

# The Distributed Annotation System

Robin D. Dowell<sup>1</sup>, Rodney M. Jokerst<sup>1</sup>, Allen Day<sup>2</sup>, Sean R. Eddy<sup>1</sup>, and Lincoln Stein<sup>2</sup> \*

<sup>1</sup>{robin, jokerst, eddy}@genetics.wustl.edu  
*Howard Hughes Medical Institute  
Department of Genetics,  
Washington University, St. Louis, MO 63110 USA*

<sup>2</sup>{day, lstein}@cshl.org  
*Cold Spring Harbor Laboratory,  
1 Bungtown Road, Cold Spring Harbor, NY 11724 USA*

## Abstract

### Background

Currently, most genome annotation is curated by centralized groups with limited resources. Efforts to share annotations transparently among multiple groups have not yet been satisfactory.

### Results

Here we introduce a concept called the Distributed Annotation System (DAS). DAS allows sequence annotations to be decentralized among multiple third-party annotators and integrated on an as-needed basis by client-side software. The communication between client and servers in DAS is defined by the DAS XML specification. Annotations are displayed in layers, one per server. Any client or server adhering to the DAS XML specification can participate in the system; we describe a simple prototype client and server example.

### Conclusions

The DAS specification is being used experimentally by Ensembl, WormBase, and the Berkeley Drosophila Genome Project. Continued success will depend on the readiness of the research community to adopt DAS and provide annotations. All components are freely available from the project website <http://www.biodas.org/>.

---

\*To whom correspondence should be addressed. Tel: +1 516 367 8380; Fax: +1 516 367 8389.

## Background

With the rise of computational biology and the decrease in hardware costs, high throughput annotation is now possible within many laboratories. They can now annotate entire genomes relatively quickly and efficiently. What has not kept up with the pace of annotation is the ability for multiple groups to exchange and compare their data, leading to fragmentation of annotation information among multiple databases and web sites, and to a certain level of frustration among the bench biologists who are the intended beneficiaries of this data.

Ideally, an annotation system should give individual experts the ability to contribute to the collective annotation in a quick, robust, and mostly painless fashion. They should have complete control over their annotations in order to keep them current and relevant. These annotations should not need approval from a central authority. Simultaneously, it should be easy for a user to obtain and visualize the most recent data about their particular region of interest. Users would also prefer not to be swamped by bogus information. Unfortunately, these goals seem to be at odds in the current sequence annotation environment.

Initial database efforts were largely centralized repositories such as GenBank, established in 1982 [1]. These databases act primarily as archival storage of sequence information. Consequently, each entry is owned by the sequence provider and integrating annotation information is, by design, nearly impossible.

A number of specialized databases have developed to serve a curatorial role within particular communities, such as Swissprot [2], Refseq [3], and WormPD [4]. A *C. elegans* database (ACeDB) is one particularly successful community database [5] (<http://www.acedb.org/>). It has served as the central database of phenotyping, bibliographic, mapping, and sequencing information for the *Caenorhabditis elegans* community since 1990 [6]. Individuals are encouraged to submit annotations and changes to the central database curatorial group. The group then reviews the request and decides what and how it is to be incorporated into the next official release. With limited numbers of curators available, these databases find it difficult to keep up with the requests of many expert annotators.

To overcome the restrictions of archival databases and the bottlenecks of curatorial databases, a number of groups have attempted to develop third party annotation systems. Examples include the Worm Community System [7], the Genome Sequence Database [8], and GDB [9, 10]. These systems typically require global coordination by either keeping all annotations in a centralized open repository or by forcing all parties to adhere to a common database format or by requiring a controlled vocabulary.

Another recent experiment with third party annotation has been the “annotation party,” exemplified by Celera’s Fly Jamboree and the Human Genome Project Consortium’s Analysis Group (HGPCAG). Parties gather together a large number of experts to produce the best annotations possible in a limited time frame. However, it is not clear that the annotation party model is sustainable once the initial flush of enthusiasm has worn off.

The HGPCAG model has a notion of annotation “tracks”, where a track contains a particular kind of annotation produced by a particular participating group. For example, the Eddy lab

provides a noncoding RNA track that annotates the positions of RNA genes in the human genome. Annotation tracks are independent of each other and therefore easy to integrate into a single display. The concept is essentially identical to the independent columns of annotation displayed by an ACeDB browser, except that the tracks in the HGPCAG annotation are curated by a variety of groups at different institutions, as opposed to a centralized curation group. However, the data for every track are still kept on a single centralized server; updating an annotation track after it has been submitted is cumbersome.

Here we introduce a genome annotation strategy that enables third-party annotation in a way that allows annotators to control and update their work, and which does not require much centralized coordination. The Distributed Annotation System (DAS) was designed as a lightweight system for integrating data from a number of heterogeneous distributed databases. The DAS system has a notion of annotation “layers”, which are essentially identical to tracks, except that now the data for each layer are on “third party servers” that are controlled by each annotation provider. The key idea was to produce a data exchange standard (the DAS XML specification) that enables layers to be provided in real time from 3rd party servers and overlaid to produce a single integrated view by a DAS client.

Figure 1 shows a cartoon example of the DAS paradigm. The client selects a single reference genome server and any number of annotation servers. The display layers the data returned from each server. A particular annotation can then be queried to retrieve more information from its providing server, as HTML pages.

## Implementation

The basic system is composed of a genome server, one or more annotation servers, and an annotation viewer. The genome server is responsible for serving genome maps, sequences, and information related to the sequencing process. Annotation servers are responsible for responding to requests on a region and delivering annotations. The client, an annotation viewer, is a lightweight application whose behavior is analogous to a web browser. The viewer communicates with the genome and annotation servers using a well defined language specification.

