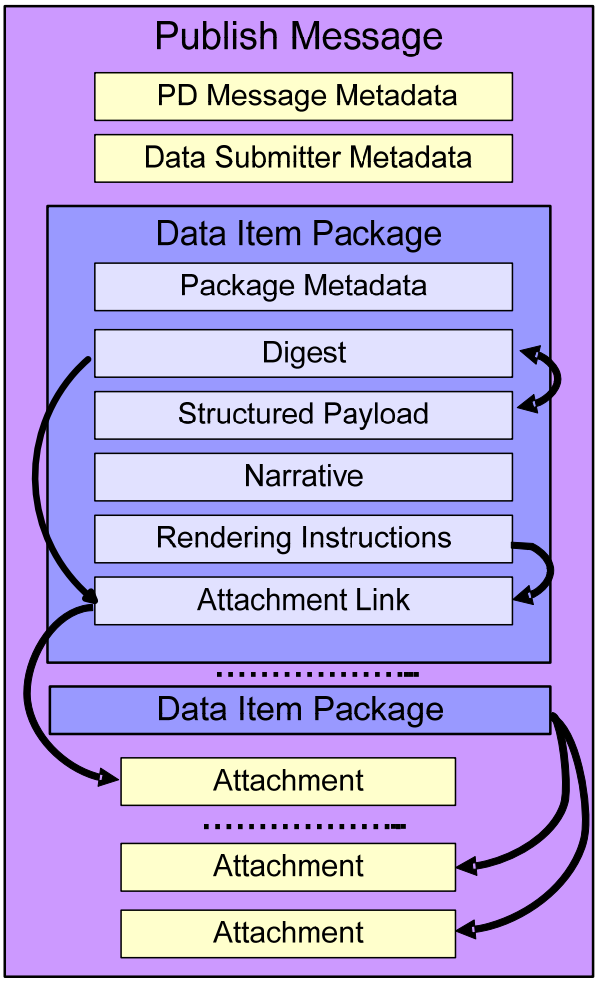

Figure 2. Publish Message

1.4.2 Package

In ULEX, the term **package** refers to a standard representation of any data item used for information sharing. ULEX defines two similar package structures to support data item publication and data item retrieval. Both package structures contain metadata, a Digest, and optionally multiple Structured Payloads, Rendering Instructions, and/or Attachments. As mentioned above, ULEX uses the term data item to refer to the standard, generic, flexible unit of information sharing as it may exist at the source system. For instance, data item might refer to a record in a database. When an application extracts this data item or record and encodes it in a format that is compliant with the ULEX schema, the data item is contained within a package structure. The data source may or may not encode all the information in the data item into the package based on the business rules that it must follow. Conceptually, a package represents a unit of information that is self-contained and able to convey the knowledge from the data source to the data consumer or between partner systems. The data source (owner of information)

ultimately decides what information is relevant and should be shared, while the definition of a package in ULEX allows the data source to map such information to a standard format.

1.4.3 Metadata

Metadata, which is information about the data being shared, exists at several levels and is organized into different structures based on content and usage. For example, the Publish Message illustrated in Figure 2 includes PD Message Metadata and Package Metadata. The PD Message Metadata contains the ULEX version used within the document, the ULEX-based standard and its version, the data and time of the message, a message sequence number, and an indication of the sensitivity of the data being shared. The Package Metadata provides a unique identifier for the package, contact information for the individual who can provide additional information about the data item, data owner information, dissemination criteria, and a number of other pieces of information about the data item contained within the package. Some metadata structures are specific to one ULEX component and others are used in a number of ULEX components. For instance, the PD Message Metadata is specific to the Publish Message, but Package Metadata is used in all messages that contain data item packages.

1.4.4 Digest

The concept of the **Digest** is central to the power of ULEX as a framework for information sharing. A ULEX data item may represent any kind of underlying information—a report on a person, an incident, a shipping manifest. The Digest is the common anchor for systems to use to handle this heterogeneous data without having to understand the specific context and meaning of the source. As long as the entities relevant to the packaged data item are represented in the Digest, users will be able to discover, link, map, etc. the information within.

Even though the Digest provides the common level of understanding, it does not mean that all sources have to populate all elements, or that all consumers have to use all elements; merely that at a schema level all applications understand the Digest. Implementers only need to build one module in order to produce or consume a basic set of data understandable by many. It also means that implementers do not have to develop large applications for each exchange, but rather build one that handles the basics and then additional smaller modules in order to produce or consume more complex exchanges.

