

A-DAS

A-DAS Interface Specification

Document Information	
Document ID:	A-DAS-IS-001
Version:	0.2
Originator	<i>Roger Wallace</i>
Approval Date:	
Approver:	<i>Colm Hayden</i>

Abstract
<p><i>The Anaeko Data Agility Server, A-DAS™ is a real-time data access service designed to provide uniform and simplified data access across heterogeneous Data Sources. The primary means of interacting with A-DAS™ is through its unique HTTP interface. This document provides an introduction to and description of the A-DAS™ HTTP interface, including the HTTP Data Service API and the supported data and MIME types. The Interface is approached from two distinct perspectives: 1) accessing the A-DAS™ Services using a common Web Browser and 2) accessing the Service programmatically using the RESTful API.</i></p>

Distribution
<i>Anaeko</i>

History			
Version	Modified By	Date	Description
<i>0.2</i>	<i>Roger Wallace</i>	<i>16/02/2010</i>	<i>Still in Draft</i>

Table of Contents

1	Objectives and Scope	4
2	References and Abbreviations.....	5
2.1	Abbreviations.....	5
3	Anaeko’s Data Agility Server – A-DAS™.....	6
3.1	Data as a Service.....	6
3.2	Unified Data access through a Uniform Interface	7
3.3	Data Agility and Customised Views	8
4	Accessing A-DAS™ in a Browser	9
4.1	A-DAS™ Home Page.....	9
4.2	How do I find out what Data Services are available?.....	10
4.3	How can I explore the data provided by a Service?	11
4.4	How can I access the data?	13
4.5	Can I see what Data Views are available?.....	14
4.6	How do I use a Data View?	15
4.7	How do I make ad-hoc Queries?	17
4.8	Can I get the data in different formats?	19
4.9	Do I have to use a Browser?.....	23
4.10	What is the status of the service?.....	25
5	A-DAS™ REST API	28
5.1	Hubs and Nodes.....	29
5.2	Accessing data using a View	31
5.2.1	Limiting the results returned.....	35
5.2.2	Requesting different Media Types.....	36
5.3	Ad-hoc Querying	38
5.4	Working with Views.....	40
5.4.1	Creating a View	41
5.4.2	Deleting a View	42
5.4.3	Editing a View	43
6	Appendix A: URL Catalogue	44
7	Appendix B: XML Formats	47
7.1	Microformats	47
7.1.1	<link>.....	47
7.1.2	<table>	50
7.1.3	<thead>	52
7.1.4	<tbody>.....	52
7.1.5	<th>.....	53
7.1.6	<tr>.....	55
7.1.7	<td>.....	56
7.2	Proprietary Formats	57
7.2.1	<response>	57

7.2.2	<data>	60
7.2.3	<metadata>.....	61
7.2.4	<properties>	63
7.2.5	<relatesTo>	64
7.2.6	<property>	65
8	Appendix C: Glossary	68
8.1.1	General Terms.....	68
8.1.2	System Specific Definitions.....	70

1 Objectives and Scope

This document is an introduction to and specification for the main HTTP Interface of *Anaeko's Data Agility Server, A-DAS™*. It introduces the key design principals of A-DAS™ and assumes no prior experience or knowledge of Data Services or RESTful design. The introductory sections will enable a User to navigate the A-DAS™ Interface and access data using a Web Browser while the later sections should contain sufficient detail to enable an Engineer to program a client for A-DAS™ services.

Audience

Sections up to and including Section 4 - Accessing A-DAS™ in a Browser - are intended for Users of the A-DAS™ Web Interface. Section 5 - A-DAS™ REST API - and the Appendices are intended for Engineers looking to consume Data Services.

Organisation

Section 3 - Anaeko's Data Agility Server – A-DAS™ - Introduces the key concepts behind RESTful design and Agile Data Services. Section 4 - Accessing A-DAS™ in a Browser – begins with the A-DAS™ Home Page and describes how to explore the A-DAS™ Web Interface. Section 5 - A-DAS™ REST API – describes in detail the data access API and how to access and manage Data Views programmatically. The Appendices Appendix A: URL Catalogue and Appendix B: XML Formats complement Section 5 with a complete reference to the A-DAS™ API. Appendix C: Glossary defines the terms used in this document, providing the necessary background for reader less familiar with HTTP, REST and Data Services.

2 References and Abbreviations

Unless otherwise stated, the latest document version must be consulted in all cases.

1. *A-DAS™ Query Language Specification*
 Roger Wallace, [TBC]
 A-DAS-DS-0001
2. *A-DAS™ Product Specification v2.0*
 Roger Wallace, 08-09-2009
 A-DAS™-PS-0001

2.1 Abbreviations

<u>REST</u>	Representational State Transfer
<u>MIME</u>	Multipurpose Internet Mail Extensions
<u>URL</u>	Uniform Resource Locator
<u>URI</u>	Uniform Resource Identifier
<u>JSON</u>	JavaScript Object Notation
<u>CSV</u>	Comma Separated Values
<u>RDF</u>	Resource Description Framework
<u>RSS</u>	Really Simple Syndication
<u>XSL</u>	Extensible Style-sheet Language
<u>XSLT</u>	Extensible Style-sheet Language Transformation
<u>AJAX</u>	Asynchronous JavaScript and XML
<u>SQL</u>	Structured Query Language

Table 1 Abbreviations - detailed definitions are available in Appendix C: Glossary

3 Anaeko’s Data Agility Server – A-DAS™

Anaeko’s Data Agility Server, A-DAS™, has been created to solve a fundamental integration problem, to **enable data**, freeing it through simple, accessible services that scales from the individual User to Enterprise Applications.

At its core integration is about reuse, the need to communicate and share information, to reuse resources - both data and systems. Yet existing integration solutions tend to focus on the interfaces between systems, not the data that they contain. The hidden cost of deploying an integration solution is often an increase in complexity; inhibiting reuse. Data shared between the integrated systems is not freed, it is still **firmly locked within the application silos**. Enterprise Architects are now realising that they cannot build the next generation of Services and Cloud based solutions with data as a second class citizen.

A-DAS™ focuses on the Data, on delivering **the data that is needed in the format that is expected**. A-DAS™ provides a system designed from the ground up to support reuse. With A-DAS™ there is no costly and lengthy consultation nor is there an inflexible unified Data Model, to constrain the data to a single version of the truth. A-DAS™ provides **simple, unified access** to existing sources of data through the introduction of light touch Services and customised Data Views, specifically created for the task at hand. **A-DAS™ promotes pay-as-you-go economics for data integration**.

3.1 Data as a Service

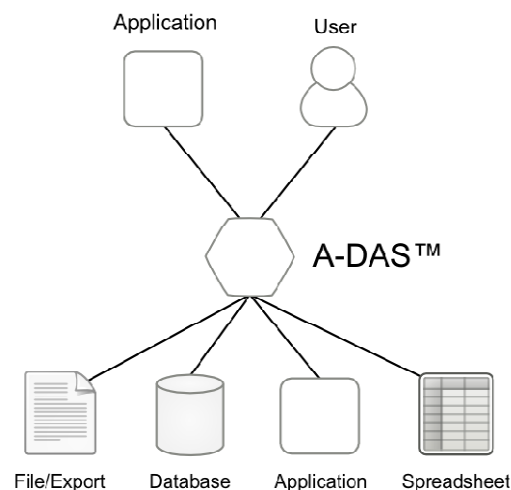


Figure 1 Data as a Service - reduce complexity and enable data in the Enterprise

Data is a fundamental component of software systems and the business processes that rely on them. Databases, documents, log files, reports and exports play their part in every Office and Back-Office application and system. Yet accessing this information is often not straightforward. The technical expertise to work with Relational Databases and Legacy Systems may not always be available and can easily become a

bottleneck. The structure and format of the data can require *cleansing, sorting and reformatting* using labour intensive processes that are often ad-hoc and difficult to reproduce, in a systematic and reliable manner.

By promoting Data as a Service A-DAS™ acts as an *enabler*; it removes the barriers for use and re-use, eliminating complex manual processes and opens up data access to the individuals and applications that need it.

3.2 Unified Data access through a Uniform Interface

A-DAS™ Services connect to underlying Data Sources and provide access to the data through a *simple, uniform HTTP Service interface*, unique to A-DAS™. The interface is designed to take advantage of the key unifying aspects of the Web: the power and ubiquity of the URL and the flexibility of Media Type negotiation built into the HTTP protocol.

To A-DAS™ every Resource, Data Model and Query is identified and accessed through a unique URL that can be reused, shared and embedded across countless applications. Consumers of A-DAS™ Services can include in their requests a preferred data format that A-DAS™ will honour. If the calling application asks for data served as XML then that is what A-DAS™ serves, if an Excel spreadsheet is more appropriate then A-DAS™ responds with a spreadsheet. The same URL can be used to present Office Word Documents to End Users, serve HTML to Web Browsers and custom XML and binary formats to applications. *A-DAS™ removes the complexity of accessing data by meeting the expectations of the Data Consumer.*

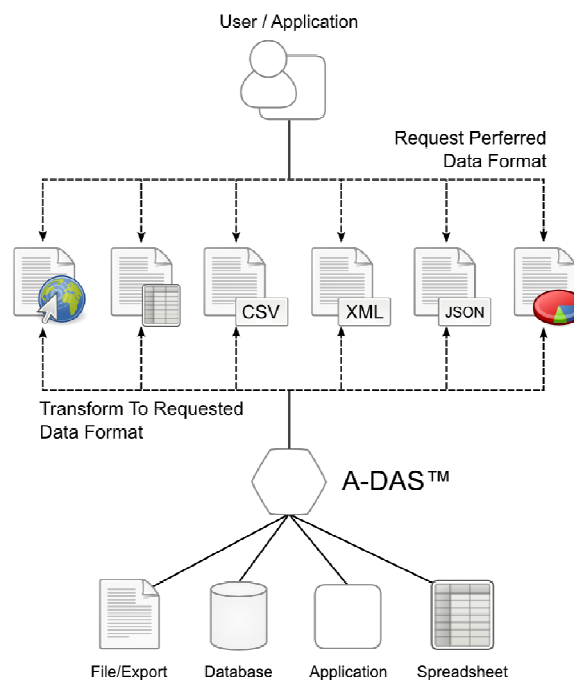


Figure 2 Access data in multiple formats from the same Uniform interface

Unlike traditional SOAP Web Services, A-DAS™ Services leverage the principles of the Web, the ubiquity of the URL. This enables them to interact with the broadest range of clients possible, opening up simple data access to non-technical End Users using Office applications and yet with the ability to scale to very large volume request from multiple complex systems and applications, requiring custom view of the data and custom formats.

3.3 Data Agility and Customised Views

[TO BE COMPLETED]

“There is no Global Model. Rapid change is unavoidable and a System that cannot adapt is just another part of the integration problem.”

4 Accessing A-DAS™ in a Browser

4.1 A-DAS™ Home Page

A-DAS™ services provide a standard point of entry for Service discovery. Viewed in a Web Browser this is the service’s *Home Page*. The Home Page provides some configuration metadata but it essentially a starting point for following links and exploring the service capabilities. Note that by convention the Service’s Home Page is always at the *root* URL, for example the Anaeko sandbox service Home Page is: <http://sandbox.anaeko.com:7007/>.

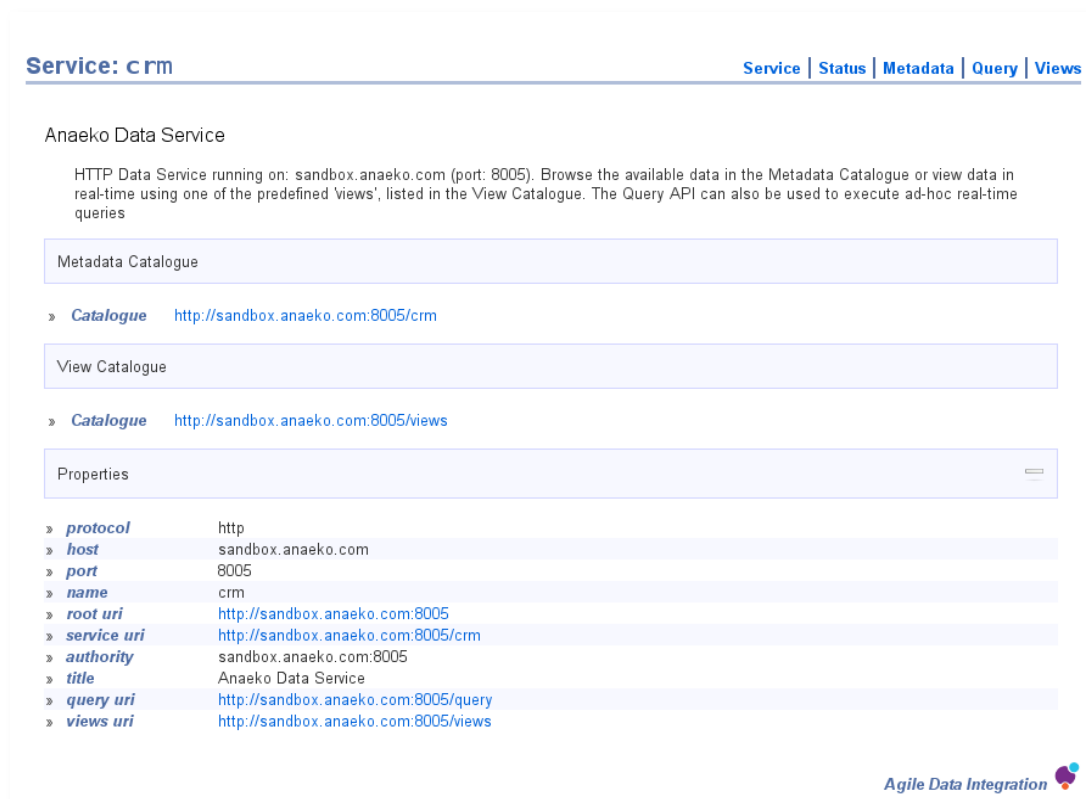


Figure 3 A-DAS Home Page, also known as the *Service Context* ([view on-line](#))

As the primary role of the A-DAS™ Home Page is Service discovery it must at a minimum provide the URLs for accessing the following key Service Resources:

- The available Data Model, through the Service’s *Metadata Catalogue*
- The available Data Views, stored in the Service’s *View Catalogue*
- The *Service Status* and system reports
- The address to send ad-hoc *Queries*.

All A-DAS™ services follow the same convention for these key service URLs.

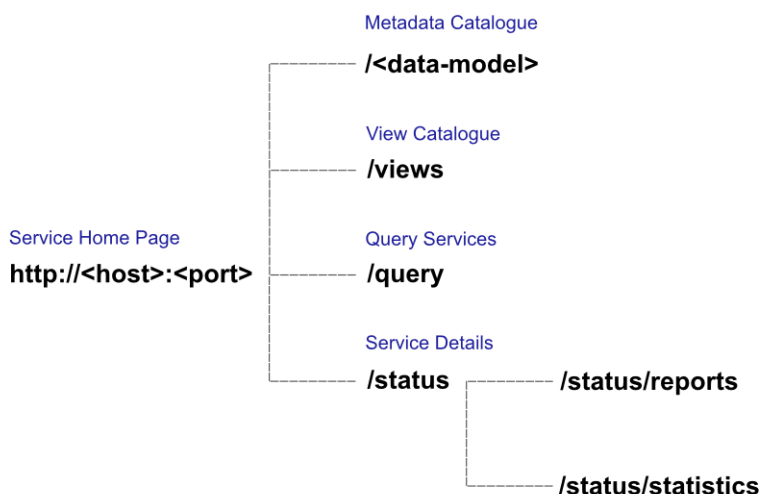


Figure 4 The default A-DAS™ URL hierachy is a strict convention

From the example Home Page in Figure 3 it is possible to navigate the sandbox service with no prior knowledge of its capabilities. The available Data Model can be found at <http://sandbox.anaeko.com:8005/crm>, the catalogue of Data Views can be found at <http://sandbox.anaeko.com:8005/views>. Details statistics and reports on the state of the service are available at <http://sandbox.anaeko.com:8005/status>. For more complex, ad-hoc service interactions the address of the Service’s Query URL is: <http://sandbox.anaeko.com:8005/query>.

For convenience the Web Browser interface to A-DAS™ includes shortcuts to these resources at the top right hand corner of most pages.