At a fundamental level, all annotations can be reduced to their coordinates relative to a particular sequence landmark. The DAS viewer retrieves annotations from the various annotation servers and uses the sequence coordinates to generate an integrated index of what is on the genome. This integration is then presented to the user in tabular or graphical form. Annotation providers can provide a suggestion of how their annotations should be rendered in a graphical display, and can provide links back to their databases and web sites to allow the researcher to retrieve further information about the annotation.

Because it relies entirely on sequence coordinates to achieve integration, DAS does not attempt to resolve semantic contradictions between different data sources. The goal of the system is to provide indexing and visualization, thereby making contradictions between annotations visible.

## Reference Sequence

The distributed annotation system relies on there being a common “reference sequence” on which to base annotations. The reference server consists of a set of “entry points” into the sequence, and the lengths of each entry point. Entry points will vary from genome to genome. For some genome projects, entry points correspond to entire chromosomes. For others, entry points may be a series of contigs.

The entry points describe the top level items on the reference sequence map. It is possible for each entry point to have substructure, basically a series of subsequences (components) and their start and end points. This structure is recursive. Annotations take the form of a statement about a region of the reference sequence. Each annotation is unambiguously located by providing its position as the start and stop positions relative to a “reference sequence.”

To give a concrete example, the *C. elegans* reference map consists of six top level entry points, one per chromosome. Each chromosome is formed from several contigs called “superlinks,” and each superlink contains one or more smaller contigs called “links.” Links in turn are composed of one or more fully-sequenced clones [11]. One could refer to an annotation by specifying its start or stop positions in clone, link, superlink, or chromosome coordinates.

The reference sequence server is responsible for providing the reference sequence map and the underlying DNA. The server can provide a list of sequence entry points or given a component of the map it can return its parent and children components. The reference server can provide arbitrarily long stretches of raw DNA sequence given a reference subsequence, start position, and stop position. Needless to say, bandwidth becomes a limiting factor for retrieving multi-megabase segments of DNA. However, in practice it is rare for users to retrieve more than a gene’s worth of raw DNA at a time.

## Annotation Servers

Annotation servers are specialized for returning lists of annotations across defined regions of the genome. Each annotation is anchored to the genome map by way of a start and stop position relative to one of the entry points. Annotations have an identifier that is unique to the providing server and a structured description of its nature and attributes. The general description of an annotation follows loosely the general feature format (GFF) which intentionally aims for a basic lowest common denominator description (<http://www.sanger.ac.uk/Software/formats/GFF/>). Annotations may also be associated with URLs where additional human or machine readable information about the annotation can be found.

The annotator is free to describe his annotations using any terms which he feels are appropriate, as DAS does not impose a controlled vocabulary. Annotations have *categories*, *types*, and *methods* defined by the annotator. The annotation **type** corresponds to a biologically significance description. In the Eddy Lab RNA track of the HGP three types are defined, “tRNA”, “snoRNA”, and “miscRNA”. The annotation **method** is intended to describe how the annotated feature was discovered, and may include a reference to a software program. The annotation **category** is a broad functional category. “Homology”, “variation” and “transcribed” are example

categories. This structure allows researchers to add new annotation types if the existing list is inadequate without entirely losing all semantic value. It is intended that larger annotation servers provide URLs to human-readable information that describes its types, methods and categories in more detail.

Another optional feature of annotation servers is the ability to provide hints to clients on how the annotations should be rendered visually. This is done by returning a DAS “stylesheet.” Stylesheets use the **type** and **category** information to associate each annotation with a particular graphical representation, a glyph.

Although the servers are conceptually divided between reference servers and annotation servers, there is in fact no key difference between them. A single server can provide both reference sequence information and annotation information. The main functional difference is that the reference sequence server is required to serve the coordinate map and the raw DNA, while annotation servers have no such requirement.

## Specification

The main component of DAS is the XML specification, which defines all valid DAS communication. As with HTML, our goal is a language which is human readable, easily parsed, and extensible. The additional file [appendix.pdf] provides a summary of version 1.01 of the DAS specification.

While a client can query multiple servers simultaneously, the communication between the client and any single server follows a simple client server model. Clients query the reference and annotation servers by sending a formatted URL request to each server. Each URL has a site-specific prefix, followed by a standardized path and query string. The standardized path begins with the string **/das**. This is followed by URL components containing the data source name and a command. For example:

<http://stein.cshl.org/das/elegans/features?segment=ZK154:1000,2000>

In this case, the site-specific prefix is **http://stein.cshl.org/**. The request begins with the standardized path **/das**, and the data source, in this case **/elegans**. This is followed by the command **/features**, which requests a list of features relative to a given set of named arguments (*?segment=ZK154:1000,2000*). The data source component allows a single server to provide information on several genomes.

Servers process the request and return a response as defined by the DAS specification, typically a formatted XML document. The response from the server to the client consists of a standard HTTP header with DAS status information within that header followed optionally by an XML file that contains the answer to the query. The DAS status portion of the header consists of two lines. The first is X-DAS-Version and gives the current protocol version number, currently DAS/1.0. The second line is X-DAS-Status and contains a three digit status code which indicates the outcome of the request. The defined status codes are listed in Table 1.

An example HTTP header: (*provided by server*)

```
HTTP/1.1 200 OK
Date: Sun, 12 Mar 2000 16:13:51 GMT
Server: Apache/1.3.6 (Unix) mod_perl/1.19
Last-Modified: Fri, 18 Feb 2000 20:57:52 GMT
Connection: close
Content-Type: text/plain
X-DAS-Version: DAS/1.0
X-DAS-Status: 200
DATA FOLLOWS ...
```

The specification outlines seven basic queries which a client can use to interrogate a DAS server. The valid queries are briefly summarized in Table 2. Two queries, “dsn” and “entry\_points”, essentially provide information to the client about the structure of the server and the reference sequence. The “dna” query can be used to fetch a segment of DNA from a reference server. A client can request annotations, “features”, or a summary of the annotations available, “types”, from any DAS server. The main annotation content query, “features”, basically follows the general feature format (GFF). The servers provide a “stylesheet” to suggest representations to the client’s graphical display. When more information is desired about a particular annotation, the client makes a “link” request. The “link” request, the only query which does not return a structured XML document, returns HTML. It is anticipated that DAS clients will hand off the link requests to the local web browser or other web-accessible genome database.