Each ULEX-based standard defines their own base level of understanding, incorporating the appropriate objects for their needs. The objective of the Digest is to present the most common characteristics of real-world objects that can be supported by any (including the least sophisticated) data source or data consumer. Digest-level data objects may be further augmented or described with additional details in the Structured Payload or the unstructured Narrative portion of the package.

1.4.5 Structured Payload

The **Structured Payload** provides an extension mechanism so that different communities can "extend" the Digest to incorporate richer data sets without compromising compatibility across applications. Therefore, users of a ULEX-based standard are not limited to just the high level entities provided by that standard, but can define Structured Payloads to build upon the contents of the standard's Digest.

Each Structured Payload is based on schemas created by communities outside of the program that created the ULEX-based standard. Ideally these independent schemas are built to be aware of the ULEX Digest, that is, Structured Payload entities build on and link to ULEX Digest entities. Structured payloads can be ignored if not recognized/understood/implemented by consumers. Exchange instances do not have to include any Structured Payloads. Each Structured Payload includes metadata that specifies what community defined the Structured Payload schema. Consumers can use that metadata to determine if the contents can be processed or needs to be ignored. If the consumer can understand Structured Payloads from that particular community, then the content of the Structured Payload is extracted and processed based on the schema for that particular Structured Payload.

Since Structured Payloads are based on separate schemas, Structured Payload instances must be valid as standalone instances. While the data contents of the Structured Payload may not make logical sense by itself without the contents of the Digest, the Structured Payload instance must be valid against the Structured Payload schema.

Exchange instances can include multiple Structured Payloads and a consumer might understand some and not others. Structured payloads can build upon the contents of the Digest, or even upon the contents of another Structured Payload. Some consumers may understand everything in an instance, others may understand the Digest and one Structured Payload, and still others may only understand the Digest.

Each ULEX-based standard includes a mechanism to connect elements in the Structured Payload to elements in the Digest. In general, ULEX is designed so that Structured Payloads augment the contents of the Digest; it is not expected that Structured Payloads include Digest contents plus additional elements.

1.4.6 Narrative

The Narrative structure provides a container for **unstructured data** associated with the data item. The ULEX provision for unstructured data allows for the inclusion of data sources that are text-based or elements from any data source that are free-form in nature and can not easily be represented as attribute-value data components.

1.4.7 Rendering Instructions

Rendering instructions are used to display the information in a package in a specific viewing or output format for human users. A data source may want to influence the way in which the information it shares is viewed by a user at a consuming system. Using a specific rendering allows a source to present data items more intuitively to users who may not be familiar with the underlying context of the data. This is especially true when the package contains information that is specific to an agency or community and the data source is knowledgeable about how best to format the content to make it more comprehendible and user friendly. Since ULEX allows for the sharing of heterogeneous data from different sources with different contexts, user interfaces for displaying ULEX data will often need to be very generic, sacrificing intuitiveness.

Rendering instructions come in two general forms: instructions (i.e., XSL) for converting the XML data in the package into a suitable display format, and instructions to display a pre-rendered version of the data item (i.e., an attached image or document). The HTML rendering format is frequently the choice for viewing reports online and other rendering formats, such as Adobe Acrobat Portable Document Format (.pdf) files, are ideal for distributing large page-oriented reports that can be sent to a printer (assuming of course, that printing is authorized) or viewed using an Acrobat Reader.

1.4.8 Attachment and Attachment Link

ULEX Attachments are based on the NIEM Binary element, and may contain data-related binaries such as a mug shot or fingerprint, or may contain binaries or stylesheets used by ULEX Rendering Instructions. Attachments are at the message level rather than the package level since each may be part of multiple packages, such as a message with multiple incident reports that include the same person and his mug shot.

An Attachment Link provides a mechanism to connect an element in a package, such as a person, to an Attachment, such as a mug shot.

1.4.9 Levels of understanding

The ULEX Digest concept allows data sources to encode a well-defined representation of objects. ULEX also includes a layered mechanism for communities to define objects that are not defined in the Digest for a given ULEX-based standard. Communities that need additional data not provided in the Digest can supply that data in one or more Structured Payloads. This allows groups or projects who are interested in information sharing to leverage a ULEX-based standard as a base while developing their own specialized schemas targeted to address their own business missions.