Figure 5 The key A-DAS™ resource can be accessed from the shortcut menu on most pages

4.2 How do I find out what Data Services are available?

A-DAS™ is a peer-to-peer Data Federation service. It provides stand-alone service access to underlying Data Sources over a simple RESTful HTTP interface but it also provides automatic *cooperative querying across a loosely coupled federation* of A-DAS™ services. A-DAS™ services grouped together in a peer-to-peer network of services are known as *Nodes in a Data Federation*. To simplify the management of a Data Federation one of the Nodes is designated as the Federation’s *Hub Service*. By convention the Hub Service is usually available on port 7007, as is the case for the sandbox Hub Service at <http://sandbox.anaeko.com:7007>.

To explore the available Data Services the best place to start is the Hub Service’s *Home Page* (see section 4.1). Although it is not mandatory, by default the Hub Service will be hosted on a server that has less restrictive access control. This enables the Hub Service to act as a gateway to the data, where direct access to data is not possible or simply not desirable.

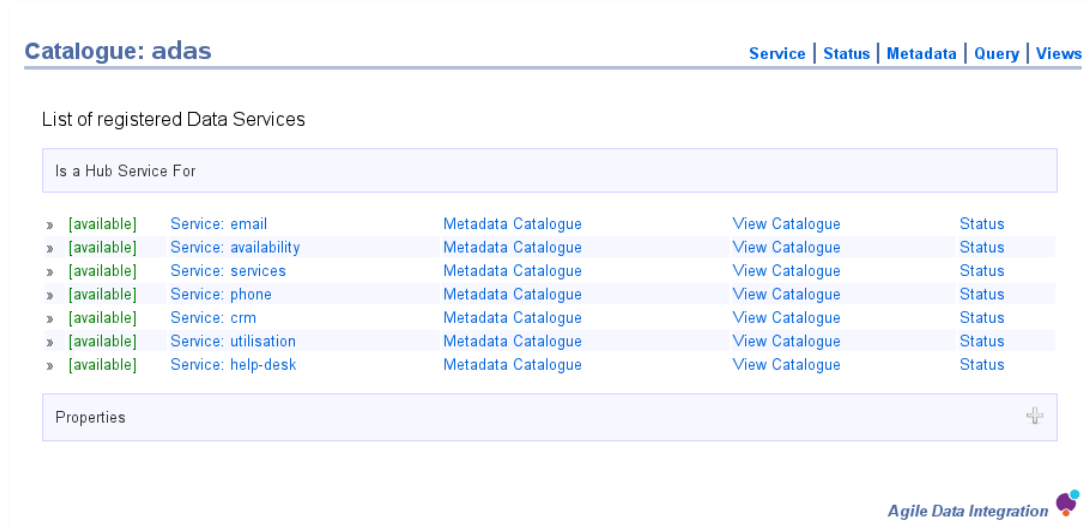


Figure 6 The Hub Service's Catalogue, linking to the available A-DAS™ Services ([view on-line](#))

Like other A-DAS™ Services there will be a link to the Hub’s Metadata Catalogue on the Hub Service’s Home Page. Unlike other A-DAS™ service, which provide service access to underlying Data Sources, the Hub Service’s Data Sources are the other A-DAS™ Nodes in the federation. Follow the Metadata Catalogue link to view the available A-DAS™ services, as illustrated in Figure 6.

The status of the available A-DAS™ services is clearly indicated as either **[available]** or **[offline]** and a link to the Service’s Home Page is provided. It is also possible to jump directly to the Metadata Catalogue of the Service, the Catalogue of Data Views and the Service’s Status Reports.

4.3 How can I explore the data provided by a Service?

Starting from either the Hub Service catalogue (Figure 6) or the Service Home Page (Figure 3) select the link for the *Metadata Catalogue*. This is the starting point for discovering the Data Model that an A-DAS™ service provides.

Each individual Service in a federation is typically serving a specific logical group of data. By convention the logical name of this group is the name of the A-DAS™ service *and* the name of the Metadata Catalogue. From Figure 6 we can see that one of the services in the sandbox is providing access to “email” data, following the naming and URL convention used by A-DAS™ we can infer that the Metadata Catalogue for this Service is: <http://sandbox.anaeko.com:8001/email>

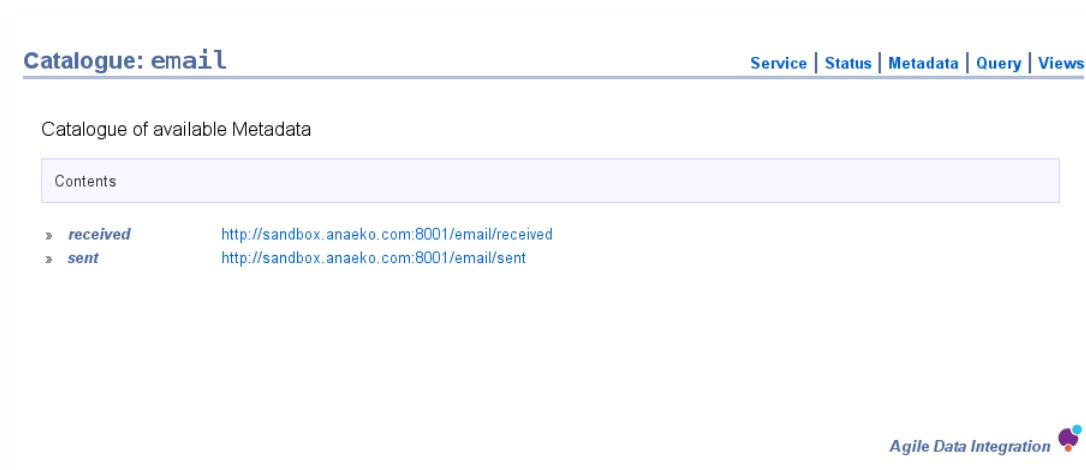


Figure 7 Use the A-DAS™ Metadata Catalogue to discover what data is available ([view on-line](#))

Figure 7 shows the Metadata Catalogue of the sandbox “email” Service, which happens to provide access to a helpdesks email logs. From the screenshot we see that there are two options available; one for data relating to outgoing mail (<http://sandbox.anaeko.com:8001/email/sent>) and another for data relating to incoming mail (<http://sandbox.anaeko.com:8001/email/received>). By following these links we can drill down into the structure of the data sets and discover details such as:

- What fields make up an email log?
- What types of data do the logs contain, Dates and Strings?
- What data can be queried?
- What size is the data set?
- What parameters can be used to search/filter the data?

If we follow the [received](#) email link we can see an example of the details structural and descriptive information available, Figure 8.

The level to which it is possible to drill down into the Data Model depends on the nature and structure of the underlying Data Source. For example a relational database can be explored from the level of the Schema down to the properties and metadata of individual Columns in a Table.

Metadata: received Service | Status | Metadata | Query | Views

Belongs to Catalogue

- » **email** <http://sandbox.anaeko.com:8001/email>

Is Made of

- » **From** <http://sandbox.anaeko.com:8001/email/received/From>
- » **Id** <http://sandbox.anaeko.com:8001/email/received/Id>
- » **Timestamp** <http://sandbox.anaeko.com:8001/email/received/Timestamp>
- » **To** <http://sandbox.anaeko.com:8001/email/received/To>

Can be Filtered using

- » **From** <http://sandbox.anaeko.com:8001/email/received/From>
- » **Id** <http://sandbox.anaeko.com:8001/email/received/Id>
- » **Timestamp** <http://sandbox.anaeko.com:8001/email/received/Timestamp>
- » **To** <http://sandbox.anaeko.com:8001/email/received/To>

Properties

- » **footer**
- » **excel.sheet** 0
- » **title**
- » **qname** received
- » **excel.source** data/excel/email-received.xls
- » **name** received
- » **Last-Modified** 2010-02-26 11:46:42
- » **cardinality** 22299
- » **queryable** true




Figure 8 A-DAS™ provides self-describing metadata ([view on-line](#))

4.4 How can I access the data?

The easiest, and recommend, way to access the data provided by A-DAS™ Services is to use pre-defined Data Views but it is also possible to perform more advanced *ad-hoc querying* of data using the Query interface.

Accessing A-DAS™ services using ad-hoc queries is a powerful but advanced capability, requiring some experience with constructing SQL-like queries. A more typical User experience of A-DAS™ centres on the access, sharing and re-use of a number of pre-defined Data Views. A-DAS™ enables advanced, and authorised, users to create and test Queries and then store these as *customised views of the data*, enabling simplified, direct and repeated access. This is analogous to a Relational Database View - although accessing an A-DAS™ View is considerably easier.

Data can be queried from any Service, with the appropriate permission, and can be sourced from multiple cooperating Services in a single query. Entries in the Metadata Catalogue of A-DAS™ Services which are marked as “*queryable*” (Figure 8) can be requested as part of an ad-hoc query. Similarly any entry in the catalogue that is marked as a “*Filter For*” another entry can be used to search and filter data. Data can

be combined and merged from single data services or across multiple A-DAS™ services and Data Sources.

4.5 Can I see what Data Views are available?

To browse the Views that have already been created either select the Views link in the top right-hand corner of the Browser or go directly to the View Catalogue of any A-DAS™ Service by requesting the Service's [/views](#) URL.



Figure 9 Browse the available Data Views in the View Catalogue ([view on-line](#))

The View Catalogue of a Service lists the Views that have been defined and stored on that particular Service instance. It is worth noting that a Service's Views are not restricted to accessing data sourced from the Service itself as Views are capable of spanning, combining and merging any permitted data set in the entire A-DAS™ federation. By convention Views that target a single data set are stored with the Service that provides access to that data, Views that span multiple data sets are stored on the Hub Service. It is also common to control access to data by restricting unprivileged user access to the Hub Service, in this type of deployment publicly accessible data would be stored as Views on the Hub Service, with tighter access controls on the individual A-DAS™ Services.

The View Catalogue Page provides links to execute the Views with a single click but it also provides links to the *definition* of each of the Views. The View Definition Page provides detailed information on the options that are available for executing the view, including the *optional and mandatory parameters* that may have been defined. The View Definition Page provides a simple way of building up a custom URL for a given set of parameters.

View: site-status
Service | Status | Metadata | Query | Views

combined site utilisation and availability

Display customer site utilisation and availability figures for the current year to date. Export to Excel to see problem sites highlighted

View Results: <http://sandbox.anaeko.com:7007/views/site-status>

Export Excel: <http://sandbox.anaeko.com:7007/views/site-status.xls>

Export CSV: <http://sandbox.anaeko.com:7007/views/site-status.csv>

Parameters

Name	Value	Type	Reset
site	<input type="text"/>	java.lang.String	

Definition +

Belongs to Catalogue

» **views** <http://sandbox.anaeko.com:7007/views>

Properties +

Agile Data Integration

Figure 10 A sample View Definition, illustrating the use of mandatory parameters ([view on-line](#))

Each of the available sections in the View Definition Page can be expanded and collapsed to view and hide details are required. The *Definition* section, hidden in Figure 10, contains a complete copy of the Query that drives the View. The *Properties* section, also hidden, contains details and links to additional metadata about the view, including a link to any *Sample data* that may be available. The sample data helps with exploring and understanding the types and structure of the data set returned by the View – follow the [sample.data](#) link provided in the properties section of the View Definition Page.

4.6 How do I use a Data View?

When an ad-hoc Query is promoted to a Data View it is assigned a *permanent and unique URL* that can be accessed by simply following a link. Users can make on-demand customised queries of the complete A-DAS™ catalogue by simply selecting, sharing or embedding a URL.

Figure 10 shows an example of the View Definition Page of a Data View which queries the *utilisation* and *availability* of customer “sites”. The unique view URL is displayed at the top of the page: <http://sandbox.anaeko.com:7007/views/site-status>. The URL of this View tells us that it is a Hub Service View. Referring back to Figure 6 we see that the *utilisation* and *availability* data sets are served by separate A-DAS™ services, therefore this View is querying and consolidating data across two separate Data Sources.

View Results: <http://sandbox.anaeko.com:7007/views/site-status>
Export Excel: <http://sandbox.anaeko.com:7007/views/site-status.xls>
Export CSV: <http://sandbox.anaeko.com:7007/views/site-status.csv>

Figure 11 All Data Views have a unique URL for direct access to the data

Below the option of viewing the results in the browser are two complementary URLs for general purpose export. By selecting one of the alternative links the results returned by the View can be exported in both Excel and CSV formats. These options are the most commonly used export formats supported by A-DAS™ but they are not the only ones. For further information on A-DAS™ support for MIMEs see Sections 4.8 Can I get the data in different formats? and 5.2.2 Requesting different Media Types.

On closer inspection of Figure 10 we can see that this particular Data View has been *parameterised* with a mandatory parameter called **site**. By entering one or more values here it is possible to build up a parameterised URL that can be executed directly, exported or copied for sharing or use in another application. If we continue the example in Figure 10 and enter the values “*Site-0001,Site-0031*” as the **site** parameter we will see the unique view URL automatically updates to reflect our change.

View Results: <http://sandbox.anaeko.com:7007/views/site-status?site=Site-0001,Site0032>
Export Excel: <http://sandbox.anaeko.com:7007/views/site-status.xls?site=Site-0001,Site0032>
Export CSV: <http://sandbox.anaeko.com:7007/views/site-status.csv?site=Site-0001,Site0032>

Parameters

Name	Value	Type	Reset
site	<input type="text" value="Site-0001,Site0032"/>	java.lang.String	

Figure 12 Parameterised Views will automatically update their URLs

Selecting the new URL, <http://sandbox.anaeko.com:7007/views/site-status?site=Site-0001,Site-0032>, will take us to the Results Page, shown in Figure 13. On the Results Page the results of a query or View are automatically paged for conveniently viewing large data sets. The data can be sorted by selecting the table headers and filtered by entering text in the Search box provided.

Response [Service](#) | [Status](#) | [Query](#) | [Views](#)

Showing 1 to 14 of 14 entries

Site	average utilisation	peak utilisation	Availability	Month
Site-0001	0.05199	0.18595	99.98	10-09-01
Site-0001	0.0659	0.97422	100	10-08-01
Site-0001	0.11197	0.78736	99.99	10-07-01
Site-0001	0.10882	1.01272	100	10-06-01
Site-0001	0.20092	0.38799	100	10-05-01
Site-0001	0.13639	0.29792	100	10-04-01
Site-0001	0.07237	0.38017	100	10-03-01
Site-0001	0.45726	0.54051	100	10-02-01
Site-0001	0.11432	0.54051	100	10-01-01
Site-0032	0.05205	0.98698	99.94	10-09-01
Site-0032	0.10085	0.78736	99.99	10-08-01
Site-0032	0.1369	0.82918	99.96	10-07-01
Site-0032	0.11197	0.0656	100	10-06-01
Site-0032	0.07156	0.73554	100	10-05-01

Show entries ◀ ▶ Search:


Agile Data Integration 

Figure 13 The Results Page shows the output of both Queries and Views ([view on-line](#))

If we had selected either the Excel or CSV options, depending on the Browser’s settings, we would have had the option to view the output directly in Excel or save the results locally for later use.

4.7 How do I make ad-hoc Queries?

A link to the Query Form, shown in Figure 14, is available on the Home Page of A-DAS™ services (Figure 3), or at the top right-hand corner of most A-DAS™ pages. Following the Query link in a Web Browser will cause the Query Form to be displayed. This form enables ad-hoc queries to be executed and results returned in various sample formats.

By convention it is possible to access the Query Form of any A-DAS™ service directly by requesting the service’s [/query](#) URL.

A-DAS™ supports a proprietary XML query format that closely follows the familiar SQL conventions of SELECT, WHERE and JOIN. The Query Form presents a template Query as a convenient starting point from which more complex queries can be built. The example shown in Figure 14 gives a flavour of the Query syntax but Ref1: *A-DAS™ Query Language Specification* is recommended for a detailed introduction to the A-DAS™ query language.



Figure 14 The A-DAS™ Query Form, for submitting ad-hoc Queries on demand

Continuing the example of Email logs outlined in section 4.3 we can construct a sample query that illustrates how an ad-hoc request can be made to return all email logs sent in the last week. Copy and paste the following sample query into the Query Form to test the example:

```
<query callback="none" limit="10">
  <select>
    <target value="http://sandbox.anaeko.com:8001/email/sent" metadata="true" />
  </select>
  <where>
    <filter target="http://sandbox.anaeko.com:8001/email/sent/Timestamp"
           action="&gt;="
           against="now(-1w)" />
    <filter target="http://sandbox.anaeko.com:8001/email/sent/Timestamp"
           action="&lt;="
           against="now()" />
  </where>
</query>
```

Figure 15 A sample Query to try out on the Anaeko sandbox

We can select a number of format options for testing this query but for the purposes of this example we will *accept* the default and select the *Submit Query* button to see the results as illustrated in Figure 16.