## Prototypes

A series of prototypes for both the client and server components were developed to test various versions of the DAS specification.

## Servers

A server is expected to respond to the DAS specification’s defined queries with the appropriate content, usually XML. The details of server implementation are left to the various annotation source providers. We provide a sample Perl script for converting ACeDB-based databases into DAS servers, and the Dazzle Java library does the same thing for annotation databases based on the Ensembl code base (T. Down, personal communication, 2001).

The first reference DAS server was written for WormBase [11] and piggybacks on the WormBase software architecture: an Apache/mod\_perl web server communicating with an ACeDB database via the AcePerl database access library. The Perl DAS server accepts incoming DAS requests, translates them into the ACeDB query language, reformats the results as XML, and returns them. The WormBase DAS server is currently serving as the *C. elegans* reference server at <http://www.wormbase.org/das/>. A set of servers containing test data, one reference and four annotation, are available at <http://skynet.wustl.edu/cgi-bin/das/>.

## Viewers

We have developed two prototype DAS client programs. One, called Geodesic, is a stand alone Java application. It connects to one or more DAS servers, retrieves annotations, and displays them in an integrated map, as seen in Figure 2. The other, called DasView, is a Perl application that runs as a server-side script. It connects to one or more DAS servers, constructs an integrated image, and serves the image to a web browser as a set of click-able image map, as seen in Figure 3.

Geodesic is mouse and menu driven. The user can choose which data sources to display. The user identifies a segment of the genome to view by browsing through entry points or entering a region name directly. By clicking on a feature, the user obtains additional information in the Feature Details tab and can optionally follow available links back to the original data source. The user can save displayed data as FASTA, GFF, or DAS XML. The user can, to a limited extent, customize the display within the preferences menu.

The DasView prototype implements an alternative mode of using DAS, browserless server side integration. A database can hook into trusted third party servers behind the scenes. The third party data are then integrated into the normal data displays of the database. In this scenario, no DAS client software would be needed.

Both viewers provide the user with one-click linking back the primary data sources where they can learn more about a selected annotation, and are sufficiently flexible to accept a wide range of annotation types and visualization styles. The stand alone Java viewer is appropriate for extensive, long-term use. The Perl implementation is suitable for casual use because it does not require the user to preinstall the software.

## Discussion

DAS distributes data sources across the Internet improving scalability over monolithic systems. This distribution of data encourages a divide-and-conquer approach to annotation, where experts provide and maintain their own annotations. It also permits annotation providers to disagree about a particular region, encouraging informative dissension and dialogue. The separation of sequence and map information from annotation allows them to be stored and represented in a variety of database schema. A number of different database backend alternatives could arise.

The use of links as a method of referencing back to the data provider's web pages provides even greater power of expression and content control. Annotation providers can make available complex query mechanisms for fine access to more information about the data provided to DAS. Alternatively they can link directly to webpages.

DAS does not enforce third party annotations to be peer reviewed. A strict requirement of peer review would block data sharing activities between collaborating labs. However, nothing prevents DAS layers from being "blessed" by a data provider, peer reviewer, or by both.

We made a design decision to use an XML-based format. This gives us a strongly typed, extensible data exchange format, but at the cost of non-trivial bandwidth demands. Bandwidth requirements are a substantial concern in the continued design and development of DAS. A user browsing a large genome can easily request more information than their network connection can reasonably handle. The DAS spec attempts to minimize bandwidth demands by representing each annotation with the minimal set of attributes needed for integration. Further bandwidth reductions will be useful, and the extreme redundancy of XML suggests that compression methods are a natural way forward. The HTTP protocol allows web clients to request byte-level compression of the response by sending the HTTP header “accept-encoding”. Web servers can reply with a “content-transfer-encoding” header and a compressed body. The Dazzle server and Bio::Das client have already utilized this feature to reduce their bandwidth requirements. Other compression schema are possible including DAS specific approaches that take advantage of the structure of DAS data.

The World Wide Web Consortium has developed a number of technologies to support XML based systems. A number of these technologies should be considered for future integration into DAS. The Simple Object Access Protocol (SOAP) 1.1 describes a lightweight protocol for the exchange of information in a decentralized, distributed environment. A DAS request may be replaced with a SOAP-style XML-encapsulated document in future versions of this specification. Each annotation is identified by its site-specific database identifier. The combination of this identifier with the server URL and data source produces a feature identifier which is globally unique. Future versions of DAS could utilize this identifier with XPATH and XLINK technologies to permit meta-annotations.

In large part, the continued success of this project will depend on the readiness with which the research community creates annotation sources. To facilitate this, we are working with the BioPerl and BioJava software developer communities (<http://open-bio.org/>) to develop a core set of servers, clients and software modules to support DAS. It is particularly important that the general biological community should be enabled to develop their own DAS annotation servers, without learning XML and Web software development. Easy, well-documented DAS annotation servers that take input data in simple flat file formats and convert it automatically to DAS XML are currently under development.

The DAS specification is under continued development. It does not detail how data source URLs will be publicized. It is anticipated that word of mouth and publications will be the driving forces in user selection. In addition, search engines can be developed to work with the DAS specification.

## Conclusions

The DAS specification is already being used in real-world applications. The July 9 2001 release of the Ensembl database of human genome annotations contains support for DAS, including an integrated DAS viewer and multiple annotation servers (M. Pocock, personal communication, 2001). The WormBase DAS server has recently been supplemented by a third

party annotation source of cDNA alignments contributed by The Institute for Genome Research, and a prototype DAS reference server for the *Drosophila* genome is also available, courtesy of the Berkeley *Drosophila* Genome Project (B. Marshall, personal communication, 2001). Table 3 lists the URLs where one can learn more about the current state of the art in DAS implementations.