This layering concept provides multiple **levels of understanding** by allowing communities to extend the Digest of a given ULEX-based standard with community-specific Structured Payloads. Layering and levels of understanding are best illustrated through concrete examples. The Department of Justice Law Enforcement Information Sharing Program (LEISP) has an LEISP Exchange Specification (LEXS) and the discussion below describes the ULEX concepts of layering and levels of understanding as they could apply to LEXS exchanges.

In the LEXS program, an incident community needs to exchange information beyond what is available in the Digest and, thus, creates its own Structured Payload schema that builds upon what is in the Digest. Bomb and arson, and cyber communities review the Digest contents along with the incident Structured Payload schema and determine that what the incident community has provided doesn't include everything they need, but that it supplies a good foundation. So rather than creating their own Structured Payload that just builds upon the Digest, each develops an individual Structured Payload schema that builds upon what is in both the incident Structured Payload and Digest. There is no limit to how many layers there can be and yet another specialized community, the terrorism bomb and arson community, can utilize the bomb and arson payload that in turn builds upon the incident payload. The arrest community reviews what is available, and determines that the incident payload is not really suitable, and so develops its own Structured Payload built on the Digest. The federal arrest community also reviews what is

available, and determines that the arrest payload provides a better foundation than any other available payload, so the federal arrest community builds a payload that in turn builds upon the arrest payload. These layers and how they build upon each other are shown in Figure 3.

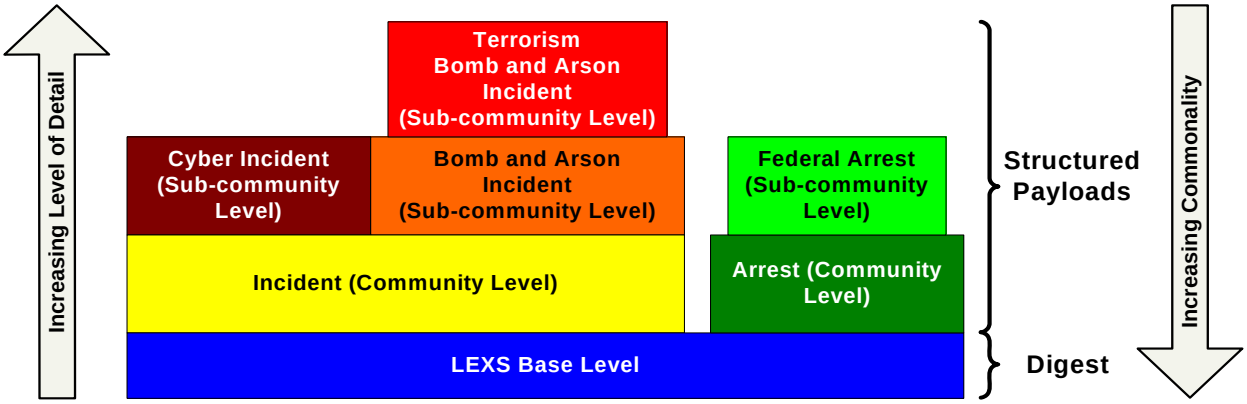

Figure 3. Structured Payload Layering

To illustrate the levels of understanding, the diagram above can be used in conjunction with a data element example.

The LEXS implementation of a ULEX Digest defines a person to include a person name, using the NIEM Person and PersonName. The Digest's PersonName only includes first, middle, last and full name. If the Incident community determines that they need a person's title as well, their Structured Payload would include a PersonName with NIEM's PersonNamePrefixText. So the Incident community can build an instance that represents a person with the name Dr. John Doe. If the Cyber Incident community determines that the Incident Structured Payload plus the Digest has most of what they need but they also need a hacker name, then the Cyber Incident community's Structured Payload might include a new element called HackerName. So the Cyber community can represent Dr. John Doe with hacker name Doc.

If the Cyber community sends such an instance to the Arrest community, the Arrest consumer would be able to determine that it does not understand either the Cyber or Incident Structured Payloads, but would still understand John Doe. So in this case, the Arrest community only understands one level, the Digest.

If the Cyber community sends the same instance to the Incident community, the Incident consumer would be able to determine that it understands the Incident Structured Payload, but not the Cyber Structured Payload. So the Incident consumer would understand Dr. John Doe. So in this case, the Incident community understands two levels, but not the third.