Query
Service | Status | Query | Views

Query Editor

```

<query callback="none" limit="10" >
  <select>
    <target value="http://sandbox.anaeko.com:8001/email/sent" metadata="true" />
  </select>
  <where>
    <filter target="http://sandbox.anaeko.com:8001/email/sent/Timestamp" action="&gt;=" against="now(-1w)" />
    <filter target="http://sandbox.anaeko.com:8001/email/sent/Timestamp" action="&lt;=" against="now()" />
  </where>
</query>

```

Accept: XHTML Raw XML Submit Query

Response

Id	Timestamp	To	From
203:276:1F0	2010-02-24 12:35:29	Byron.Reese@Ablaze-Info-Tech.Com	Raul.Barnes@Service-Desk.Com
167:39B:77	2010-02-25 17:53:55	Byron.Reese@Ablaze-Info-Tech.Com	Carson.Moore@Help-Desk.Com
1Cb:37E:148	2010-02-23 16:48:18	Byron.Reese@Ablaze-Info-Tech.Com	Roland.Holmes@Service-Desk.Com
1D5:2D7:148	2010-02-26 12:12:48	Byron.Reese@Ablaze-Info-Tech.Com	Anthony.Burnell@Help-Desk.Com
2B9:9C:5A	2010-02-25 12:00:18	Don.Childs@Allied-Biscuit.Com	Chris.Clarke@Service-Desk.Com
70:3D5:362	2010-02-23 15:51:21	Jeffrey.White@Aventies-It-Solutions.Com	Britney.Mclaughlin@Help-Desk.Com
3E:2E7:157	2010-02-21 16:45:18	Jeffrey.White@Aventies-It-Solutions.Com	Colin.Owen@Service-Desk.Com
1A3:1D1:3B1	2010-02-21 15:13:30	Jeffrey.White@Aventies-It-Solutions.Com	Armani.Mccullough@Help-Desk.Com
Ce:376:2D0	2010-02-22 10:15:25	Jeffrey.White@Aventies-It-Solutions.Com	Alison.Hayden@Service-Desk.Com
1F0:309:1F1	2010-02-25 13:10:25	Clara.Gooda@Axis-Chemical-Co.Com	Reagan.Wallace@Service-Desk.Com

Agile Data Integration

Figure 16 An example of an ad-hoc A-DAS query, showing logs of mail sent in the last week ([view on-line](#))

4.8 Can I get the data in different formats?

There can be little doubt that the de-facto standard, in recent years, for data exchange across heterogeneous systems is XML. In this respect A-DAS™ is a typical data service, favouring a widely supported interoperable XML data format by default. However, A-DAS™ provides much more than just XML – a prime example being the HTML support that has been used in previous sections; where A-DAS™ has been serving data direct to the Web Browser, for rendering as an HTML page.

A-DAS™ is an HTTP service with full support for the media negotiation parts of the [HTTP standard](#) and a *plug-in MIME handling system* that supports key interchange and Office formats as standard. A-DAS™ also includes drop in support for proprietary XML formats, through XSL transformations.

Continuing the Data View example of Section 4.5 we can make a direct call to A-DAS™ (not using a Web Browser) and see that the response is returned in the default XML format, described in detail in *Appendix B: XML Formats*.

```

<response status="SUCCESS_OK"
  id="http://sandbox.anaeko.com:7007/query/293"
  success="true">

<data xml.parser="com.anaeko.utils.data.xml.ArrayTableReader">
  <table footer="" title="" description="" id="table9">
    <thead>
      <th java.class="java.lang.String" id="table44/Site">Site</th>
      <th java.class="java.lang.Double"
        id="table44/average-utilisation"
        name="average-utilisation">average utilisation</th>
      <th java.class="java.lang.Double"
        id="table44/peak-utilisation"
        name="peak-utilisation">peak utilisation</th>
      <th java.class="java.lang.Double"
        id="table44/Availability">Availability</th>
      <th java.class="java.util.Date" id="table44/Month">Month</th>
    </thead>
    <tbody>
      <tr id="row0">
        <td>Site-0001</td>
        <td>0.05199</td>
        <td>0.18595</td>
        <td>99.98</td>
        <td>10-09-01</td>
      </tr>
      ...

```

Figure 17 By default A-DAS™ returns XML ([view on-line](#))

Alternatively we can remain in the browser and request that raw XML is sent, rather than the HTML rendering, by specifying the MIME type `text/xml` in the Request: <http://sandbox.anaeko.com:7007/views/site-summary>

A key strength of the A-DAS™ Interface is that the same URL can be used to return the data in range of formats tailored for the requesting application. In previous examples the same URL requested by a Web Browser, capable of rendering HTML, will result in the response was displayed as an HTML document in a fully searchable, paged table, as illustrated in Figure 13. Similarly if asked for an alternative format such as Excel, CSV, JSON, etc A-DAS™ will return the result as requested.

Figure 18 illustrates the same Data View response in JSON format.

```

adasResponse({
  "response": {
    "transformerType": "com.anaeko.service.response.Response",
    "url": "http://sandbox.anaeko.com:7007/query/296",
    "code": "SUCCESS_OK",
    "data": {
      "transformerType": "com.anaeko.utils.data.Tabular",
      "properties": {
        "footer": "",
        "title": "",
        "description": ""
      },
      "columns": [ {
        "name": "Site",
        "type": "java.lang.String"

```

```

    }, {
      "name" : "average-utilisation",
      "type" : "java.lang.Double"
    }, {
      "name" : "peak-utilisation",
      "type" : "java.lang.Double"
    }, {
      "name" : "Availability",
      "type" : "java.lang.Double"
    }, {
      "name" : "Month",
      "type" : "java.util.Date"
    } ],
    "data" : [ [ "Site-0001", "0.05199", "0.18595", "99.98", "" ],
               [ "Site-0032", "0.05205", "0.98698", ... ] ],
               ...
    ]
  }
}
})

```

Figure 18 Example JSON response ([view on-line](#))

Shortcuts for Excel and CSV are provided in the View Definition Page, Figure 12, but these are simply convenient links to these common formats. As a Data Service A-DAS™ is capable of returning any of the formats that have been installed as plug-ins to its MIME Handling System.

A-DAS™ supports *three* separate mechanisms for requesting a specific format. This offers the calling application a range of options to simplify working with A-DAS™ Services. The default mechanism is the HTTP standard for media negotiation, the **Accept** header.

1. HTTP Accept

The HTTP protocol supports the use of an Accept header in client requests. This header should list in order of preference, or by indication using a **q** value, the media types supported by the client. A-DAS™ will attempt to return the preferred format but will also inspect the alternative options in order of preference, finally returning the default XML if none of the requested MIMEs are supported.

Media Type	Example HTTP Request
Default XML	<pre>GET /views/list-customer-sites HTTP/1.1 Host: sandbox.anaeko.com:7007 Accept: application/xml; charset=utf-8</pre>
MS Excel	<pre>GET /views/list-customer-sites HTTP/1.1 Host: sandbox.anaeko.com:7007 Accept: application/vnd.ms-excel</pre>

CSV	<pre>GET /views/list-customer-sites HTTP/1.1 Host: sandbox.anaeko.com:7007 Accept: text/csv; charset=UTF-8</pre>
JSON	<pre>GET /views/list-customer-sites HTTP/1.1 Host: sandbox.anaeko.com:7007 Accept: application/json; charset=UTF-8</pre>
Crystal Reports XML	<pre>GET /views/list-customer-sites HTTP/1.1 Host: sandbox.anaeko.com:7007 Accept: application/crystal+xml</pre>

Figure 19 Example media types with specific HTTP Accept Requests

2. Common File extensions

Some key formats have well established file naming conventions, particularly on Microsoft Windows systems. Where there is a clear standard, such as ***.xls** for Microsoft Excel files, A-DAS™ will support this extension directly in the request URL.

Media Type	Example URL
Default XML	http://sandbox.anaeko.com:7007/views/list-customer-sites
MS Excel	http://sandbox.anaeko.com:7007/views/list-customer-sites.xls
CSV	http://sandbox.anaeko.com:7007/views/list-customer-sites.csv
JSON	http://sandbox.anaeko.com:7007/views/list-customer-sites.json
Crystal Reports XML	http://sandbox.anaeko.com:7007/views/list-customer-sites.report

Figure 20 Example Media types with in-line URL support

3. http-accept query parameter

Not all HTTP capable clients provide full support for all of the HTTP Headers. Where it is not possible or undesirable to manipulate the HTTP request directly A-DAS™ supports the ability to override the HTTP Request Accept Header, using the query parameter: **http-accept**.

Media Type	Example URL with HTTP Accept override parameter
Default XML	http://sandbox.anaeko.com:7007/views/list-customer-sites
MS Excel	http://sandbox.anaeko.com:7007/views/list-customer-sites?http-accept= application/vnd.ms-excel
CSV	http://sandbox.anaeko.com:7007/views/list-customer-sites?http-accept= text/csv
JSON	http://sandbox.anaeko.com:7007/views/list-customer-sites?http-accept=application/json
Crystal Reports XML	http://sandbox.anaeko.com:7007/views/list-customer-sites?http-accept=application/crystal+xml

Figure 21 Example Media types with the http-accept override parameter

4.9 Do I have to use a Browser?

A-DAS™ is an HTTP data service that supports any HTTP capable client. In practice the most commonly used End User application is likely to be a Web Browser, however, because of ubiquity of the Web and Web URLs a surprising number of options are available for accessing data through A-DAS™, without resorting to programming.

In addition to the general utility and accessibility of HTTP URLs the built in support for HTTP media type negotiation in A-DAS™, as outlined in Section 4.8, enables direct integration with many common Office tools, including Microsoft Word and Excel.

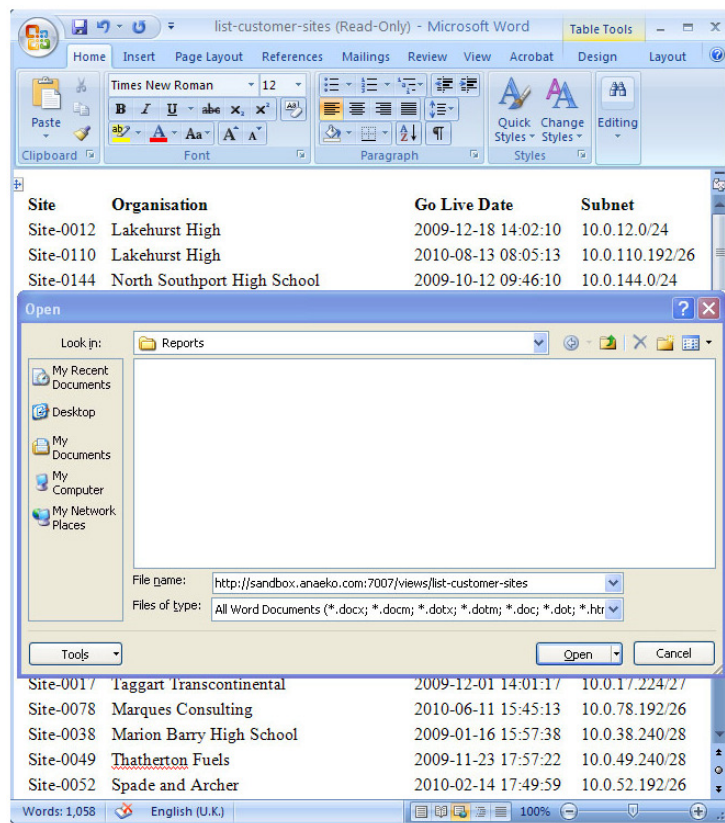


Figure 22 Access A-DAS™ data directly from within Microsoft Word

Open any A-DAS™ View directly in MS Word and the results of the query will be displayed in a Word Document. Similarly, open the same URL in Excel and the results will be in spreadsheet form, ready for processing into pivot tables or Excel graphs. By specifying CSV as the desired format most modern text editors will be capable of opening A-DAS™ View URLs directly. The option of CSV is also the most commonly supported import/export format for loading data into many Enterprise applications, including Relational Databases. By providing data in the most appropriate format, from a single unique URL and A-DAS™ view can simplify sharing of data between individuals and applications, potentially reducing a multi-step database ETL to a single HTTP request.

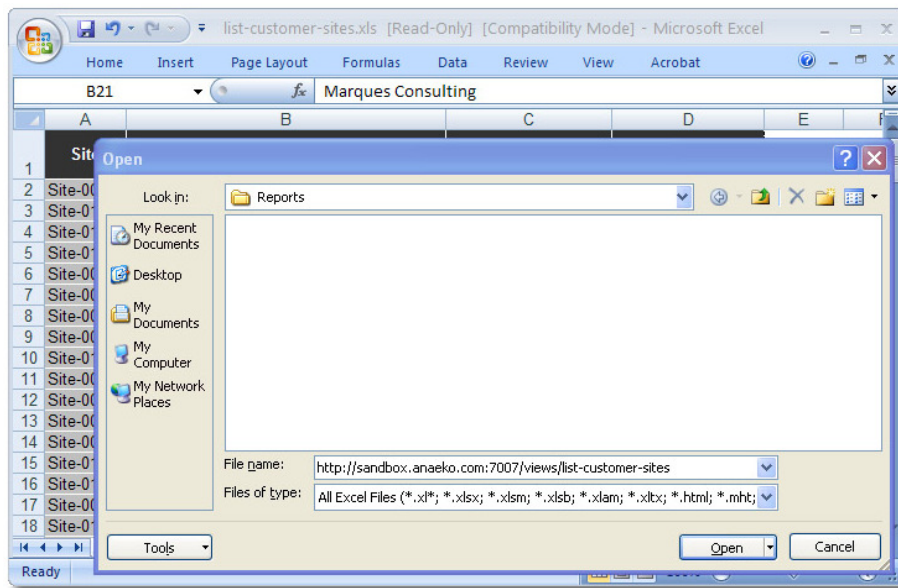


Figure 23 Access A-DAS™ data without leaving Microsoft Excel

4.10 What is the status of the service?

As illustrated in Figure 6 the Hub Service can provide a simple available/offline status indicator for the configured A-DAS™ Services. However, a much more detailed description of the state of the Service is available on the individual Service Status Pages. In general this can be found by following the Status link at the top right of the Browser but it is possible to access the status directly by simply requesting the Service's [/status](#) URL.

The Status Page provides a collection of metrics that are accurate at the time the request is made. These *statistics* include the number of queries currently being processed, the total number of queries processed, the service uptime and many more. Although there are many common statistics available across all A-DAS™ Services the complete list of available statistics depends on the nature of the underlying Data Source, with different statistics available for SOAP services, Relational Databases, etc.

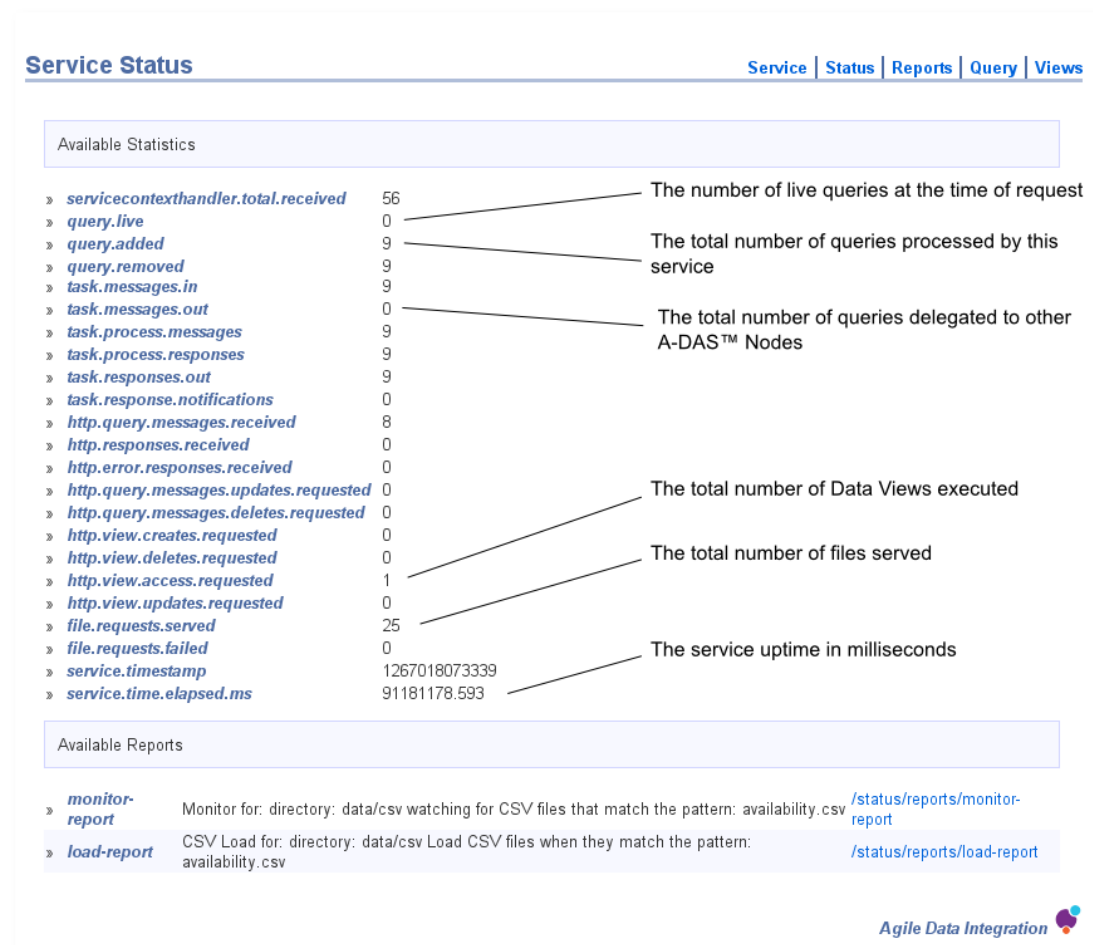


Figure 24 A-DAS™ Status Page ([view on-line](#))

In addition to the service statistics the Status Page provides links to available reports that have been collated from key internal processes. These reports may include error reports but will usually consist of detailed Data Source monitoring and load information. The example in Figure 24 shows two reports; one that provides details of a file that is being monitored for changes and another that provides detailed information about the last data load of a [CSV](#) file.