## **Acknowledgements**

The initial ideas for DAS were developed in conversations with LaDeana Hillier of the Washington University Genome Sequencing Center. This work was primarily supported by NIH National Human Genome Research Institute (NHGRI) grant 2-P01-HG00956 for the *Caenorhabditis elegans* genome project, and by a Howard Hughes Medical Institute (HHMI) Predoctoral Fellowship to RDD. We also gratefully acknowledge additional funding support from HHMI and the NIH NHGRI.

## References

- [1] TF Smith: **The history of genetic sequence databases.** *Genomics* 1990, **6**: 701–707.
- [2] A Bairoch, R Apweiler: **The SWISS-PROT protein sequence data bank and its supplement TrEMBL in 1999.** *Nucleic Acids Research* 1999, **27**: 49–54.
- [3] KD Pruitt, KS Katz, H Sicotte, DR Maglott: **Introducing RefSeq and LocusLink: curated human genome resources at the NCBI.** *Trends Genet* 2000, **16**: 44–47.
- [4] MC Costanzo, ME Crawford, JE Hirschman, JE Kranz, P Olsen, LS Robertson, MS Skrzypek, BR Braun, KL Hopkins, P Kondu, *et al.*: **YPD, PombePD and WormPD: model organism volumes of the BioKnowledge library, an integrated resource for protein information.** *Nucleic Acids Res* 2001, **29**: 75–9.
- [5] FH Eeckman, R Durbin: **ACeDB and macace.** *Methods Cell Biol* 1995, **48**: 583–605.
- [6] R Waterson, J Sulston: **The genome of the *Caenorhabditis elegans*.** *Proc. Natl. Acad. Sci.* 1995, **92**: 10836–10840.
- [7] LM Shoman, E Grossman, K Powell, C Jamison, BR Schatz: **The Worm Community System, release 2.0 (WCSr2).** *Methods Cell Biol* 1995, **4**: 607–25.
- [8] MP Skupski, M Booker, A Farmer, M Harpold, W Huang, J Inman, D Kiphart, S Root, F Schilkey, J Schwertfeger, *et al.*: **The Genome Sequence DataBase: towards an integrated functional genomics resource.** *Nucleic Acids Res* 1999, **27**: 35–8.
- [9] SI Letovsky, RW Cottingham, CJ Porter, PW Li: **GDB: the human genome database.** *Nucleic Acids Res* 1998, **26**: 94–9.
- [10] AJ Cuticchia: **Future vision of the GDB human genome database.** *Hum Mutat* 2000, **15**: 62–67.
- [11] L Stein, P Sternberg, R Durbin, J Thierry Mieg, J Spieth: **WormBase: network access to the genome and biology of *Caenorhabditis elegans*.** *Nucleic Acids Res* 2001, **29**: 82–86.

## Figures

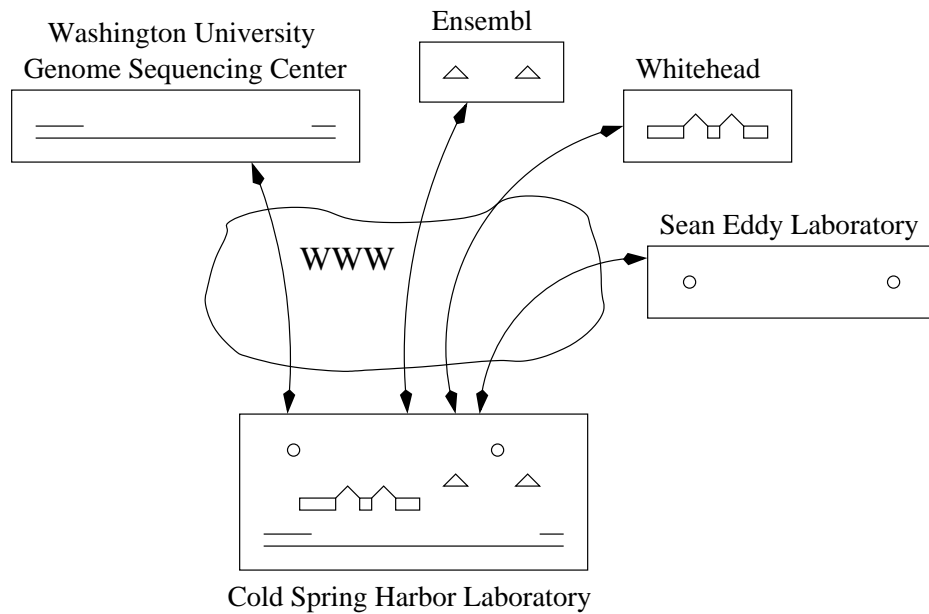


Figure 1: **Basic distributed annotation system architecture**

One server is the designated reference server, in this case the Washington University Genome Sequencing Center. One or more annotation servers, shown above as Ensembl, Whitehead, and the Sean Eddy Laboratory, provide annotations relative to the reference sequence. The client, at Cold Spring Harbor Laboratory in our example, fetches data from multiple servers and automatically generates an integrated view.