If the Incident community sends an instance to the Cyber community, the Cyber consumer could understand everything in the instance, such as Dr. John Doe. But an Incident Community instance would never include a hacker name. So in this case, the Cyber community understands both levels available from the producer (the Incident community), but cannot get information in the Cyber communities third level.

Appendix A – Search Result Paging Support

There may be cases where a search query results in more hits than can be returned, either because the user requested fewer or because the service provider limits the number of hits that can be returned. Users, or applications acting on behalf of a user, must be able to specify a maximum number of hits that should be returned.

Limiting the number of hits

doTextSearch and doStructuredSearch requests include a mandatory element called MaxItemMatchesRequested. This element allows the query to specify the maximum number of hits that should be returned in the response. Note that the service provider may return less than this either because of fewer hits or because the service provider has a lower limit for the number of hits that can be returned.

If a service provider has more hits than can be returned in a single response, the service provider limits responses based on the type of search performed. Responses to structured searches are limited based on dates, with the most recent being returned for the initial query. Responses to unstructured searches are limited based on whatever relevancy score is used by the service provider, with the most relevant being returned for the initial query.

For all responses, the service provider populates three elements.

- ServerLimitFlag, a mandatory Boolean element to indicate whether the number of hits was limited due to a service provider limit. True means the number returned was limited due to a server restriction.

- MaxItemMatchesRequested, a mandatory numeric element, which contains the value supplied in MaxItemMatches in the doStructuredSearch or doTextSearch. This makes the quantity originally requested readily available to any client application consuming the response without having to refer to the original query.

- NumberItemMatches, a mandatory text element to indicate how many matches there really were. This is a text field so that the service provider does not necessarily have to determine an exact number, which may require retrieving a large number of hits. For example, if 20 hits were requested and there are really 50 matches, the service provider can count the actual number of matches and populate NumberItemMatches with the value 50, or the service provider can skip counting all the matches and return the string 20+ to indicate that there were more than 20.

Service providers may support the capability to return additional hits when all hits cannot be returned in the initial response. Since some service providers may not want to retain any state information about searches and responses that have already been processed, ULEX SR provides a mechanism to support the request for additional hits without requiring service providers to maintain state information. However, since other service provider may retain state information, ULEX also supports an identifier that can be used to facilitate requests for additional hits.

If supported by the service provider, when there are more hits than can be returned in the response, the service provider populates an optional text element called MatchEndPoint. This can be utilized in a follow-up query to get the next set of hits. If the response is from a "next" query, the service provider populates an optional text element called MatchBeginPoint. These point elements are used by a service provider on a "next" or "previous" query to determine what set of data to return, and could be timestamps, or relevancy scores, or record numbers; but are only intended to be meaningful to the service provider.

Service providers that retain information about what hits were returned in response to queries can populate an optional text element called ServiceProviderSearchID with a value that is meaningful to the service provider for returning additional hits. This allows the service provider to utilize any state information it may retain so that it does not have to rerun the query again to get the next or previous set of results. Service providers define this value; requesters merely copy the value back into the follow-up query.

Requesting additional hits (next/previous)

If supported by the service provider and the user's client application, additional hits may be requested if there are more hits than can be returned in the initial response. So for example, if the maximum number of hits that can be returned is 20, and there are 50 hits in total, the next 20 can be requested (hits 21-40). Once hits 21-40 are received, the previous 20 can also be requested (hits 1-20) or the next 20 (hits 41-50 in this example).

In order to get the next set of hits, the client application submits the same search query that was used to generate this set of results, but also populates an optional text element called MatchBeginAfter to indicate to the service provider where the next set of hits should start. MatchBeginAfter is populated with the value of MatchEndPoint in the response the user is following up.

In order to get the previous set of hits, the client application submits the same search query, but also populates an optional text element called MatchEndBefore to indicate to the service provider where the previous set of hits should end. MatchEndBefore is populated with the value of MatchBeginPoint in the response the user is following up.

In either case, if the response that is being followed-up included the element ServiceProviderSearchID, then the follow-up query must populate the ServiceProviderSearchID with the value provided in that response. Service providers define this value; requesters merely copy the value back into the follow-up query. Note that a service provider may provide different values for this element for each set of hits; for example, the service provider may supply one value that indicates to it that the set returned was hits 1-20, and a different value to indicate that another set includes hits 21-40.