As with the statistics the full list of reports that are available depends on the nature of the underlying Data Source and the individual Service configuration. The load report example in Figure 25 provides detailed information on the outcome of loading a CSV file, called `availability.csv`, into the A-DAS™ service. The report includes relevant internal log messages. From the report properties it is possible to determine when the load occurred, how many files were loaded, the total number of rows of data loaded and the number of rows skipped due to data quality issues. In addition to this we can see from the log messages in Figure 25 that the CSV file was scanned and determined to contain UTF-8 encoded text, possibly indicating that it did not originate from a Windows PC.

Report: Load- report Service | Status | Reports | Query | Views

CSV Load for: directory: data/csv Load CSV files when they match the pattern: availability.csv

Properties

- » **Previous File List** Empty
- » **service.name** availability
- » **Charset** UTF-8
- » **Skipped Row Count** 0
- » **File Count** 1
- » **Previous File Count** 0
- » **File List** availability.csv
- » **Previously Loaded** Never
- » **Load Timestamp** 2010-02-26 11:46:33
- » **Row Count** 1960

Messages

Category	Title	Message	Source	Time Stamp
INFO	Load CSV	Loading CSV data from file: /home/roger/workspace/kpi-demo/data/csv/availability.csv at: 2010-02-26 11:46:33	com.anaeko.error.UserReport	2010-02-26 11:46:33
INFO	Reading CSV File	Reading CSV file: /home/roger/workspace/kpi-demo/data/csv/availability.csv start:0 end:EOF	com.anaeko.utils.csv.CSVLoadReport	2010-02-26 11:46:33
INFO	Character Encoding	File: /home/roger/workspace/kpi-demo/data/csv/availability.csv appears to be encoded as: UTF-8	com.anaeko.utils.csv.CSVLoadReport	2010-02-26 11:46:33



Figure 25 A CSV file report, with details of the file load ([view on-line](#))

5 A-DAS™ REST API

Accessing A-DAS™ using a web browser is one facet of the A-DAS™ interface, a user friendly way of navigating, browsing and discovering the Data Model and Data. However, at its core A-DAS™ is a fully compliant **HTTP/1.1 Data Service**; it provides **unified and uniform access** to heterogeneous Data Sources using the ubiquitous HTTP protocol. The A-DAS™ interface is designed to be fully RESTful, a key advantage over more typical SOAP services, or simple Web Service that serve XML (sometimes called XML/HTTP or XML-RPC services).

As illustrated in Section 4 it is the HTTP compliant RESTful design that enables A-DAS™ to serve user friendly web pages direct to the browser, or Excel files direct to Microsoft Excel, using the **same interface** as the core data service API. Each web page in Section 4 is available as a single RESTful API call to the same URL used by the Web Browser. By default A-DAS™ serves industry standard XML but it is also capable of responding to custom, application specific requests for Media Types, such as JSON, CSV, HTML and many others.

The principal design goal of the A-DAS™ interface is to serve data in the format most suitable for the client from a single HTTP request made to a reusable, multi-purpose, URL. A deliberate effort has been made to make the A-DAS™ data service API simple with the full capabilities of the service available over just a handful of well structured URLs (Appendix A: URL Catalogue).

As an example open the following URL in a browser, in Microsoft Word or Excel or even a Text Editor: <http://sandbox.anaeko.com:8002/views/availability-this-month>. The ubiquity of the URL and HTTP standards make A-DAS™ Data Service accessible in ways that are simply not possible with typical Web Services, built on SOAP. ***The A-DAS™ API meets the expectations of the client and significantly reduces the effort of consuming data.***

A-DAS™ RESTful credentials include:

- Support for client requested MIME types.
- Hypermedia representations of Resources, enabling service discovery and state transitions.
- Strict adherence to the semantics of HTTP GET, PUT, POST, DELETE and HEAD requests.
- Idempotent GET and PUT requests are guaranteed.
- Strict use of query parameters only as Resource filters on GET requests.
- Support for proxies and other intermediaries through the HTTP cache directives

The remainder of this Section details the core service API of A-DAS™ and illustrates how to work with Data Views and ad-hoc queries. The more advanced interactions with the Service, such as processing the self-describing Hypermedia and how to dynamically discover and interact with data, are also introduced.

5.1 Hubs and Nodes

In its simplest configuration A-DAS™ acts as a Data Service *enabler* for a single source of data. A-DAS™ sits between the consumer, a user or application, and the underlying Data Source providing SQL like query capabilities through a simple HTTP service.

Applications and Users can create ad-hoc queries or re-use stored and parameterised Views onto the underlying Data Source with a full range of query capabilities regardless of the source. The same HTTP service interface is used for applications and users, serving XML, Excel, HTML, JSON, CSV or a proprietary format as the requested by the consumer.

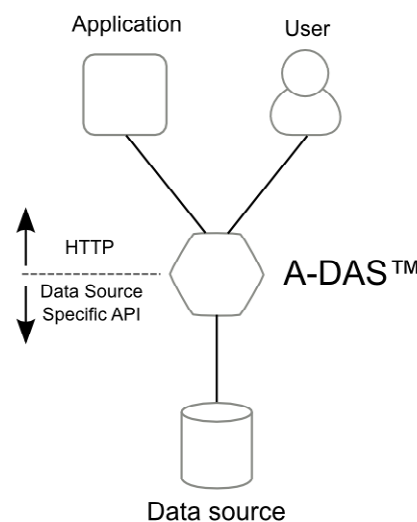


Figure 26 A-DAS™ configuration - single Data Source

The most powerful, and common, deployment configuration of A-DAS™ is where multiple instances are configured as a *cooperating federation of loosely coupled services* across a range of different Data Sources. In this type of deployment the same simple HTTP interface is available to the consumer; the difference is that multiple sources of data are seamlessly merged into a single Service that supports the same SQL like query features across multiple Data Sources.

Figure 27 illustrates how a single *Hub Service* can be added as a central point of access. This is the configuration used in the Anaeko sandbox deployment of A-DAS™, where a Hub Service is deployed at <http://sandbox.anaeko.com:7007> to provide a consolidated access point to 7 separate services, details of which can be found at <http://sandbox.anaeko.com:7007/adas> (also show as a screenshot in Figure 6).

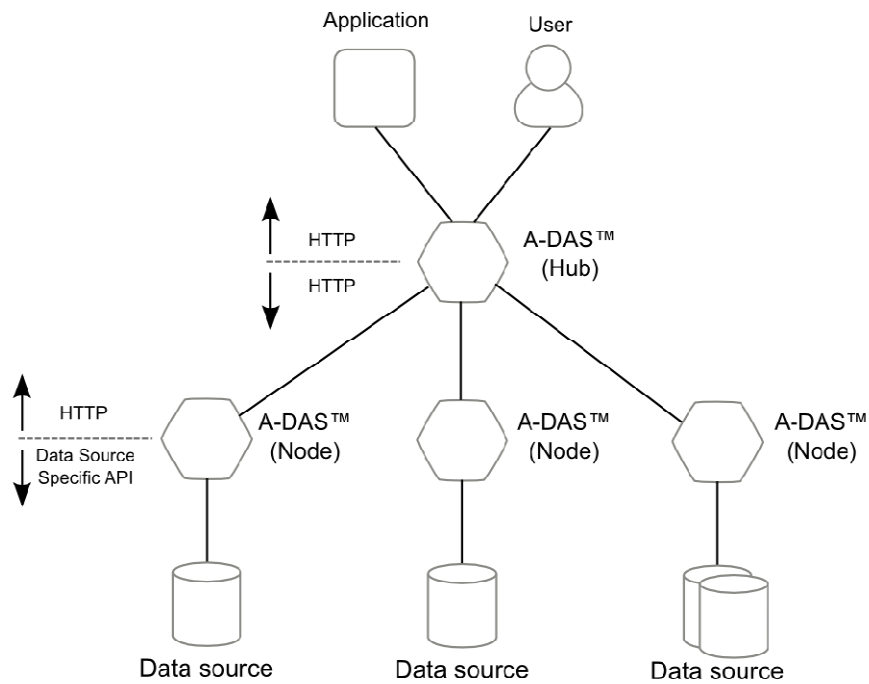


Figure 27 A-DAS™ configuration - multiple Data Sources with Hub Service

The introduction of a Hub Services does not preclude the direct access of the other Nodes in the Federation. All of the Data Services can be accessed directly, depending on the underlying network configuration and permissions.

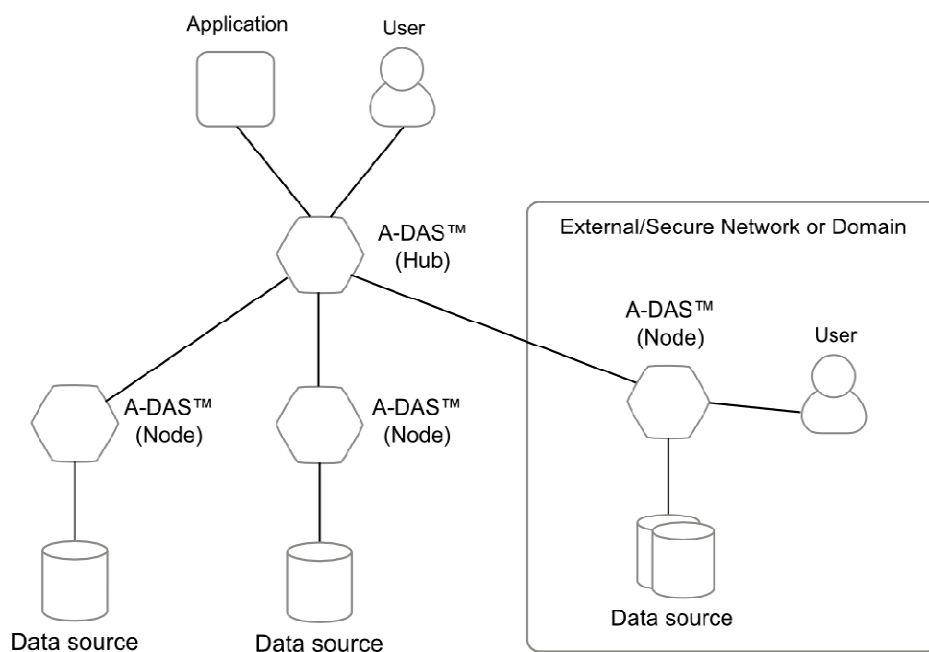


Figure 28 A-DAS™ configuration - cross domain/network Data Federation

5.2 Accessing data using a View

The primary purpose of the A-DAS™ Interface is to serve data to any HTTP capable application, which given the ubiquity of HTTP and URLs opens up a huge range of options for interacting with an A-DAS™ service. The most obvious way of accessing A-DAS™ is using a standard Web Browser but every modern programming or scripting language provides HTTP and URL support, as do most office applications.

The simplest and recommended approach to requesting data from A-DAS™ is to use a *pre-configured* Data View. A Data View is a named, stored and optimised Query that has been assigned a unique URL. To request the data, in real-time, from an A-DAS™ View the consumer simply makes an HTTP GET request to its unique URL.

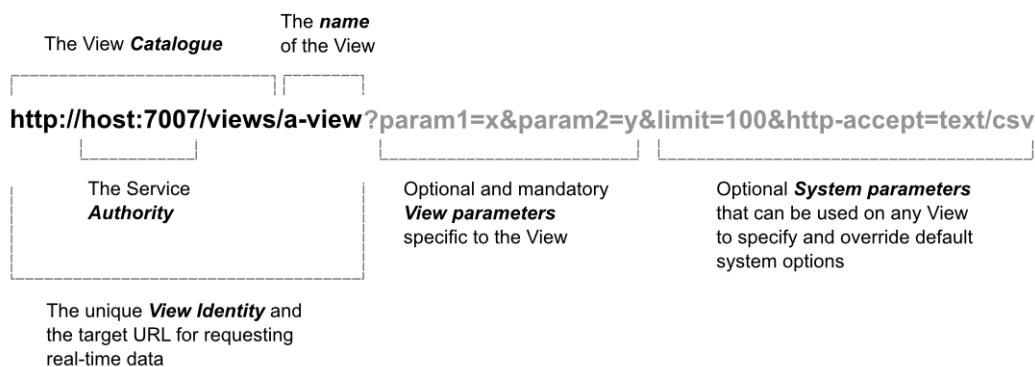


Figure 29 The anatomy of an A-DAS™ URL - Data View

The HTTP protocol enables the calling application to provide details of how it would prefer the data be formatted and returned. For this reason the specifics of the request can effect how A-DAS™ responds. The right combination of **User-Agent** and **Accept** headers should enable A-DAS™ to respond to the request with data in a format suitable for the application that made the request.

If we examine (Figure 30) one of the Web Browser examples used in Section 4 we can see that the browser identifies itself as Mozilla/5.0 and it also asks that text/html be used as the preferred response format. It also includes a number of alternative MIME options, including application/xml. On receiving the Web Browser's request to the [list-customer-sites](#) URL A-DAS™ inspects the **User-Agent** and **Accept** headers and determines that the calling application is capable of rendering XML using XSL transformations. Rather than render the data as HTML itself A-DAS™ responds with XML data and an XSL header, which enables the browser to perform the HTML rendering. The rationale behind this is that the HTML rendering is a small but significant overhead that can be offloaded to the calling application, freeing the A-DAS™ Service to focus on serving data.

Response			
		Service	Status
		Query	Views
Showing 1 to 10 of 156 entries			
Site	Organisation	Go Live Date	Subnet
Site-0000	Version Technologies	2010-09-07 16:39:30	10.0.0.0/28
Site-00	Request		
Site-00	REQUEST	GET http://sandbox.anaeko.com:7007/views/list-customer-sites HTTP/1.1	
Site-00	Host	sandbox.anaeko.com:7007	
Site-00	User-Agent	Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6	
Site-00	Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8	
Site-00	Accept-Language	en-us,en;q=0.5	
Site-00	Accept-Encoding	gzip,deflate	
Site-00	Accept-Charset	ISO-8859-1,utf-8;q=0.7,*;q=0.7	
Site-00	Keep-Alive	115	
Site-00	Connection	keep-alive	
Site-00	Response		
Site-00	RESPONSE	HTTP/1.1 200 OK	
Site-00	Content-Type	application/xml; charset=UTF-8; xml.parser=com.anaeko.service.response.xml.in.ResponseReader	
Site-00	Content-Length	37684	

Figure 30 A closer look at the Web Browser's HTTP Request

If we make the same request that the Web Browser made but we replace the **Accept** header with a different MIME, or list of MIMEs, A-DAS™ will respond with a different, appropriate format. Figure 31 shows the same request to the [list-customer-sites](#) View made using a Telnet client.

```
$telnet sandbox.anaeko.com 7007

Trying...
Connected to sandbox.anaeko.com.
Escape character is '^]'.

GET /views/list-customer-sites HTTP/1.1
host: sandbox.anaeko.com
Accept: text/xml

HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: 37623

<response status="SUCCESS_OK"
id="http://sandbox.anaeko.com:7007/query/341895" success="true">
  <data xml.parser="com.anaeko.utils.data.xml.ArrayTableReader">
    <table footer="" title="" description="" id="table1">
      <thead>
        <th column.type.precision="20" column.type.name="VARC...
        <th column.type.precision="128" column.type.name="VAR...
        <th java.class="java.util.Date" id="table2/Go-Live-Da...
        <th java.class="java.lang.String" id="table2/Subnet">...
      </thead>
      <tbody>
        <tr id="row0">
          ...
```

Figure 31 Example Request/Response to an A-DAS™ Data View

Note that all example **Request/Response** headers are highlighted and that responses have been truncated for clarity.