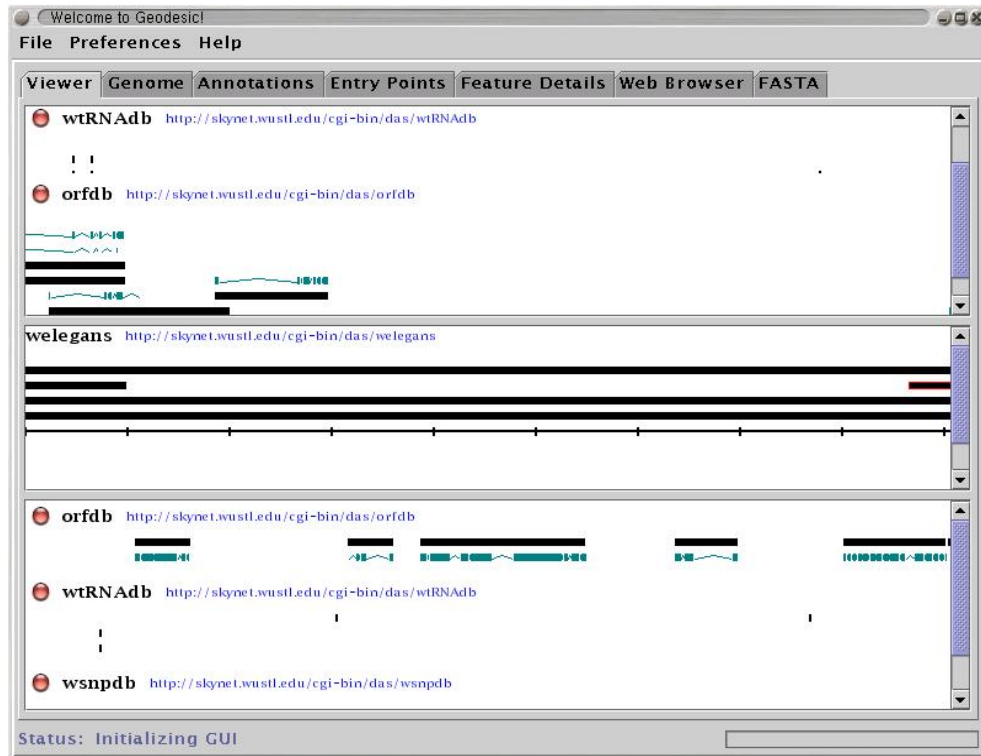


Figure 2: **Geodesic**

A screen-shot of the current version of Geodesic. The view is on clone ZK154 using sources from the *C. elegans* test server set.

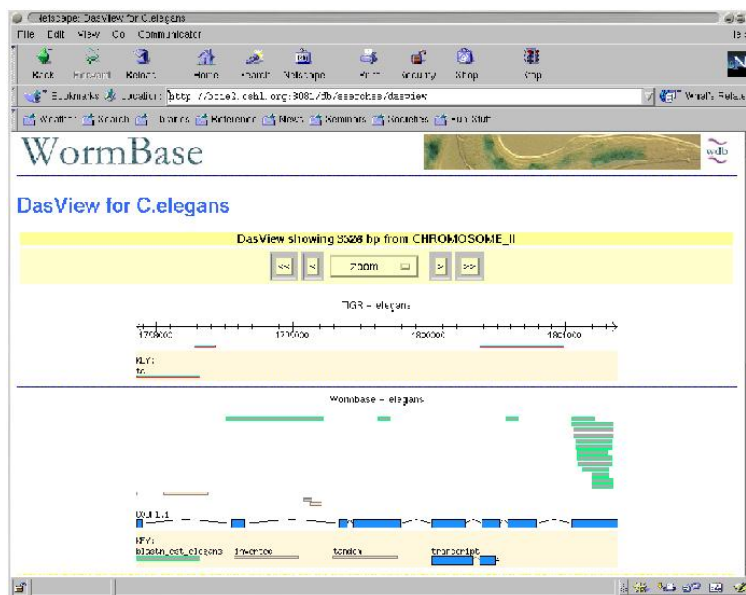


Figure 3: **DasView**

A screen-shot of the current version of DasView. The view is on Chromosome II of WormBase.

## Tables

Code	Meaning
200	OK, data follows
400	Bad command (command not recognized)
401	Bad data source (data source unknown)
402	Bad command arguments (arguments invalid)
403	Bad reference object (reference sequence unknown)
404	Bad stylesheet (requested stylesheet unknown)
405	Coordinate error (out of bounds/invalid)
500	Server error, not otherwise specified
501	Unimplemented feature

Table 1: **Server Status Codes**

Server status codes are modeled after the familiar status codes of the HTTP 1.0 protocol.

Command	Basic Format	Scope
dsn	PREFIX/das/dsn	both
entry_points	PREFIX/das/DSN/entry_points	reference
dna	PREFIX/das/DSN/dna?segment=SEG	reference
types	PREFIX/das/DSN/types?segment=SEG	both
features	PREFIX/das/DSN/features?segment=SEG	both
stylesheet	PREFIX/das/DSN/stylesheet	both
link	PREFIX/das/DSN/link?field=TAG;id=ID	both

Table 2: **Queries Summary**The basic seven queries of the DAS 1.01 specification.

Site	URL
DAS project website	<a href="http://www.biodas.org/">http://www.biodas.org/</a>
Current specification	<a href="http://www.biodas.org/documents/spec.html">http://www.biodas.org/documents/spec.html</a>
Wormbase reference server	<a href="http://www.wormbase.org/db/das/">http://www.wormbase.org/db/das/</a>
Dazzle Java Library	<a href="http://www.biojava.org/dazzle/">http://www.biojava.org/dazzle/</a>
Test server cluster	<a href="http://skynet.wustl.edu/cgi-bin/das/">http://skynet.wustl.edu/cgi-bin/das/</a>
Geodesic	<a href="http://www.biodas.org/geodesic/">http://www.biodas.org/geodesic/</a>
Ensembl DAS	<a href="http://www.ensembl.org/das/">http://www.ensembl.org/das/</a>
Drosophila DAS	<a href="http://www.fruitfly.org/cgi-bin/das/">http://www.fruitfly.org/cgi-bin/das/</a>

Table 3: **Summary of DAS URLs**

For the latest information on the DAS project, see the project website. To learn more about one of the prototype components of DAS, see the appropriate website.

# The DAS XML Specification

The following is a summary of the current DAS specification, It is based on version 1.01 (October 2001) of the specification. The specification was originally written by Lincoln D. Stein, Sean R. Eddy, and Robin D. Dowell. The current version of the specification is available at <http://www.biodas.org/documents/spec.html>.