Examples
Original query requests no more than 20 hits, and there are 50 matches. Namespace aliases are left out to simplify examples. MatchEndPoint, MatchBeginPoint, MatchBeginAfter, and MatchEndBefore are shown here as timestamps, but could be anything meaningful to the service provider. Sequence shown may change in the actual schemas.

Example 1
Service provider does not retain any state information about previous searches and responses, and supports next and previous queries.

Initial search result
<ServerLimitFlag>false</ServerLimitFlag>
<MatchItemMatchesRequested>20</MatchItemMatchesRequested>
<NumberItemMatches>50</NumberItemMatches> <!-- could be populated with 20+ -->
<MatchEndPoint> 2006-12-01T09:30:47.0Z</MatchEndPoint>

Follow-up search for the next 20 hits
(includes same search criteria as initial search)
<MaxItemMatches>20</MaxItemMatches>
<MatchBeginAfter>2006-12-01T09:30:47.0Z</MatchBeginAfter>

Second set of results
<ServerLimitFlag>false</ServerLimitFlag>
<MatchItemMatchesRequested>20</MatchItemMatchesRequested>
<NumberItemMatches>50</NumberItemMatches> <!-- could be populated with 20+ -->
<MatchBeginPoint>2006-11-30T09:30:47.0Z</MatchBeginPoint>
<MatchEndPoint>2005-12-01T09:30:47.0Z</MatchEndPoint>

Follow-up search for the previous 20 hits, which is actually the original 20
(includes same search criteria as initial search)
<MaxItemMatches>20</MaxItemMatches>
<MatchEndBefore>2006-11-30T09:30:47.0Z</MatchEndBefore>

Example 2
Service provider does retain state information about previous searches and responses, and does support next and previous queries.

Initial search result
<ServerLimitFlag>false</ServerLimitFlag>
<MatchItemMatchesRequested>20</MatchItemMatchesRequested>
<NumberItemMatches>50</NumberItemMatches> <!-- could be populated with 20+ -->
<MatchEndPoint> 2006-12-01T09:30:47.0Z</MatchEndPoint>
<ServiceProviderSearchID>ABC123</ServiceProviderSearchID>

Follow-up search for the next 20 hits
(includes same search criteria as initial search)
<MaxItemMatches>20</MaxItemMatches>
<MatchBeginAfter>2006-12-01T09:30:47.0Z</MatchBeginAfter>
<ServiceProviderSearchID>ABC123</ServiceProviderSearchID>

Second set of results
<ServerLimitFlag>false</ServerLimitFlag>
<MatchItemMatchesRequested>20</MatchItemMatchesRequested>
<NumberItemMatches>50</NumberItemMatches> <!-- could be populated with 20+ -->
<MatchBeginPoint>2006-11-30T09:30:47.0Z</MatchBeginPoint>
<MatchEndPoint>2005-12-01T09:30:47.0Z</MatchEndPoint>
<ServiceProviderSearchID>ABC124</ServiceProviderSearchID>
<!-- doesn't have to be different ID, up to service provider to determine how to track -->

Follow-up search for the previous 20 hits, which is actually the original 20
(includes same search criteria as initial search)
<MaxItemMatches>20</MaxItemMatches>
<MatchEndBefore>2006-11-30T09:30:47.0Z</MatchEndBefore>
<ServiceProviderSearchID>ABC124</ServiceProviderSearchID>

Example 3
Service provider does not support next and previous queries.

Initial search result
<ServerLimitFlag>false</ServerLimitFlag>
<MatchItemMatchesRequested>20</MatchItemMatchesRequested>
<NumberItemMatches>50</NumberItemMatches> <!-- could be populated with 20+ -->

Appendix B - Error/Warning Categories

The following warning and error categories are provided in an enumerated list for the AdvisoryCategory element. The list below breaks them out separately for warning and error for organizational purposes; the enumerated list in schema is not separated into warning versus error lists or sections. Note that some warnings might become errors. For example, if an unsupported element is provided in a search query and that is the only element provided, the query cannot be processed and the result should be an error; versus an unsupported element that can be ignored with the result based on remaining elements.