With a range of built in MIME types and the option to plug-in custom MIME handlers, and XSL transformations, the **Accept** header is the key mechanism used to control how A-DAS™ responds. For example, a Web application might prefer to make AJAX calls to A-DAS™ and receive JSON formatted data in response. To achieve this we can perform the same HTTP request, used in previous examples, with an **Accept** header requesting `application/json`.

```
$telnet sandbox.anaeko.com 7007

Trying...
Connected to sandbox.anaeko.com.
Escape character is '^]'.

GET /views/list-customer-sites HTTP/1.1
host: sandbox.anaeko.com
Accept: application/json

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 13293

adasResponse({
  "response" : {
    "transformerType" : "com.anaeko.service.response.Response",
    "url" : "http://sandbox.anaeko.com:7007/query/341916",
    "code" : "SUCCESS_OK",
    "data" : {
      "transformerType" : "com.anaeko.utils.data.Tabular",
      "properties" : {
        "footer" : "",
        "title" : "",
        "description" : ""
      },
      "columns" : [ {
        "name" : "Site",
        "type" : "java.lang.String"
      }, {
        "name" : "Organisation",
        "type" : "java.lang.String"
      }, {
        "name" : "Go-Live-Date",
        "type" : "java.util.Date"
      }, {
        "name" : "Subnet",
        "type" : "java.lang.String"
      } ],
      "data" : [ [ "Site-0012", "Lakehurst High",
        ...
```

Figure 32 Example Request/Response with an Accept header requesting JSON

On receiving the request to serve `application/json` A-DAS™ formats its response to meet this requirement. Similarly we might request CSV data, a common data exchange format, from the same URL, Figure 33.

```
$telnet sandbox.anaeko.com 7007

Trying...
Connected to sandbox.anaeko.com.
Escape character is '^]'.

GET /views/list-customer-sites HTTP/1.1
host: sandbox.anaeko.com
Accept: text/csv

HTTP/1.1 200 OK
Content-Type: text/csv
Content-Length: 10118

Site,Organisation,Go Live Date,Subnet
Site-0012,Lakehurst High,2009-12-18 14:02:10,10.0.12.0/24
Site-0110,Lakehurst High,2010-08-13 08:05:13,10.0.110.192/26
Site-0144,North Southport High School,2009-10-12 09:46:10,10.0.144.0/24
...
```

Figure 33 Example Request/Response with an Accept header requesting CSV

Or a custom XML format suitable for the Crystal Reports engine.

```
$telnet sandbox.anaeko.com 7007

Trying...
Connected to sandbox.anaeko.com.
Escape character is '^]'.

GET /views/list-customer-sites HTTP/1.1
host: sandbox.anaeko.com
Accept: application/crystal+xml

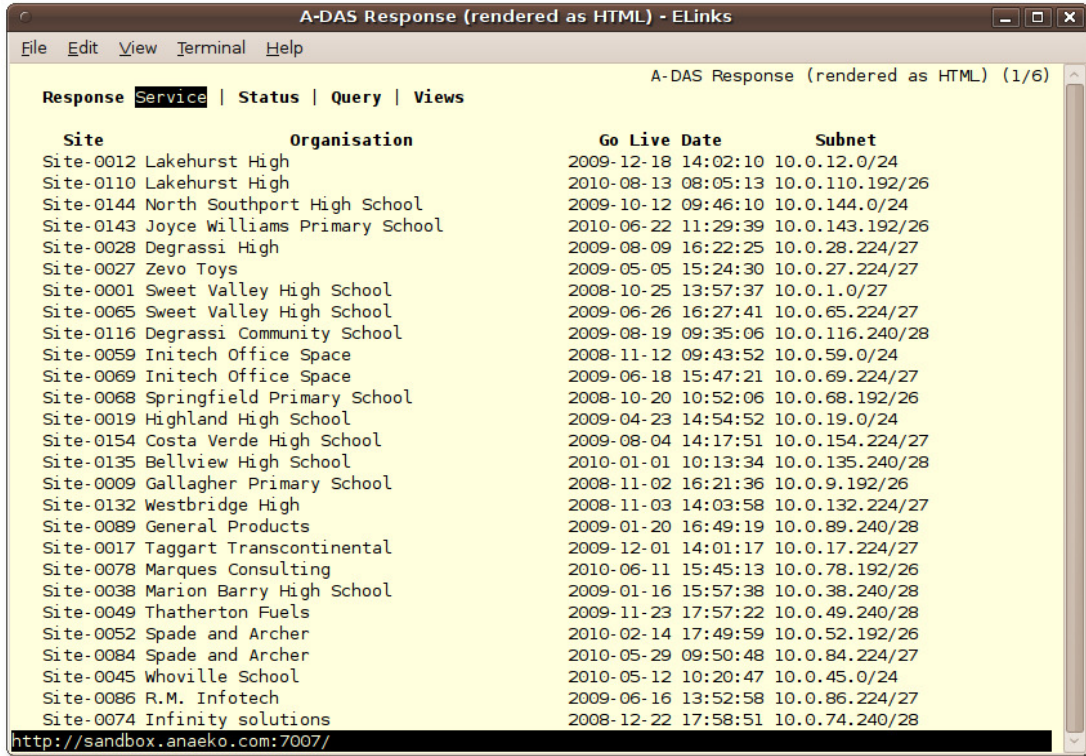
HTTP/1.1 200 OK
Content-Type: application/crystal+xml
Content-Length: 31781

<table>
  <tr>
    <Site>Site-0012</Site>
    <Organisation>Lakehurst High</Organisation>
    <Go-Live-Date>2009-12-18T14:02:10</Go-Live-Date>
    <Subnet>10.0.12.0/24</Subnet>
  </tr>
  ...

```

Figure 34 Example Request/Response with an Accept header requesting Crystal XML

As a final example, in Figure 35, we make the request using a Web Browser that does *not* support XSL transformations, to illustrate that A-DAS™ honours the request for text/html and formats the data as HTML before returning the response.



Site	Organisation	Go Live Date	Subnet
Site-0012	Lakehurst High	2009-12-18 14:02:10	10.0.12.0/24
Site-0110	Lakehurst High	2010-08-13 08:05:13	10.0.110.192/26
Site-0144	North Southport High School	2009-10-12 09:46:10	10.0.144.0/24
Site-0143	Joyce Williams Primary School	2010-06-22 11:29:39	10.0.143.192/26
Site-0028	Degrassi High	2009-08-09 16:22:25	10.0.28.224/27
Site-0027	Zevo Toys	2009-05-05 15:24:30	10.0.27.224/27
Site-0001	Sweet Valley High School	2008-10-25 13:57:37	10.0.1.0/27
Site-0065	Sweet Valley High School	2009-06-26 16:27:41	10.0.65.224/27
Site-0116	Degrassi Community School	2009-08-19 09:35:06	10.0.116.240/28
Site-0059	Initech Office Space	2008-11-12 09:43:52	10.0.59.0/24
Site-0069	Initech Office Space	2009-06-18 15:47:21	10.0.69.224/27
Site-0068	Springfield Primary School	2008-10-20 10:52:06	10.0.68.192/26
Site-0019	Highland High School	2009-04-23 14:54:52	10.0.19.0/24
Site-0154	Costa Verde High School	2009-08-04 14:17:51	10.0.154.224/27
Site-0135	Bellview High School	2010-01-01 10:13:34	10.0.135.240/28
Site-0009	Gallagher Primary School	2008-11-02 16:21:36	10.0.9.192/26
Site-0132	Westbridge High	2008-11-03 14:03:58	10.0.132.224/27
Site-0089	General Products	2009-01-20 16:49:19	10.0.89.240/28
Site-0017	Taggart Transcontinental	2009-12-01 14:01:17	10.0.17.224/27
Site-0078	Marques Consulting	2010-06-11 15:45:13	10.0.78.192/26
Site-0038	Marion Barry High School	2009-01-16 15:57:38	10.0.38.240/28
Site-0049	Thatherton Fuels	2009-11-23 17:57:22	10.0.49.240/28
Site-0052	Spade and Archer	2010-02-14 17:49:59	10.0.52.192/26
Site-0084	Spade and Archer	2010-05-29 09:50:48	10.0.84.224/27
Site-0045	Whoville School	2010-05-12 10:20:47	10.0.45.0/24
Site-0086	R.M. Infotech	2009-06-16 13:52:58	10.0.86.224/27
Site-0074	Infinity solutions	2008-12-22 17:58:51	10.0.74.240/28

Figure 35 Example of A-DAS™ data rendered as HTML for a client with limited capabilities

The “Web Browser” used in Figure 35 is a console/text based browser called [ELinks](#) and is shown as an extreme example of how data can be served in the format most suitable for the requesting application.

In most cases it is likely that the calling application will prefer and explicitly request one of the common data exchange formats illustrated in this section, such as the default XML, JSON or CSV. A-DAS™ uses a plug-in MIME Handling System that can support custom/proprietary formats with minimal effort. Text formats, such as custom XML, can be generated dynamically by specifying an appropriate XSL file as a MIME option using the standard HTTP extension mechanism. Complex binary formats are supported by implementing a single Java Interface to convert the default tabular data structure.

Internally no distinction is made between built-in MIME types, such as Excel or CSV, and custom MIME type. This enables seamless integration of proprietary formats.

5.2.1 Limiting the results returned

A-DAS™ provides two mechanisms for limiting the results that are returned from a View. An upper *limit* can be set on the number of data rows returned *or* by using the *server-side paging support* to fetch blocks of rows as required.

The limit option is specified by appending the maximum number of desired rows to the end of the request URL, using the system parameter: **limit** as illustrated in Figure 36.

```
http://sandbox.anaeko.com:7007/views/list-customer-sites?limit=10
```

Figure 36 Request View results be limited to a maximum number of rows using the limit query parameter

Server-side paging is possible by appending the range of, inclusive, rows that are desired. Note that A-DAS™ results are indexed starting at row 0.

```
http://sandbox.anaeko.com:7007/views/list-customer-sites?rows=0-9
```

Figure 37 Request View results in pages using the rows query parameter

The row option is also available as part of the standard HTTP, using the **Range** header. The typical use of the HTTP **Range** header is to specify the range of bytes that is desired, for example when streaming media. However, the standard allows for arbitrary *units* of range and A-DAS™ supports the unit: `rows`.

```
GET /views/list-customer-sites HTTP/1.1
host: sandbox.anaeko.com
Accept: text/csv
Range: rows=20-22

HTTP/1.1 200 OK
Content-Range: rows 20-22/156
Content-Type: text/csv
Content-Length: 231

Site,Organisation,Go Live Date,Subnet
Site-0038,Marion Barry High School,2009-01-16 15:57:38,10.0.38.240/28
Site-0049,Thatherton Fuels,2009-11-23 17:57:22,10.0.49.240/28
Site-0052,Spade and Archer,2010-02-14 17:49:59,10.0.52.192/26
```

Figure 38 Using the HTTP Range header for server-side paging

Using the `rows=n-m` as a query parameter is a convenient way of specifying the HTTP **Range** header, when setting it explicitly is either not possible or is inconvenient.

5.2.2 Requesting different Media Types

A-DAS™ supports three separate mechanisms for a calling application to specify a preferred MIME type.

1. HTTP Accept header

The primary and recommended mechanism, illustrated in the previous section, is to use the HTTP **Accept** header. As part of the [HTTP standards](#) this provides the highest level interoperability. However, not all HTTP enabled applications, tools or libraries support customisation of HTTP headers. To support situations where manipulating the HTTP headers is either impossible or undesirable A-DAS™ provides two additional mechanisms.

2. http-accept query parameter

When making a request to A-DAS™ the calling application can override the HTTP **Accept** header by appending the query string parameter **http-accept** to the request URL.

```
http://.../views/list-customer-sites?http-accept=text/html
```

Figure 39 Request rendered HTML using the http-accept override query parameter

In the above example A-DAS™ will ignore the HTTP Accept header sent as part of the request and return the data as rendered HTML. The **http-accept** query parameters supports a single MIME type plus MIME options in the exact same format defined for the HTTP **Accept** header.

```
http://.../views/list-customer-sites?http-accept=application/xml; charset=UTF-16
```

Figure 40 Request using the http-accept override with a character set MIME option set

3. Common file extensions

Many of the common formats that are used for data exchange have well known file extensions. Formats such as Excel are typically saved with names like *.xls or *.xlsx and a file with a name in the form of *.csv is generally accepted to contain CSV formatted text. A-DAS™ supports this convention for all standard MIMEs.

```
http://sandbox.anaeko.com:7007/views/list-customer-sites.xls  
http://sandbox.anaeko.com:7007/views/list-customer-sites.json  
http://sandbox.anaeko.com:7007/views/list-customer-sites.csv
```

Figure 41 Example Requests using the file extension convention

The file extension convention is an extremely useful one when accessing A-DAS™ using common office applications as many of these applications do not provide full HTTP support and rely on extension recognition. Older versions of Microsoft Excel for example do not make MIME specific request, nor do they identify themselves as Excel in the User-Agent header. Without this metadata to guide its choice of response MIME A-DAS™ responds using the default XML format. Because the default format

uses a microformat HTML Table Excel can process the return XML seamlessly, however, by place a **.xls** suffix on the URL A-DAS™ will respond with an Excel formatted file, which will including the additional formatting and highlighting that A-DAS™ supports.

5.3 Ad-hoc Querying

In addition to the predefined Data Views A-DAS™ provides support for *ad-hoc real-time queries*. Queries are written in a proprietary XML format that blends the rules and concepts of SQL with the URL based RESTful access of an A-DAS™ service. The syntax of the Query XML format in detail in Ref 1: *A-DAS™ Query Language Specification*. For the purposes of this document the query examples illustrated are relatively easy to follow and can be tested on the Anaeko sandbox installation at <http://sandbox.anaeko.com/query>.

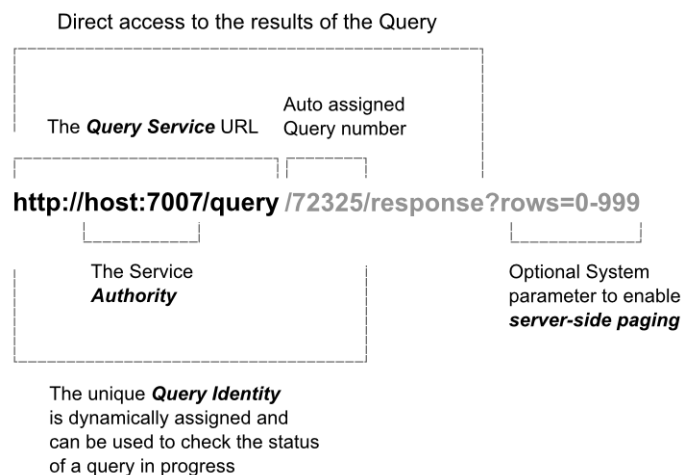


Figure 42 The anatomy of the A-DAS™ - Query URL

Queries are sent to A-DAS™ by making an HTTP POST request to the *Query Service URL*. The URL can be inferred for any A-DAS™ service by appending **/query** to the root Service URL, as illustrated in Figure 42.

By default all ad-hoc queries are executed *asynchronously*, although it is also possible to request that a query be executed *synchronous*, which will result in a blocking call to the service until the results are returned – note that the syntax for requesting a synchronous query is part of the Query definition. The results of an asynchronous query can be forwarded to a *call-back URL*, provided by the calling application, or they can be retrieved directly from the unique *Query Identity URL* that is returned in response to a successful POST request to the Query Service.

```
POST /query HTTP/1.1
host: sandbox.anaeko.com:8001
Content-Type: text/xml

<query limit="10">
<select>
  <target value="http://sandbox.anaeko.com:8001/email/sent" metadata="true" />
</select>
<where>
  <filter target="http://sandbox.anaeko.com:8001/email/sent/Timestamp"
    action="&gt;=" against="now(-1w)" />
  <filter target="http://sandbox.anaeko.com:8001/email/sent/Timestamp"
    action="&lt;=" against="now()" />
</where>
</query>

HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Content-Length: 40

http://sandbox.anaeko.com:8001/query/123
```

Figure 43 An example ad-hoc Query sent to the A-DAS™ Query URL

Figure 43 illustrates a complete **Request/Response** cycle for an *asynchronous* query, showing how A-DAS™ responds to the request by returning the unique auto-assigned URL. The status of the query can be checked in real-time using an HTTP GET to this URL or the results can be fetched directly by appending **/response** to the unique URL of the query: <http://sandbox.anaeko.com:8001/query/123/response>.