## Reference Sequence IDs

Reference sequence IDs indicate a segment of the genome. They can correspond to low-level primary sequences such as sequenced clones, or to higher-level assemblies such as contigs.

A reference ID can contain any set of printable characters (including the space character), but not the colon character (“:”), which is reserved for separating reference IDs from sequence ranges (see below). The newline, tab and carriage return characters are also reserved for future use.

A data source that uses the colon character for its internal IDs must map this character to another one on the way out and on the way in. For example:

Client's request	Server's internal id	Response to client
gi-123456	gi:123456	gi-123456
gi-123456:1,1000	gi:123456 start=1 stop=1000	gi-123456:1,1000

## The Queries

This section lists the queries recognized by sequence and annotation servers. Each of these queries begins with some site-specific prefix, denoted here as *PREFIX*. The other meta-variable used in these examples is *DSN*, which is a symbolic data source name.

**Data Sources** The **dsn** query returns a list of data sources available from the server. A single annotation provider (unique *PREFIX*) may present a number of annotation databases by using different data source names.

**Scope:** Reference and annotation servers.

**Command:** *dsn*

**Format:** *PREFIX/das/dsn*

**Arguments:** *none*

**Return Document:**

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE DASDSN SYSTEM "http://biodas.org/dtd/dasdsn.dtd">
<DASDSN>
  <DSN>
    <SOURCE id="id1" version="version"> source name 1 </SOURCE>
```

```

    <MAPMASTER> URL </MAPMASTER>
    <DESCRIPTION> descriptive text 1 </DESCRIPTION>
</DSN>
<DSN>
    <SOURCE id="id2" version="version"> source name 2 </SOURCE>
    <MAPMASTER> URL </MAPMASTER>
    <DESCRIPTION href="url"> descriptive text 2 </DESCRIPTION>
</DSN>
. . .
</DASDSN>

```

**<!DOCTYPE>** (required; one only) The doctype indicates which formal DTD specification to use. For the dsn query, the doctype DTD is “dasdsn.dtd”.

**<DASDSN>** (required; one only) The appropriate doctype and root tag is DASDSN.

**<DSN>** (required; one or more) There are one or more <DSN> tags, one for each data source. Each <DSN> contains one <SOURCE> tag, one <MAPMASTER> tag, and optionally one <DESCRIPTION> tag.

**<SOURCE>** (required; one per DSN tag) This tag indicates the symbolic name for a data source. The symbolic name to use for further requests can be found in the **id** (required) attribute. A source **version** attribute is optional, but strongly recommended. The tag body contains a human-readable label which may or may not be different from the ID.

**<MAPMASTER>** (required; one per DSN tag) This tag contains the URL (http://site.specific.prefix/das/data\_src) that is being annotated by this data source. For an annotation server, this is the reference server which is being annotated. For a reference server this would echo its own URL.

**<DESCRIPTION>** (optional) This tag contains additional descriptive information about the data source. If an **href** (optional) attribute is present, the attribute contains a link to further human-readable information about the data source, such as its home page.

**Entry Points** The **entry\_points** query returns the list of sequence entry points available and their sizes in base pairs.

**Scope:** Reference servers.

**Command:** *entry\_points*

**Format:** PREFIX/das/DSN/entry\_points

The **entry\_points** command is not intended for transmitting sequence assembly, but rather to provide the top level entry points for browsers. See fetching the sequence assembly for the mechanism for retrieving assembly information.

**Return Document:**

```
<?xml version="1.0" standalone="no"?>
```

```

<!DOCTYPE DASEP SYSTEM "http://biodas.org/dtd/dasep.dtd">
<DASEP>
  <ENTRY_POINTS href="url" version="X.XX">
    <SEGMENT id="id1" size="length1"> descriptive text </SEGMENT>
    <SEGMENT id="id2" size="length2"> descriptive text </SEGMENT>
    <SEGMENT id="id3" size="length3"> descriptive text </SEGMENT>
    . . .
  </ENTRY_POINTS>
</DASEP>

```

**<!DOCTYPE> (required; one only)** The doctype indicates which formal DTD specification to use. For the entry\_points query, the doctype DTD is “dasep.dtd”.

**<DASEP> (required, one only)** The appropriate doctype and root tag is DASEP.

**<ENTRY\_POINTS> (required, only one)** There is a single <ENTRY\_POINTS> tag. It has a **version** number (required) in the form “N.NN”. Whenever the sequence map changes, the version number should change as well. The **href** (required) attribute echoes the URL query that was used to fetch the current document.

**<SEGMENT> (optional; zero or more)** Each segment contains the attributes **id** and **size** (both required). The id is a unique identifier, which can be used as the reference ID in further requests to DAS. The size is the full length of the segment object. The body of the <SEGMENT> sections contains human-readable text (optional) for the purposes of display and selection.

**DNA** The **dna** query returns the DNA corresponding to the indicated segment.

**Scope:** Reference servers.

**Command:** *dna*

**Format:** PREFIX/das/DSN/dna?segment=RANGE            [;segment=RANGE...]

**Arguments:**

**segment (required; one or more)** A reference sequence and optional range in the format: **ref:start,stop** or simply **ref**. Multiple segments are allowed. Here is an example of a request to fetch three non-overlapping segments:

dna?segment=chr1:1,1000;segment=chr1:5000,5200;segment=AC12345

**Return Document:**

```

<?xml version="1.0" standalone="no"?>
<!DOCTYPE DASDNA SYSTEM "http://biodas.org/dtd/dasdna.dtd">
<DASDNA>
  <SEQUENCE id="id" start="start" stop="stop" version="X.XX">
    <DNA length="NNNN">
      atttccttggcgtaaataagagtctcaatgagactctcagaagaaaattgataaatattat

```