For Errors:

- Network failure (e.g.. cannot contact service provider)
- Invalid Response (response from service provider cannot be processed)
- Invalid Request (service provider could not process incoming request)
- Timeout (service provider did not respond in allotted time)
- Business Rules Not Met (e.g. service provider requires SSN in query, but query didn't provide)
- Next/Previous Not Supported
- Structured Search Not Supported (structured search submitted, but service provider or data owner only support unstructured searches)
- Unstructured Search Not Supported (unstructured search submitted, but service provider or data owner only support structured searches)
- Fuzzy Match Not Supported
- Wildcards Not Supported (service provider does not support wildcard queries in unstructured searches)
- Logical Operators Not Supported (service provider does not support logical AND/OR queries in unstructured searches)
- Phrases Not Supported (service provider does not support exact phrase queries in unstructured searches)
- Invalid Attachment Requested (service provider received request for invalid Attachment)
- Invalid Data Item Requested (service provider received getDataItem request for data item that does not exist)
- Other Error (for errors that don't fall into categories above)

For Warnings:

- Query Object Not Supported (e.g. query was on vehicle and service provider doesn't handle vehicles)

- Query Field Not Supported (e.g. query was on SSN and service provider doesn't include SSN)
- Query Operator Not Supported (e.g. query included a wildcard and service provider only does exact)
- Quantity Mismatch (user asked for maximum # of hits, service provider supplied different #)
- Pointer Information Only (service provider can only return contact info for follow-up, but no data)
- Multiple Values Not Supported (search query included multiple values for a search field, but service provider only supports single values)
- Other Warning (other issues that don't fall into above categories)

Appendix C - Data Owner/Submitter and Message Origin/Destination

ULEX PD explicitly labels the data so that there is no confusion about who the owner is versus a submitter that may not own the data. ULEX SR search responses also explicitly indicate the owner, versus the responding system, versus the system that submitted the data to the responding system. The different terms and their definitions and usage are discussed below.

Organization and Placement

Data Owner is for an individual data item for both PD and SR.

Data Submitter is for an entire message for PD since all submissions in a single message are from the same submitter. Data Submitter is for an individual data item in SR. For SR, service providers may not retain information about the submitting system, so in this case Data Submitter is optional.

Message Origin and Data Submitter for PD are synonymous, so Message Origin is not included in PD. For SR requests, the Message Origin indicates the system making the request. However, for SR responses, the Message Origin indicates the responding system, which may be returning data from multiple submitters and/or multiple data owners.

Message Destination applies to SR and specifies the service provider the message is intended for.

Scenarios and Usage

This section provides examples of how the Owner, Submitter, Origin, and Destination are used and populated.

For PD, assume that SysA owns a record, and submits to SysB which then submits to SysC, which in turn submits to SysE. In this case, SysA must be specified as the Owner in all cases. The Submitter is the "last" submitter in the chain; so for the example above, when SysA submits to SysB, SysA is the Owner and Submitter, but when SysB submits to SysC, SysA is still the Owner, but SysB is the Submitter. This is illustrated in top portion of Figure 4 below.

For SR, assume that SysF performs a search that matches the data submitted by SysA above, and that SysD directly submitted another record to SysE that also matches the search criteria. As is shown in Figure 4 below, the query will have SysF as the Origin and SysE as the Destination; while the response will have two hits, with SysA and SysD, respectively, shown as Owners. Since the SysA record was submitted to SysE by SysC, the response shows SysC as the Submitter. SysE is shown as the response's Origin since SysE is the one actually providing the hits to SysF, and SysF is listed as the Destination in the response.