As with Data Views the results of an ad-hoc query can be fetched using server-side paging:

```
http://sandbox.anaeko.com:8001/query/123/response?rows=0-9
```

Figure 44 Request Ad-Hoc Query results in pages using the **rows** query parameter

To get the results of an asynchronous query forwarded when they are ready the calling application must provide a call-back address. Once the query results are complete A-DAS™ will forward them to the call-back address using an HTTP POST.

```
POST /query HTTP/1.1
host: sandbox.anaeko.com:8001
Content-Type: text/xml

<query limit="10" callback="none">
<select>
  <target value="http://sandbox.anaeko.com:8001/email/sent" metadata="true" />
</select>
<where>
  <filter target="http://sandbox.anaeko.com:8001/email/sent/Timestamp"
    action="&gt;=" against="now(-1w)" />
  <filter target="http://sandbox.anaeko.com:8001/email/sent/Timestamp"
    action="&lt;=" against="now()" />
</where>
</query>
```

continue on following page...

HTTP/1.1 200 OK
 Content-Type: application/xml; charset=utf-8
 Content-Length: 3346

```
<response status="SUCCESS_OK" id="http://sandbox.anaeko.com:8001/query/125"
success="true">
  <data xml.parser="com.anaeko.utils.data.xml.ArrayTableReader">
    <table footer="" title="" description="" id="table6">
      <thead>
        <th java.class="java.lang.String" id="table6/Id">Id</th>
        <th java.class="java.util.Date"
id="table6/Timestamp">Timestamp</th>
        <th java.class="java.lang.String" id="table6/To">To</th>
        <th java.class="java.lang.String" id="table6/From">From</th>
      </thead>
      <tbody>
        <tr id="row0">
          <td>Fc:167:3Cf</td>
          <td>2010-03-01 12:13:14</td>
          <td>Gail.Reeves@Arcam-Corporation.Com</td>
          <td>Britney.Mclaughlin@Help-Desk.Com</td>
        </tr>
        ...
      </tbody>
    </table>
  </data>
</response>
```

Figure 45 An example ad-hoc Query sent to the A-DAS™ Query URL synchronously

5.4 Working with Views

Although primarily used by clients to query data the A-DAS™ HTTP interface is not limited to data access. Each A-DAS™ service provides a range of Service Management capabilities including the option to explore, create, edit and delete Data Views.

Section 4.5 details how a Web Browser can be used to browse and examine an A-DAS™ View Catalogue. To *create*, *delete* and *edit* Views requires an HTTP client capable of HTTP PUT and DELETE operations, more commonly found in HTTP programming libraries and tools. Without these capabilities it is still possible to use simple HTTP GET requests to perform complex dynamic interactions with views, including discovery and parameter detection. Figure 46 shows the basic View URL structure for requesting the View Catalogue and a View’s definition and details.

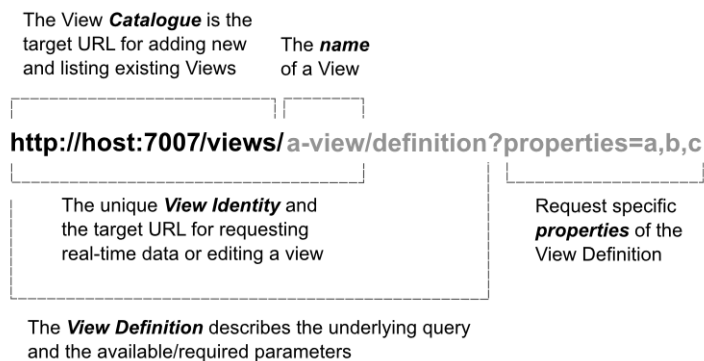


Figure 46 Anatomy of an A-DAS™ URL - View Catalogue

Unlike the data access URLs the View Management URLs respond using a fixed format; the A-DAS™ Metadata XML format. This is a proprietary format detailed in Appendix B: XML Formats. Figure 47 shows an example of how the Definition of a View might be queried to return the list of mandatory parameters defined for the View. Two calls are shown, the first requests the default XML format; the second requests the universal *name-value pair* format used to define properties.

```
GET /views/site-status/definition?properties=parameter%20list HTTP/1.1
host: sandbox.anaeko.com
```

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=UTF-8;
Content-Length: 214
```

```
<?xml-stylesheet type="text/xsl" href="/file/properties.xsl"?>
<properties>
  <property java.class="java.lang.String">
    <name>parameter list</name>
    <value>site</value>
  </property>
</properties>
```

```
GET /views/site-status/definition?properties=parameter%20list HTTP/1.1
host: sandbox.anaeko.com
Accept: application/properties
```

```
HTTP/1.1 200 OK
Content-Type: application/properties
Content-Length: 20
```

```
parameter list=site
```

Figure 47 Direct access to View Properties – list mandatory parameters

The View Management URLs also enable calling applications to inspect the structure of the data that will be returned by the View, without having to call the View and process the results. The structure and data types that are returned by a view can be accessed by appending `/metadata` to the end of the unique View URL. For example, <http://sandbox.anaeko.com:7007/views/site-status/metadata> will return the structure of the results returned by the [site-status](#) View.

5.4.1 Creating a View

New views can be created on an A-DAS™ service using the REST API. A view is a stored and optimised Query, assigned a unique URL. To *convert* a Query to a View POST the Query to the View Catalogue URL [/views](#). If the Query is valid A-DAS™ will respond with a unique URL for the newly created View. This unique URL will be the View Catalogue URL with an assigned *unique number* appended, for example: `http://sandbox.anaeko.com:7007/views/47382`. However, if the Query has a *unique name* as part of its definition this name can also be used to identify the

View, for example `http://sandbox.anaeko.com:7007/views/mail-sent-last-week` would be a valid View URL for a uniquely named Query “a-new-view”. Note that because the name of the View can be part of a URL it is recommended that View names are restricted [safe URL characters](#).

```
POST /views HTTP/1.1
host: sandbox.anaeko.com:8001
Content-Type: text/xml

<query name="mail-sent-last-week">
<select>
  <target value="http://sandbox.anaeko.com:8001/email/sent" metadata="true" />
</select>
<where>
  <filter target="http://sandbox.anaeko.com:8001/email/sent/Timestamp"
    action="&gt;=" against="now(-1w)" />
  <filter target="http://sandbox.anaeko.com:8001/email/sent/Timestamp"
    action="&lt;=" against="now()" />
</where>
</query>

HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Content-Length: 40

http://sandbox.anaeko.com:8001/views/123
```

Figure 48 An example of how an ad-hoc Query is converted to an A-DAS™ View

It is worth noting the similarity between the View Create request in Figure 48 and the ad-hoc Query request in Figure 43. This is a deliberate design decision to keep the API small and self-consistent as possible. The exact same POST request can be sent to the [/query](#) or the [/views](#) URL resulting in an asynchronous Query or a New View respectively.

5.4.2 Deleting a View

A view can be deleted from a View Catalogue, assuming the appropriate permissions are in place, by sending an HTTP DELETE request to the Views unique URL.

```
DELETE /views/mail-sent-last-week HTTP/1.1
host: sandbox.anaeko.com:8001

HTTP/1.1 200 OK
Content-Type: text/plain

Deleted View: http://sandbox.anaeko.com:8001/views/123
```

Figure 49 An example of how an A-DAS™ View is deleted using an HTTP DELETE

In Figure 49 the View created in section on Creating a View is deleted. A-DAS™ returns the assigned unique URL of the View as confirmation.

5.4.3 Editing a View

A-DAS™ supports editing of Views using the HTTP PUT operation. The HTTP protocol definition specifies that the PUT operation is a **complete replacement** of the existing resource, rather than partial edit. Partial edits might be performed using a POST but the semantics are subtly different and POST editing is not, currently, supported by A-DAS™. In effect editing an A-DAS™ view is similar to creating a new View at a pre-existing unique URL, there by replacing the original View.

```
PUT /views/mail-sent-last-week HTTP/1.1
host: sandbox.anaeko.com:8001
Content-Type: text/xml

<query name="mail-sent-last-week">
<select>
  <target value="http://sandbox.anaeko.com:8001/email/sent" metadata="true" />
</select>
<where>
  <filter target="http://sandbox.anaeko.com:8001/email/sent/Timestamp"
    action=">=" against="now(-4w)" />
  <filter target="http://sandbox.anaeko.com:8001/email/sent/Timestamp"
    action="<=" against="now()" />
</where>
</query>

HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Content-Length: 40

http://sandbox.anaeko.com:8001/views/124
```

Figure 50 An example of how to use an HTTP PUT request to edit an A-DAS™ View

6 Appendix A: URL Catalogue

Resource / Category	URL	Method			
		GET	PUT	POST	DELETE
Service	/	Service Context	405	405	405
	/?properties=name[,name...]	Properties List	405	405	405
	/?rel=[rel...]	Linked Metadata	405	405	405
	../?tag=[tag][,tag...]	Linked Metadata	405	405	405
	?relatesTo=[(rel tag)][,(rel tag)...]	Linked Metadata	405	405	405
Metadata Catalogue	/[servicename]	Metadata Catalogue	Edit the metadata	Edit Metadata	Remove Metadata
	/[servicename]?properties=name[,name...]	Properties List	405	Add/Edit Property	Remove Property
	/[servicename]?rel=[rel][,rel...]	Linked Metadata	405	Add/Edit Link	Remove Link
	/[servicename]?tag=[tag][,tag...]	Linked Metadata	405	Add/Edit Link	Remove Link
	/[servicename]?relatesTo=[(rel tag)][,(rel tag)...]	Linked Metadata	405	Add/Edit Link	Remove Link
Metadata	/[servicename]/[metadata]/[metadata...]	Metadata	Edit the metadata	Edit Metadata	405
	/[servicename]/[metadata]/[metadata...]?properties=name[,name...]	Properties List	405	Add/Edit Property	Remove Property
	/[servicename]/[metadata]/[metadata...]?rel=[rel...]	Linked Metadata	405	Add/Edit Link	Remove Link
	/[servicename]/[metadata]/[metadata...]?tag=[tag][,tag...]	Linked Metadata	405	Add/Edit Link	Remove Link
	/[servicename]/[metadata]/[metadata...]?relatesTo=[(rel tag)][,(rel tag)...]	Linked Metadata	405	Add/Edit Link	Remove Link
Queries	/query	Query (HTML) Form	405	Create New Query	405
	/query/[name id]	Query Details/Status	405	405	Cancel query
	/query/[name id]/response	Query Results	405	405	405
	/query/[name id]/response?[rows=n-m]	Inclusive rows n to m of Query Results	405	405	405

Continues on following page...

Resource / Category	URL	Method			
		GET	PUT	POST	DELETE
View Catalogue	/views	View Catalogue	405	Add New View	405
	/views/[(name id)]/definition	View Definition	405	405	405
	/views/[(name id)]/definition?properties=name[,name...]	Properties List	405	Add/Edit Property	Remove Property
	/views/[(name id)]/definition?rel=[rel...]	Linked Metadata	405	Add/Edit Link	Remove Link
	/views/[(name id)]/definition?tag=[tag][,tag...]	Linked Metadata	405	Add/Edit Link	Remove Link
	/views/[(name id)]/definition?relatesTo=[(rel tag)][,(rel tag)...]	Linked Metadata	405	Add/Edit Link	Remove Link
Views	/views/[(name id)][?[param=value[,value]][¶m=value[,value]]]	Execute View	Add/Replace View	Update View	Remove View
	/views/[(name id)][?limit=value]	Execute View (limit rows returned)	405	405	405
	/views/[(name id)][?http-accept=mime]	Execute View (override http mime)	405	405	405
	/views/[(name id)][?rows=n-m]	Execute View (return inclusive rows n to m)	405	405	405
	/views/[(name id)]/sample.data	View Sample	Add/Replace Sample Data	Update Sample Data	Remove Sample Data
View Metadata	/views/[(name id)]/metadata	View Metadata	405	405	405
	/views/[(name id)]/metadata[/metadata...]?properties=name[,name...]	View Metadata Properties List	405	Add/Edit Property	Remove Property
	/file/[static-file]	Cached Static Resource	405	405	405
Static Resources	/file/[static-file]?http-cache-control=no-cache	Fresh Static Resource	405	405	405

Continues on following page...

Resource / Category	URL	Method			
		GET	PUT	POST	DELETE
Service Statistics	/status	Service Status	405	Add/Edit Property	405
	/status?properties=name[,name...]	Properties List	405	Add/Edit Property	405
	/status/statistics	Service Statistics	405	Add/Edit Property	405
	/status/statistics?properties=name[,name...]	Properties List	405	Add/Edit Property	405
Service Reports	/status/reports	List Reports	405	405	405
	/status/reports/[name]	Report	405	405	405
	/status/reports/[name]?properties=name[,name...]	Properties List	405	405	405
	/status/reports/[name]?category=name[,name...]	Filtered Logs	405	405	405

Table 2 A complete list of the supported URL Templates

7 Appendix B: XML Formats

7.1 Microformats

Where possible A-DAS™ uses existing and well-documented formats and schemas. In particular it is common to find widely used, official or de-facto, XML standards that meet our requirements for metadata and data transfer. To reduce the cost of consuming A-DAS™ services, to promote interoperability, *proprietary XML formats are strongly discouraged.*

7.1.1 <link>

Attributes

Name	Description	Required
href	URL to the related resource	Yes
rel	The relationship between this resource and the linked resource, e.g. this resource <i>has an alternative representation</i> at the specified URL (href) – note that this is normally shortened to <i>alternative</i>	Yes
name	The name of the related resource, or the name given to this relationship	No
tags	Arbitrary comma separated tags that group and describe this link	No
rev	The relationship between the linked resource and this one, the inverse of the rel attribute	No
charset	The character set used to encode the linked resource, if it is restricted to text based MIMEs	No
type	The primary MIME type of the linked resource	No
id	A system wide unique identity for this link	No
title	Arbitrary text field associated with the <link>	No

Sub-Elements

Name	Description	Required
n/a	n/a	n/a

The official definition of the HTML [<LINK>](#) element can be found in the [HTML 4 specification](#). The use of the <LINK> (or <link>) in A-DAS™ is largely equivalent to this definition, with some minor qualifications and extensions.

The explicit intention of the <link> element is to enable two HTML Documents to be related in addition to any references that may be part of the documents content. As an element of HTML Metadata <link> elements only appear in the header of normal HTML documents. The A-DAS™ use of the <link> is analogous as it is used to define the relationship between two resources, most commonly two Metadata resources.


A-DAS™ extends the definition of the HTML <link> element by adding two optional attributes. The first is simply an optional *name* for the object of the relationship. The second provides support for the common Web grouping mechanism that enables arbitrary *tags* to be assigned to an entity for searching and grouping.

A common use of the <link> element in public web sites is as a mechanism for defining alternative URLs for use with non-standard or specialist clients. For example, many news web sites support RSS feeds as an alternative way of consuming the latest news items.