```

    taatgatataataataatcttggtgatccgttctatctccagacgattttcctagtctcc
    ...
    gaaacaatgaacactattataacattttcagaaaatgtagtatttaagcgaaggtagtgc
    acatcaaggccgtcaaacggaaaaattttgcaagaatca
    </DNA>
  </SEQUENCE>
</DASDNA>

```

**<!DOCTYPE> (required; one only)** The doctype indicates which formal DTD specification to use. For the dna query, the doctype DTD is “dasdna.dtd”.

**<DASDNA> (required; one only)** The appropriate doctype and root tag is DASDNA.

**<SEQUENCE> (required; one or more)** There is one <SEQUENCES> tag per requested segment. It has the attributes **id**, which indicates the reference ID for this sequence, **start** and **stop**, which indicate the position of this segment within the reference sequence, and **version**, which provides the sequence map version number. All four attributes are required.

**<DNA> (required; one per SEQUENCE)** This tag surrounds the DNA data. It has the attribute **length** (required), which indicates the length of the DNA. The DNA is found in the body of the tag and is required. DNA will be lower-case and adhere to the IUPAC code conventions.

**Summary Information** The **types** query returns a summary of the annotation available for a segment of sequence.

**Scope:** Reference and annotation servers

**Command:** *types*

**Format:** PREFIX/das/DSN/types?segment=RANGE  
 [;segment=RANGE...] [;type=TYPEPATTERN]

**Arguments:**

**segment (required; one or more)** A reference sequence and optional range in the format: **ref:start,stop** or simply **ref**. Multiple segments are allowed.

**type (optional)** One or more type IDs to be used for filtering annotations on the type field. If multiple type names are provided, the resulting list of features will be the logical OR of the list.

If one or more segment arguments are provided, the list of types returned is restricted to the indicated segments. If no segment argument is provided, then all feature types known to the source are returned.

**Return Document:**

```

<?xml version="1.0" standalone="no"?>
<!DOCTYPE DASTYPES SYSTEM "http://biодas.org/dtd/dastypes.dtd">
<DASTYPES>
  <GFF version="1.0" href="url">

```

```

<SEGMENT id="id" start="start" stop="stop" version="X.XX"
          label="label">
  <TYPE id="id1" method="method" category="category">
    Type Count 1 </TYPE>
  <TYPE id="id2" method="method" category="category">
    Type Count 2 </TYPE>
  . . .
</SEGMENT>
</GFF>
</DASTYPES>

```

**<!DOCTYPE> (required; one only)** The doctype indicates which formal DTD specification to use. For the types query, the doctype DTD is “dastypes.dtd”.

**<DASTYPES> (required; one only)** The appropriate doctype and root tag is DASTYPES.

**<GFF> (required; one only)** There is a single <GFF> tag. Its **version** (required) attribute indicates the current version of the XML form of the General Feature Format. The current version is 1.0. The **href** (required) attribute echoes the URL query that was used to fetch the current document.

**<SEGMENT> (required; one or more)** The <SEGMENT> tag, provides information on the reference segment . The **id**, **start** and **stop** attributes indicate the coordinate system of the segment. The **version** attribute indicates the current version of the sequence map. The id, start, stop, and version attributes are required. The optional **label** attribute supplies a human readable label for display purposes.

**<TYPE> (required; one or more per SEGMENT)** Each segment has one or more <TYPE> tags, which summarize the types of annotation available. The attributes are **id** (required), which is a unique id for the annotation type and can be used to retrieve further information from the annotation server (see Linking to a Feature), **method** (optional) which indicates the method (subtype) for a feature type and the **category** (optional) attribute, which provides functional grouping to related types. The tag contents (optional) is a count of the number of features of this type across the segment.

**Annotations** The **features** query returns the annotations across a segment of sequence.

**Scope:** Reference and annotation servers

**Command:** *features*

**Format:** PREFIX/das/DSN/features?segment=RANGE

    [;segment=RANGE...][;type=TYPEPATTERN]

    [;category=CATEGORYPATTERN]

**Arguments:**

**segment (required; one or more)** A reference sequence and optional range in the format: **ref:start,stop** or simply **ref**. Multiple segments are allowed.

**type (zero or more)** Zero or more type IDs to be used for filtering annotations on the type field. If multiple type names are provided, the resulting list of features will be the logical OR of the list.

**category (zero or more)** Zero or more category IDs to be used for filtering annotations by category. If multiple categories are provided, they are treated as the logical OR.

**categorize (optional)** Either "yes" or "no" (default). If "yes", then each annotation must include its functional category.

Annotation servers are only required to return annotations which are completely contained within the indicated segment. Servers may also return annotations which overlap the segment, but are not completely contained within them. Annotations must be returned using the coordinate system in which they were requested. For example, if a contig ID was used to specify the segment, then the annotation endpoints must use contig coordinates.

If multiple segment arguments are provided and they happen to overlap, then the annotation server may return the same annotation multiple times, possibly using different coordinate systems. It is the responsibility of the client to merge annotations based on the assembly.

### **Return Document:**

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE DASGFF SYSTEM "http://biodas.org/dtd/dasgff.dtd">
<DASGFF>
  <GFF version="1.0" href="url">
    <SEGMENT id="id" start="start" stop="stop" version="X.XX"
      label="label">
      <FEATURE id="id" label="label">
        <TYPE id="id" category="category" reference="yes|no"
          subparts="yes|no" superparts="yes|no"> label </TYPE>
        <METHOD id="id"> method label </TYPE>
        <START> start </START>
        <END> end </END>
        <SCORE> [X.XX|-] </SCORE>
        <ORIENTATION> [0|-|+] </ORIENTATION>
        <PHASE> [0|1|2|-] </PHASE>
        <NOTE> note text </NOTE>
        <LINK href="url"> link text </LINK>
        <TARGET id="id" start="x" stop="y"> target name </TARGET>
        <GROUP id="hash" label="label" type="type">
          <NOTE> note text </NOTE>
          <LINK href="url"> link text </LINK>
        </GROUP>
      </FEATURE>
      ...
    </SEGMENT>
  </GFF>
</DASGFF>
```

The positions of all returned annotations are given relative to the indicated reference

sequence.