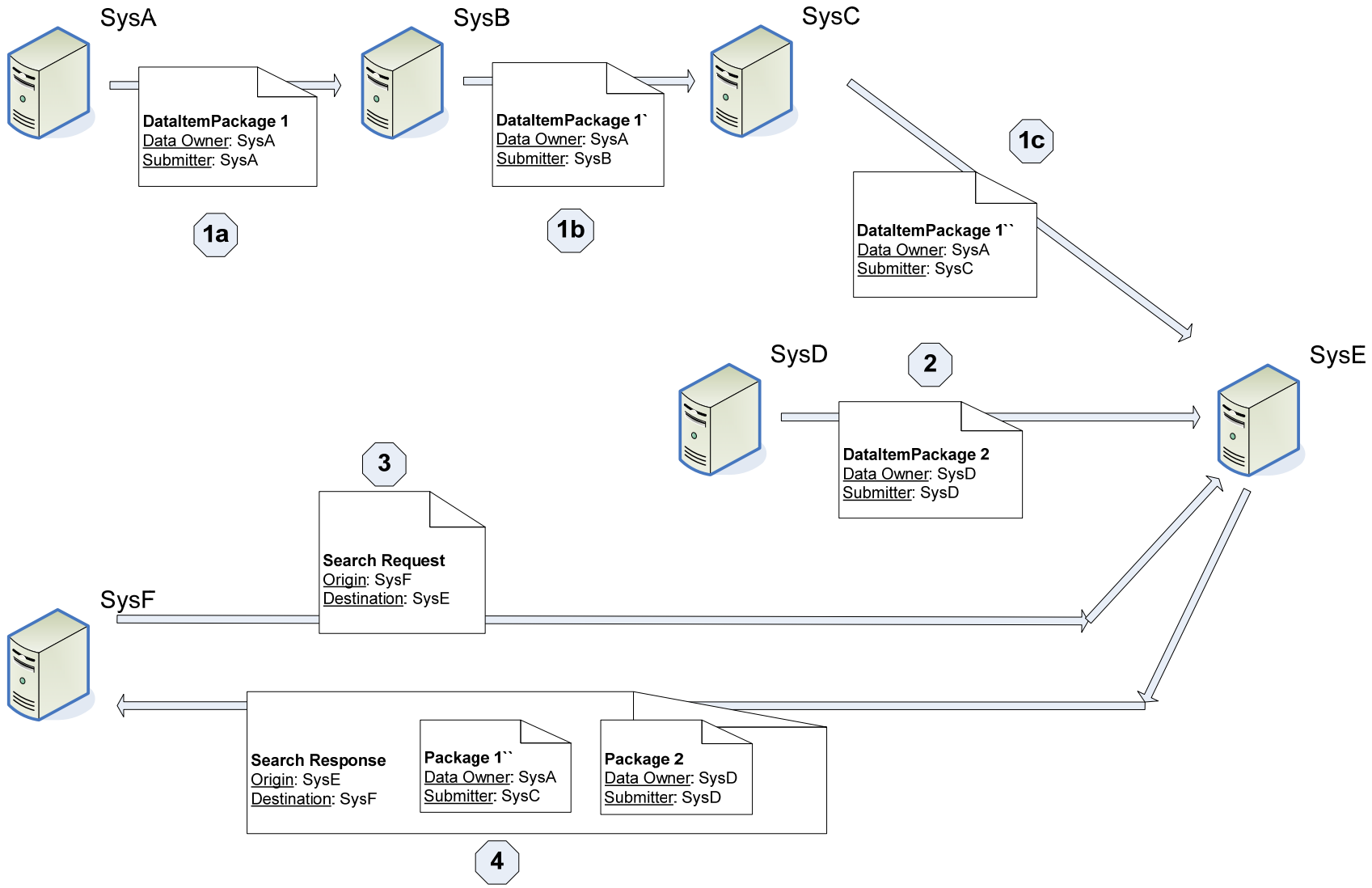

Figure 4 - Example of Data Owner, Data Submitter, Message Origin, and Message Destination

Step 1) SysA data submitted to SysE through a chain of submissions
- Step 1a) SysA submits to SysB
  - SysA as owner (in DataItemPackage/PackageMetadata/DataOwnerMetadata)
  - SysA as submitter (in DataSubmitterMetadata)
- Step 1b) SysB submits that data to SysC
  - SysA as owner (in DataItemPackage/PackageMetadata/DataOwnerMetadata)
  - SysB as submitter (in DataSubmitterMetadata)
- Step 1c) SysC submits that data to SysE
  - SysA as owner (in DataItemPackage/PackageMetadata/DataOwnerMetadata)
  - SysC as submitter (in DataSubmitterMetadata)

Step 2) SysD data submitted directly to SysE
- SysD as owner (in DataItemPackage/PackageMetadata/DataOwnerMetadata)
- SysD as submitter (in DataSubmitterMetadata)

Step 3) SysF queries SysE
Request message has SysF as message origin, SysE as message destination

Step 4) SysE returns hits to SysF, one from SysA and one from SysD
Response message has SysE as message origin, SysF as message destination
Search response includes 2 packages
- Package 1
  - SysA as owner (in SearchResultPackage/PackageMetadata/DataOwnerMetadata)
  - SysC as submitter (in SearchResultPackage/DataSubmitterMetadata)
- Package 2
  - SysD as owner (in SearchResultPackage/PackageMetadata/DataOwnerMetadata