```
<link href="http://feeds.guardian.co.uk/theguardian/rss"
      rel="alternate"
      type="application/rss+xml"
      title="rss" />

<link
ref="http://newsrss.bbc.co.uk/rss/newsonline_uk_edition/front_page/rss.xml"
rel="alternate"
type="application/rss+xml"
title="BBC NEWS | News Front Page" />
```

Figure 51 Example HTML <link> elements taken from public web sites

In the above examples, taken from <http://www.theguardian.co.uk> and <http://news.bbc.co.uk> respectively the links relate the front page of the web site with an alternative representation in RSS format, suitable for an RSS reader. Modern browsers automatically detect the RSS <link> and offer the feed as an option, usually with an appropriate icon .

The concept and purpose of the HTML <link> is complementary to the RDF mechanism for relating two Entities together using [RDF Triples](#). In RDF an Entity known as *Employee* might be related to an Entity called *Job* by stating that “*An Employee has a Job*”. This statement is as Binary Predicate. A Predicate is a statement or function that is either true or false. A Binary Predicate is a statement or function that requires two operands, usually called the Subject and the Object. In the previous example the Subject is the *Employee* and the Object is the *Job*, and the Predicate “*has a*” is either true or false for all combinations of *Employee* and *Job*. In the example of RSS feeds and web pages the <link> element has be rephrased in the

terms of an RDF Triple by stating: *“This Document has an alternative RSS Document”*.

In A-DAS™ all relationships are Binary Predicate, which enables their direct translation to HTML <link>s. The mapping of A-DAS™ relationships to <link> can be expressed, in pseudo code, as follows:

```

Link ln = new Link;

ln.title = relationship.assertion;
ln.name = relationship.object.name;

if relationship.object has tags
  for each relationship.object.tag
    ln.tags += tag;
    ln.tags += , ;

ln.href = relationship.object.url;
ln.rel = relationship.name;

if relationship has an inverse
  ln.rev = relationship.inverse.name;
  
```

Figure 52 Pseudo code showing an A-DAS™ relationship mapped to an HTML <link>

Although A-DAS™ supports arbitrary relationships between resource, and different Data Sources may require customer relationships depending on underlying Data Model and API, there are a small number of core relationships that common across all A-DAS™ services.

Relationship	Description	Inverse Relationship
hasChild	<p>This represents the fundamental tree-like hierarchical relationship between metadata entities in a standard Data Model. It is generic in that it supports the creation of arbitrarily complex network of entities, adequately describing the basic structure of XML, Relational and Object data models.</p> <p>There are many semantically equivalent ways to express this generic relationship e.g.</p> <ul style="list-style-type: none"> • Parent – Child • Branch – Leaf • Is Made Of • Is Composed Of 	childOf

	Are just a few examples.	
hasParameter	This relationship is used to define entities that can be used to filter data. For example in a relational database the rows in a table can be queried using column values to filter the rows returned. Or similarly the parameters of a SOAP service are filters on the data set returned.	parameterOf
indexes	This is a specialised form of the parent-child relationship that expresses the connection between a Catalogue and its contents.	indexedBy
servedBy	Another specialised parent-child relationship that defines the relationship between a Service and its Resource. This is generally reserved for root service entities as navigation from the extremities of the hierarchy should always be possible through the standard parent-child and index relationships.	n/a
hasMetadataCatalogue	Reserved for Service entities that manage catalogues	n/a
hasViewCatalogue	Reserved for Service entities that manage catalogues	n/a

Table 3 Key Resource relationships in A-DAS™

7.1.2 <table>

Attributes

Name	Description	Required
id	A system wide unique identity for this link	Yes
title	A display title for the data in the table	No
name	The name of the data set	No
description	A description of the contents of the table	No

Sub-Elements

Name	Description	Required
<thead>	Wrapper for the table's columns	No
<tbody>	Wrapper for the table's rows of data	No

The HTML [<TABLE>](#) element is used by A-DAS™ to exchange tabular data in XML format. An A-DAS™ XML table does not make use of the majority of attributes associated with the standard HTML table but is still a valid <TABLE> in that can be rendered by any HTML compliant browser.

An A-DAS™ <table> is a simplified version of and an HTML <TABLE>; the key structural aspects are identical as A-DAS™ tables make use of the HTML sub-elements [<TH>](#), [<TR>](#) and [<TD>](#), which define the columns, rows and cells of the table respectively. However, an A-DAS™ table is a strict XML structure that does not support the more generous parsing allowed for HTML tables.

An example of an A-DAS™ <table> is given in Figure 53:

```
<table name="" title="" description="" id="table62">
  <thead>
    <th id="table62/Site">Site</th>
    <th id="table62/Contact">Contact</th>
    <th id="table62/Telephone">Telephone</th>
    <th id="table62/Email">Email</th>
    <th id="table62/Subnet">Subnet</th>
  </thead>
  <tbody>
    <tr id="row0">
      <td>Site-0000</td>
      <td>Joy Kimble</td>
      <td>07557444425</td>
      <td>joy.kimble@primatech.com</td>
      <td>10.0.0.0/26</td>
    </tr>
    ...
  </tbody>
</table>
```

Figure 53 Example of a <table> element

There are no restrictions placed on the attributes allowed in the <table> element only the *id* attribute is mandatory. This attribute is also the root of all Column identities within the table, as illustrated in the example.

The <table> element *must* contain a <thead> sub-element if there is any data in the table's body or if the table's columns have been defined. If there is data in the table there must also be a <tbody> sub-element, to enclose the rows of data.

7.1.3 <thead>

Attributes

Name	Description	Required
n/a	n/a	n/a

Sub-Elements

Name	Description	Required
<th>	A table column	No

This is a mandatory structural element, analogous to the HTML [<THEAD>](#) element, that only exists within an enclosing <table>. It supports no attributes and can contain only <th> sub-elements.

```
<thead>
  <th id="table62/Site">Site</th>
  <th id="table62/Contact">Contact</th>
  <th id="table62/Telephone">Telephone</th>
  <th id="table62/Email">Email</th>
  <th id="table62/Subnet">Subnet</th>
</thead>
```

Figure 54 Example of a <thead> element

7.1.4 <tbody>

Attributes

Name	Description	Required
n/a	n/a	n/a

Sub-Elements

Name	Description	Required
<tr>	A row of data	No

This is a mandatory structural element, analogous to the HTML [<TBODY>](#) element, that only exists within an enclosing <table>. It supports no attributes and can contain only <tr> sub-elements.

```

<tbody>
  <tr id="row0">
    <td>Site-0000</td>
    <td>Joy Kimble</td>
    <td>07557444425</td>
    <td>joy.kimble@primatech.com</td>
    <td>10.0.0.0/26</td>
  </tr>
  <tr id="row1">
    <td>Site-0001</td>
    <td>Melissa Dill</td>
    <td>02980885038</td>
    <td>melissa.dill@elbrus-global.com</td>
    <td>10.0.1.48/28</td>
  </tr>
  ...
</tbody>

```

Figure 55 Example of a <tbody> element

7.1.5 <th>

Attributes

Name	Description	Required
id	An identity String that is unique within the enclosing <table> and is prefixed with identity of the <table>	Yes
java.class	<p>A java class that sets boundaries on the type of data that can be contained in the column. As all data in the table will be serialised to text this type indicates how the text should be converted back into its original type.</p> <p>The rules for text serialisation are defined elsewhere.</p> <p>Mappings between java types and other systems are already well documented for numerous programming and database systems*. For this reason it is unnecessary and counter productive to create a new A-DAS™ specific set of types or to adopt an arbitrary 3rd party type system such as defined by XML Schema.</p> <p>*For example, here are references to mappings from Java types to: SQL, XML Schema, SOAP and C/C++</p>	No
xml.parser	An XML parser class to use when reading the	No

	<p>contents of a cell in this column.</p> <p>This is a mandatory attribute for cells that contain non-standard complex types. If not specified the contents of the cell can only be read as a String data type.</p> <p>Internally A-DAS™ does not use this information to handle complex types; this is strictly an import/export instruction.</p>	
name	The name of the column, if the full, display name contains <u>reserved or “unwise” URI characters.</u>	No
highlights	<p>Deprecated: Base64 encoded display rules for colour highlighting the table’s cells. The format for the unencoded rules is described in the Appendix on Proprietary Formats.</p> <p>Rules are also applied to the appropriate <td> cell using the standard HTML/CSS style attribute. These styles should be used in preference to the non-standard embedded highlights attribute.</p>	No

Sub-Elements

Name	Description	Required
n/a	n/a	n/a

This is a mandatory sub-element of all tables that serves the same purpose as the HTML [<TH>](#) element. It defines the metadata for the table’s cells, what are they called and what type of data do they contain. Common synonyms for the <th> element are **Column** or **Header**, and in Relational Theory they are often referred to as **Relational Attributes**.

In A-DAS™ the <th> element is analogous to the concept of the Relational Attribute, where a single Attribute is defined completely by its Name and its Type. A set of one or more Attributes defines the **header** or rules of a Relation and the Tuples that conform to these rules are the **body** or rows of the Relation.

```

<th java.class="java.lang.String"
    id="table62/Customer-Site"
    name="Customer-Site">Customer Site</th>
<th java.class="java.lang.String"
    id="table62/Contact-Name"
    name="Contact-Name">Contact Name</th>
<th java.class="java.lang.Integer"
    highlights="PGhpZ2ZGOTkwMCIgd2hlbj0iJ ... pZ2ZGO="
    id="table62/Category">Category</th>
<th java.class="java.lang.String"
    id="table62/Email">Email</th>
<th xml.parser="com.example.parser.HTML"
    id="table62/Subnet"
    name="Email-Template">Email Template</th>

```

Figure 56 Examples of the <TH> element showing the use of various attributes

One consequence of this interpretation is that A-DAS™ considers two header elements *equivalent* if both the name and type attributes match.

7.1.6 <tr>

Attributes

Name	Description	Required
id	<p>An identity for the row that is guaranteed unique within the scope of the enclosing <tbody>.</p> <p>It is possible to split this identity and extract the original row index of the element as it was defined in the source data set, e.g. using JQuery:</p> <pre>var index = \$(row).attr(id).substring(3);</pre> <p>Note that A-DAS™ table row indices are <i>zero-based</i>.</p>	Yes

Sub-Elements

Name	Description	Required
<td>	A cell element that contains data	No

This is a mandatory structural element, analogous to the HTML [<TR>](#) element, that only exists within an enclosing <tbody>. It supports one mandatory identity attribute and can contain zero, one or more <td> sub-elements.

```

<tr id="row0">
  <td>Site-0000</td>
  <td>Joy Kimble</td>
  <td>07557444425</td>
  <td>joy.kimble@primatech.com</td>
  ...
</tr>

```

Figure 57 Example of a <tr> element, including <td> sub-elements

7.1.7 <td>

Attributes

Name	Description	Required
columnId	<p>An identity for the Column that this cell belongs in. This offers an alternative mechanism for column selection, potentially useful when using client-side JavaScript</p> <p>For example it is possible to select all the cells in a column, without knowing the structure of the table in advance, using JQuery:</p> <pre>var contacts = \$("tr[columnId =xxx]");</pre>	No
style	<p>Part of the standard HTML specification for <TD> elements this attribute support in-line styles. A-DAS™ supports cell and row highlighting rules in its Queries and Views. If defined the rules will be applied to the serialised XML and stored in the style attribute of the cells.</p>	No

Sub-Elements

Name	Description	Required
<td>	A cell element that contains data	No

This is *the* core data element used in tables, analogous to the HTML [<TD>](#) element, it only exists within an enclosing <tr>. It supports two optional attributes and contains the data elements of serialized as text. The serialisation is performed according to the rules defined in the associated Column Header, <th> element, either as a java.class or as a complex type with an associated xml.parser.


```

<td>Site-0012</td>
<td>Joy Kimble</td>
<td columnId="table62/Telephone">07557444425</td>
<td style="background-color: #FF7070">joy.kimble@primatech.com</td>

```

Figure 58 Example <td> elements, illustrating the key attributes

It is possible to support non-standard, complex, data types by using XML serialisation within the cell. If a cell contains serialised XML data the appropriate column *must* contain the instructions for de-serialisation, using the `xml.parser` attribute.

For details of the serialisation rules refer to the Appendix on Data Types.

7.2 Proprietary Formats

7.2.1 <response>

Attributes

Name	Description	Required
id	<p>The identity of the response, which will be the unique URI that generated the response.</p> <p>If the response is an error it will always be the URL that was requested. If the response is a success the URL will be the canonical URL of the resource that generated the response, this may not be the same as the URL that the client requested.</p>	Yes
status	<p>This attribute represents the status of the response. Strictly speaking it is a redundant element as the status code will <i>always</i> be part of the HTTP response return by A-DAS™ however, not all clients will be fully HTTP compliant and for this reason the status is appended as a response attribute.</p> <p>Note that the HTTP response code takes precedence and the presence of the status attribute is not guaranteed.</p>	No

success	<p>If present this attribute is always set to <i>true</i>. This is an optional element as the status value should allow the client to check if the response is an error or not, however, testing for the presence or absence of this attribute is more efficient for handling errors.</p> <p>This attribute cannot be present at the same time as the <i>error</i> attribute.</p>	No
error	<p>If present this attribute is always set to <i>true</i>. This is an optional element as the status value should allow the client to check if the response is an error or not, however, testing for the presence of this attribute is more efficient for handling errors.</p> <p>This attribute cannot be present at the same time as the <i>success</i> attribute.</p>	No

Sub-Elements

Name	Description	Required
<metadata>	Metadata relating to the Response, which may include one or more <link> elements that reference other resources and one or more <property> elements.	No
<data>	A container element that wraps the payload of the response.	No

The <response> element is the default envelope for A-DAS™ service responses. It is used to wrap both successful service calls and error responses and for this reason is can be considered a meta or structural element.

It is possible to argue that the <response> element is a redundant envelope considering that the HTTP response can contain all of the necessary metadata, however it is an *optional* element that enables future expansion of the response to include additional <link> elements, and other metadata. This is in-line with the RESTful principle of Hypermedia, which requires that a Service response should enable state in stateless client-service interactions by providing links and options for the next state transition.

```

<?xml-stylesheet type="text/xsl" href="/file/response.xsl"?>
<response status="SUCCESS_OK"
          id="http://sandbox.anaeko.com:7007/query/1863"
          success="true">

  <metadata />
  <data xml.parser="com.anaeko.utils.data.xml.ArrayTableReader">
    <table id="2323">
      <thead>
        ...
      </thead>
      <tbody>
        ...
      </tbody>
    </table>
  </data>
</response>

<?xml-stylesheet type="text/xsl" href="/file/response.xsl"?>
<response status="CLIENT_BAD_REQUEST"
          id="http://sandbox.anaeko.com:7007/query"
          error="true">

  <data java.class="java.lang.String">
    Query is invalid, Content-Type: null is NOT supported
  </data>
</response>

```

Figure 59 Example <response> envelopes, with a error/data payload

The role of document links in the interaction between the service and the client is enshrined in the, rather oblique, statement often quoted from [Roy Fielding's Theses](#) on REST:

“Hypermedia as the engine of application state”.

Roy Fielding - Representational State Transfer (Chapter 5)

The role of the A-DAS™ <response> element is analogous to the root <HTML> tag where the <metadata> and <data> elements assumes the role of the HTML <HEAD> and <BODY> elements respectively. If we follow the *microformat* principles we might expect A-DAS™ to reuse the HTML tags rather than invent a proprietary XML format, however, an A-DAS™ Response is *not* an HTML document. The HTTP Content-Type sent with all A-DAS™ responses indicates the MIME type of the response as application/xml, and this MIME type should not in general be treated as HTML, although in the case of an A-DAS™ response this is a valid fallback option. A typical <response>, as illustrated in Figure 59, will include a suitable XSL header for transforming the data. This header will reference an XSL transformation that has been tailored for the client and will enable an XSLT capable client to render the response for display.

Note that the `<response>` element is defined as an *optional* element and clients should not rely on its presence, or by association the presence of a `<metadata>` or `<data>` element.

7.2.2 `<data>`

Attributes

Name	Description	Required
<code>java.class</code>	<p>A java class that sets boundaries on the type of data that can be contained in the <code><data></code> element. As all data will be serialised to text this type indicates how the text should be converted back into its original type.</p> <p>The rules for text serialisation are defined elsewhere.</p> <p>Mappings between java types and other systems are already well documented for numerous programming and database systems*. For this reason it is unnecessary and counter productive to create a new A-DAS™ specific set of types or to adopt an arbitrary 3rd party type system such as defined by XML Schema.</p> <p>*For example, here are references to mappings from Java types to: SQL, XML Schema, SOAP and C/C++</p>	No
<code>xml.parser</code>	<p>An XML parser class to use when reading the contents of the <code><data></code> element.</p> <p>This is a mandatory attribute for <code><data></code> elements that contain non-standard complex types. If not specified the contents of the <code><data></code> element can only be read as a String data type. Internally A-DAS™ does not use this information to handle complex types; this is strictly an import/export instruction.</p>	No

Sub-Elements

Name	Description	Required
<code><table></code>	<p>The data element can contain any parsable sub-elements, depending on the <code>xml.parser</code> declaration. However, the most common sub-element is the <code><table></code>, see Section 7.1.2 for details.</p>	No

The `<data>` element is a container for arbitrary text or XML content. The actual content is determined by the declared `xml.parser` or `java.class` attributes. In practice the most likely content of a `<data>` element is either a `<table>` or a textual message from the service, usually an error message.

```
<data java.class="java.lang.String">
  Unknown resource @ /rubbish
</data>

<data xml.parser="com.anaeko.utils.data.xml.ArrayTableReader">
  <table id="2323">
    <thead>
      ...
    </thead>
    <tbody>
      ...
    </tbody>
  </table>
</data>
```

Figure 60 Example `<data>` elements showing an error response

The examples in Figure 60 show the two most typical responses; an error message and a `<table>` data set. The former is marked with the attribute `java.class` while the latter includes the `xml.parser` attribute. Both of these attributes enable the client to process the `<data>` payload when the element is enclosed in a larger Hypermedia document.

7.2.3 `<metadata>`

Attributes

Name	Description	Required
name	The name of the item of metadata, which is also the last segment of the Relative URI.	Yes
base_uri	The root URI of this metadata, typically this is the URL of the Metadata Catalogue to which this metadata belongs.	Yes
relative_uri	The relative URL of the metadata, with respect to the <code>base_uri</code> . This will be the same as the <code>name</code> attribute for metadata that is directly related to the <code>base_uri</code> .	Yes

xml.parser	A specific xml parser to use when the metadata entity can be treated as more than generic metadata. In general this is discouraged as specific types of Metadata cannot normally be instantiated by external A-DAS™ services.	No
-------------------	---	----

Sub-Elements

Name	Description	Required
<properties>	A container element that encloses a collection of <property> elements that define the <i>scalar</i> properties of this metadata entity.	No
<relatesTo>	A container element that encloses a collection of <link> elements that relate this item metadata to other metadata entities.	No

For an A-DAS™ service the <metadata> element is a *fundamental* Hypermedia *Resource*. It is central to the identification, navigation and automatic processing of the dynamic data resources that enable the inter-node cooperation that is essential in an A-DAS™ Data Federation.

Due to the dynamic nature of the cooperating services that make up A-DAS™ it is not possible to know in advance, at any one time, what data is available and what form it takes. The <metadata> element is a *self-describing* Hypermedia resource that enables automatic discovery and processing through the collection of <link> elements that relate one metadata entity to another and the scalar <property> elements that describe the nature of the metadata.

```
<metadata xml.parser="com.anaeko.utils.metadata.xml.in.GenericMetadataReader"
  name="sent"
  base_uri="http://sandbox.anaeko.com:8001/email"
  relative_uri="sent">

  <properties>
    <property java.class="java.lang.Integer" name="excel.sheet" value="0" />
    <property java.class="java.lang.String" name="title" value="Email Log" />
    <property java.class="java.lang.String" name="qname" value="sent" />
    <property java.class="java.lang.String" name="description" value="..." />
    <property java.class="java.lang.String" name="name" value="sent" />
    <property java.class="java.lang.Integer" name="cardinality" value="22299" />
    <property java.class="java.lang.Boolean" name="queryable" value="true" />
  </properties>

  <relatesTo>
    <link name="From" href="/email/sent/From" rel="hasChild" rev="childOf" />
    <link name="Id" href="/email/sent/Id" rel="hasChild" rev="childOf" />
    <link name="To" href="/email/sent/To" rel="hasChild" rev="childOf" />
    <link name="email" href="/email" rel="indexedBy" rev="indexes" />
  </relatesTo>
</metadata>
```

Figure 61 Example <metadata> element showing a collection of Properties and Links

The genesis of the A-DAS™ <metadata> Hypermedia is the [Resource Description Framework](#) (RDF), which is a general purpose modelling system for describing entities and the relationships between them. Like RDF A-DAS™ metadata is an abstract modelling framework that can, but does not have to, be expressed in XML format. Unlike RDF the A-DAS™ metadata system has a targeted, specific purpose and platform. A-DAS™ metadata is designed to provide a simple unified modelling construct that is fully RESTful, both in concept and in execution, with addressable entities, references and Hypermedia representations.

7.2.4 <properties>

Attributes

Name	Description	Required
n/a	n/a	N/a

Sub-Elements

Name	Description	Required
<property>	Zero, one or more <property> elements may be sub-elements of a <properties> collection.	No

The properties element is a structural element the sole purpose of which is to enclose, or group, a collection of <property> elements. It does not have any attributes and can contain only <property> elements, as illustrated in Figure 62.

A <properties> element can be returned as the root of an A-DAS™ response or enclosed in a <data> element, itself enclosed in a root <response> element or as part of a <metadata> resource.

```
<properties>
  <property java.class="java.lang.Integer" name="index" value="3" />
  <property java.class="java.lang.String" name="qname" value="SLA Category" />
  <property java.class="java.lang.String" name="name" value="category" />

  <property java.class="java.lang.Boolean">
    <name>queryable</name>
    <value>>true</value>
  </property>

  <property xml.parser="com.anaeko.utils.data.ArrayTable">
    <name>value-map</name>
    <value>
      <table id="map1001">
        <thead>
          <th java.class="java.lang.String" id="map1001/key" >key</th>
          <th java.class="java.lang.Integer" id="map1001/value">value</th>
        </thead>
        <tbody>
```

```

        <tr>
          <td>Category 1</th>
          <td>1</th>
        </tr>
        <tr>
          <td>Category 2</th>
          <td>2</th>
        </tr>
        <tr>
          <td>Category 3</th>
          <td>3</th>
        </tr>
      </tbody>
    </table>
  </value>
</property>
</properties>

```

Figure 62 Example <properties> collection, showing the two supported formats for <property> sub-elements

Note that the properties contained in a <properties> collection do not have to have values that are simple types, such as String or Integer, but can also be complex types – although this is more unusual.

7.2.5 <relatesTo>

Attributes

Name	Description	Required
n/a	n/a	N/a

Sub-Elements

Name	Description	Required
<link>	Zero, one or more <link> elements.	No

The <relatesTo> element is a structural element the sole purpose of which is to enclose, or group, a collection of <link> elements. It does not have any attributes and can contain only <link> elements. It is possible, depending on the nature of the request, for A-DAS™ to return a <relatesTo> element as the root of a response, enclosed in a <data> element, itself enclosed in a root <response> element or as part of a <metadata> resource.

Note that the <relatesTo> element can contain multiple <link> elements that reference the same resource. In Figure 63 the /email/sent/From resource is linked as both a related “child” and as a “parameter”. This is typical of A-DAS™ metadata relationships where the structural connection between data entities is represented as a

“parent-child” relationship. Similarly the `hasParameter` link indicates that A-DAS™ supports the use of the linked entity as a filter when querying.

```
<relatesTo>
  <link name="From" href="/email/sent/From" rel="hasChild" rev="childOf"/>
  <link name="To" href="/email/sent/To" rel="hasChild" rev="childOf"/>
  <link name="From" href="/email/sent/From" rel="hasParameter" />
  <link name="email" href="/email" rel="indexedBy" rev="indexes"/>
</relatesTo>
```

Figure 63 Example `<relatesTo>` collection

7.2.6 <property>

Attributes

Name	Description	Required
name	The name of the property is a required attribute if a <code><name></code> sub-element has not been specified.	No
value	The value for the property, serialised as text. Note that XML content is not allowed in attributes, even if it is escaped. This attribute and the <code><value></code> sub-element are mutually exclusive.	No
java.class	<p>A java class that sets boundaries on the type of data that can be contained in the property value attribute or the <code><value></code> element. As all values set in properties will be serialised to text this type indicates how the text should be converted back into its original type.</p> <p>The rules for text serialisation are defined elsewhere.</p> <p>Mappings between java types and other systems are already well documented for numerous programming and database systems*. For this reason it is unnecessary and counter productive to create a new A-DAS™ specific set of types or to adopt an arbitrary 3rd party type system such as defined by XML Schema.</p> <p>*For example, here are references to mappings from Java types to: SQL, XML Schema, SOAP and C/C++</p>	No

xml.parser	<p>An XML parser class to use when reading the contents of the <code><value></code> element enclosed by this property.</p> <p>This is a mandatory attribute for properties that contain values with non-standard complex types. If not specified the contents of the <code><value></code> element can only be read as a String data type. Internally A-DAS™ does not use this information to handle complex types; this is strictly an import/export instruction.</p>	No
-------------------	---	----

Sub-Elements

Name	Description	Required
<code><name></code>	A single instance of this element is allowed if a <i>name</i> attribute has <i>not</i> been specified.	No
<code><value></code>	A single instance of this element is allowed if a <i>value</i> attribute has <i>not</i> been specified.	No

The `<property>` element is a generic device for encapsulating name-value-pairs, these may be parameters of a Request, scalar values associated with an item of metadata, a statistical value indicating the performance of an A-DAS™ service or a configuration option of a Data View.

The `name` attribute is a text value, typically but not necessarily unique in the context of a given Request/Response. The value can be a primitive type supported by A-DAS™, serialised as a String value, or it can be a complex type that has support for XML serialisation. Which of these two options is employed specific to the individual `<property>` element and can be inferred by the presence or absence of the `xml.parser` attribute.

```

<property java.class="java.lang.Long">
  <name>service.timestamp</name>
  <value>1267118746662</value>
</property>

<property java.class="java.lang.Long"
  name="service.timestamp"
  value="1267118746662"
/>

```

Figure 64 The same `<property>` in the two supported XML representations

If neither the `xml.parser` nor the `java.class` attributes are set in a given property the value will be treated as a String value, including line-breaks. If a type or XML

attribute is defined for a property but the value fails to parse as the specified type the XML is considered invalid and will be rejected.

```
<property xml.parser="com.anaeko.utils.xml.PropertyReader">
  <name>an embedded property</name>
  <value>
    <property java.class="java.util.Date">
      <name>timestamp</name>
      <value>2010-01-11 12:22:31.231</value>
    </property>
  </value>
</property>
```

Figure 65 <property> values supported complex types using embedded XML values

8 Appendix C: Glossary

8.1.1 General Terms

These are terms that have internationally-recognised definitions.

Term	Description
(HTTP) GET	<p>The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI. The Media Type is dependant on the entity being retrieved, the transformation capabilities of the service and the clients preferred media type.</p> <p>See: rfc2616</p> <p>This is the most used and most intuitive HTTP Method. However, it should be noted that there are many subtleties to the semantics of a GET request, which are detailed in the RFC.</p>
(HTTP) PUT	<p>The PUT method requests that the enclosed entity be stored under the supplied Request-URI. If the Request-URI refers to an already existing resource, the enclosed entity <i>should</i> be considered as a modified version of the one residing on the origin server. If the Request-URI does not point to an existing resource, and that URI is capable of being defined as a new resource.</p> <p>See: rfc2616</p> <p>The semantics of the PUT method are very specific and quite restrictive. It is worth noting that an equivalent POST readily replaces many of the operations that PUT can perform. However, a PUT request is guaranteed to be <i>Idempotent</i>, where a POST request is not.</p> <p>A-DAS™ provides limited support for PUT requests, most notably when creating new Views.</p>
(HTTP) POST	<p>The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI.</p> <p>The actual function performed by the POST method is determined by the server and is usually dependent on the Request-URI. The posted entity is subordinate to that URI in the same way that a file is subordinate to a directory containing it.</p>

	<p>See: rfc2616</p> <p>A-DAS™ treats a POST request as a submission to create a new resource or to alter an existing one. It is a method for editing or annotating service resources, where this is allowed.</p>
(HTTP) DELETE	<p>The DELETE method requests that the service delete the resource identified by the Request-URI. The client cannot be guaranteed that the operation has been carried out, even if the status code returned from the origin server indicates that the action has been completed successfully.</p> <p>See: rfc2616</p>
(HTTP) HEAD	<p>The HEAD method is identical to GET except that the server <i>must not</i> return a message-body in the response.</p> <p>See: rfc2616</p>
(HTTP) OPTION	<p>The OPTIONS method represents a request for information about the communication options available.</p> <p>See: rfc2616</p>
Microformat	<p>An existing and well established Resource format, typically but not exclusively in XML, that can be repurposed for general use. A microformat is specific, self-contained, subset of a large specification that when taken in isolation still maintains its context and meaning.</p> <p>For example, the HTML <TABLE> element is both an efficient and an effective XML data format for storing and sharing tabular data, as long as strict XML correctness is observed. To create an alternative XML format, with out very clear and specialized requirements, would be a waste of effort.</p> <p>For further information, examples and discussion visit: http://microformats.org/</p>
REST / RESTful	<p>A descriptive term for a service that follows the principles of self-discovery, decoupling and interoperability enshrined in Roy Fielding's Theses on the architectural style that drives the scalability of the Web. The key characteristics of a REST, or RESTful, Service can be summarized as:</p> <ol style="list-style-type: none"> 4. Identification of resources 5. Manipulation of resources through representations 6. Self-descriptive messages

	<p>7. Hypermedia as the engine of application state. In practice a RESTful <i>Web</i> Service will:</p> <p>8. Manage and serve Resources. It will not provide Process Oriented services</p> <p>9. <i>Every</i> Resource must have an Identity which in the means that every Resource will have at least one unique URL</p> <p>10. Resources should be linked together, even if the option is just to go back to the previous Resource</p> <p>11. Provide a Uniform Interface to operate on the Service’s Resources, at a minimum HTTP GET and POST but preferably PUT, DELETE, HEAD and OPTION for full HTTP compliance</p> <p>12. Provide alternative/multiple representations of Resources to suit the Client by fully supporting the HTTP Accept header and Content-Type MIMEs</p> <p>13. Support caching and intermediaries by processing Client requests statelessly</p>
Hypermedia	<p>A general term used to describe data or content that contains within itself links to other data/content. It is an extension of Hypertext applied to arbitrary media, such as audio or video.</p>

8.1.2 System Specific Definitions

These are terms that have been defined for the purpose of clarifying the requirements. They may have different meanings outside the scope of the project and may be referred to differently in the user interface and documentation, but the definitions provided here shall be used throughout the project scope.

Term	Description
URI	<p>In general the URIs referred to in this document can be assumed to be URLs as A-DAS™ does not support URIs that cannot be addressed or accessed in some meaningful way.</p> <p>See rfc3986</p>
URL	<p>Although not normally the case A-DAS™ URLs are interchangeable with URIs (see above)</p> <p>See rfc3986</p>
Metadata	<p>This term is used in its strict sense, meaning “Data about Data” but it is also used when referring to data entities in the abstract. For example there are almost 30million</p>

	addresses in the UK but the <i>Address</i> as an entity that defines the form that each of these 30million addresses take is considered to be Metadata. In programming terms there are parallels with this use of Metadata and the Object Oriented concept of a <i>Class</i> .
Service Context	Metadata related to an A-DAS™ service configuration, for example the service’s name, host and port.
Data View / View	A particular view of one or more data sets, often combined, filtered and transformed for a particular User, client or purpose. An A-DAS™ View is a cached query that has been assigned a unique, reusable URI. A-DAS™ View URLs can be shared, embedded and used in any HTTP capable application.
Media Type	A synonym for MIME type. A Media Type may define more than just the format of the data it may also include additional encoding and contextual information. For example, text based Media Types should include the encoding information, in MIME format: <code>text/html; charset=UTF-8</code>
Property	A property is a scalar value that has been assigned a name. It is often referred to as a name-value-pair. The value of a property may be a URL but this does not mean it is a Link to a Resource, it merely represents the String value of URL.
Link / Reference / Relationship	<p>Interchangeable terms for a reference that relates one resource to another. This is equivalent to an RDF Triple or a Binary Predicate, where it is asserted to be true that a <i>Subject</i> has some <i>Relationship</i> to an <i>Object</i>. In A-DAS™ all Subjects and Objects are addressable through URLs with the consequence that any relationship between two resources is also a URL/HTML Link between them.</p> <p>The HTML specification provides for an equivalent concept, the Document Relationship, using the <LINK> construct.</p>
Data Source	An original, usually unique, source of data that may or may not have a programming, query or service interface. For example, a Relational Database or an Excel spreadsheet.
Data Model / Metadata Catalogue	The use of the terms Data Model and Metadata Catalogue are interchangeable. An A-DAS™ Data Service will provide a representation of the available data, the Data Model, in the form of a Metadata Catalogue.

Data Service	A network enabled, usually supporting the HTTP application protocol, service that serves data in response to client queries.
(Data Source) Adapter	A pluggable A-DAS™ module that provides a bridge between an underlying Data Source and the A-DAS™ Data Service interface. An Adapter translates requests to and replies from a Data Source enabling a Data Service to be created. An Adapter is also responsible for the Metadata model that describes the available data.
Data Federation	A collection of data services that can be centrally or individually managed and can cooperate, exchanging data and delegating requests peer-to-peer, in such a way as to present a single unified Data Service.
A-DAS™ Hub (Service)	A central management service in a Data Federation. This service does not provide access to its own Data Source but as a member of the Federation it can serve data from one or more of the other Data Services.
A-DAS™ Node (Service)	This is Data Service in a Data Federation that takes part in the peer-to-peer processing of client requests. It typically provides access to one or more Data Source, although this is not a requirement.