**<!DOCTYPE>** (required; one only) The doctype indicates which formal DTD specification to use. For the feature query, the doctype DTD is “dasgff.dtd”.

**<DASGFF>** (required; one only) The appropriate doctype and root tag is DASGFF.

**<GFF>** (required; one only) There is a single <GFF> tag. Its **version** (required) attribute indicates the current version of the XML form of the General Feature Format. The current version is (arbitrarily) 1.0. The **href** (required) attribute echoes the URL query that was used to fetch the current document.

**<SEGMENT>** (required; one or more) The <SEGMENT> tag, provides information on the reference segment coordinate system. The **id**, **start** and **stop** attributes indicate the position of the segment. The **version** attribute indicates the current version of the sequence map. The **id**, **start**, **stop**, and **version** attributes are required. The optional **label** attribute provides a human readable label for display purposes.

**<FEATURE>** (optional; zero or more per SEGMENT) Each <FEATURE> tag provides information on one annotation. The **id** attribute (required) is a unique identifier for the feature. It can be used as a reference point for further navigation. The **label** attribute (optional) is a suggested label to display for the feature. If not present, the **id** attribute can be used instead.

**<TYPE>** (required; one per FEATURE) Each feature has just one <TYPE> field, which indicates the type of the annotation. The attributes are **id** (optional), which is a unique id for the annotation type and can be used to retrieve further information from the annotation server (see Linking to a Feature), and the **category** (optional) attribute, which provides functional grouping to related types. The reference server’s annotations can consist of additional overlapping landmarks (parents, children, and neighbors), which should be marked “yes” in the third attribute **reference** (optional, defaults to “no”) to indicate that the feature is a structural landmark within the map (this feature can be annotated). The tag contents (optional) is a human readable label for display purposes. If a **reference** annotation has either or both of the optional attributes **subparts = “yes”** and **superparts = “yes”** then in addition to being useable as a reference sequence, the annotation contains subparts that themselves can act as reference features. This can be used to reconstruct reference server’s assembly (see the Entry Points document).

**<METHOD>** (required; one per FEATURE) Each feature has one <METHOD> field, which gives the method used to identify the feature. The **id** (optional) tag can be used to retrieve further information from the server. The tag contents (optional) is a human readable label.

**<START>**, **<END>** (required; one apiece per FEATURE) These tags indicate the start and end of the feature in the coordinate system of the reference sequence given in the <SEGMENT> tag. The relationship between the feature start and stop positions and the segment start and stop is that the two spans are guaranteed to overlap.

**<SCORE>** (required; one per FEATURE) This is a floating point number indicating the “score” of the method used to find the current feature. The number can only be understood in the context of information retrieved from the server by linking to the method. If this field is inapplicable, the contents of the tag can be replaced with a “-” symbol.

- <**ORIENTATION**> (**required; one per FEATURE**) This tag indicates the orientation of the feature relative to the direction of transcription. It may be **0** for features that are unrelated to transcription, **+**, for features that are on the sense strand, and **-**, for features on the antisense strand.
- <**PHASE**> (**required; one per FEATURE**) This tag indicates the position of the feature relative to open reading frame, if any. It may be one of the integers **0**, **1** or **2**, corresponding to each of the three reading frames, or **-** if the feature is unrelated to a reading frame.
- <**NOTE**> (**optional; zero or more per FEATURE**) A human-readable note in plain text format.
- <**LINK**> (**optional; zero or more per FEATURE**) A link to a web page somewhere that provides more information about this feature. The **href** (required) attribute provides the URL target for the link. The link text is an optional human readable label for display purposes.
- <**TARGET**> (**optional; zero or more per FEATURE**) The target sequence in a sequence similarity match. The **id** attribute provides the reference ID for the target sequence, and the **start** and **stop** attributes indicate the segment that matched across the target sequence. All three attributes are required. More information on the target can be retrieved by linking back to the annotation server. See Linking to a Feature. (Earlier versions of this specification required the TARGET tag to be in the GROUP section.)
- <**GROUP**> (**optional; if present, one per FEATURE**) The <GROUP> section is an oddity, as it is derived from an overloaded field in the GFF flat file format. It provides a unique “group” ID that indicates when certain features are related to each other. The canonical example is the CDS, exons and introns of a transcribed gene, which logically belong together. The group **id** (required) tag provides an identifier that should be used by the client to group features together visually. Unlike other IDs in this protocol, the group ID cannot be used as a database handle to retrieve further information about the group. Such information can, however, be provided within <GROUP> section, which may contain up to three optional tags. The **label** attribute (optional) provides a human-readable string that can be used in graphical representations to label the glyph. The **type** attribute (optional) provides a type ID for the group as a whole, for example “transcript”. This ID can be used as a key into the stylesheet to select the glyph and graphical characteristics for the group as a whole.
- <**NOTE**> (**optional; if present, one per GROUP**) A human-readable note in plain text format.
- <**LINK**> (**optional; if present, one per GROUP**) A link to a web page somewhere that provides more information about this group. The **href** (required) attribute provides the URL target for the link. The link text is an optional human readable label for display purposes.

**Linking to a Feature** The **link** query can be issued in order to retrieve further human-readable information about an annotation. It is best to pass this URL directly to a browser, as the type of the returned data is not specified (it will typically be an HTML file, but any MIME format is allowed).

**Scope:** Annotation Servers

**Command:** *link*

**Format:** PREFIX/das/DSN/link?field=TAG;id=ID

**Arguments:**

**field (required)** The field to fetch further information on. Options are:

- **feature** – the feature itself
- **type** – the feature type
- **method** – the feature method
- **category** – the feature category
- **target** – the target, applicable to sequence similarities only

**id (required)** The ID of the indicated annotation field.

**Returns:** A web page.

**Stylesheet** The **stylesheet** query can be issued to an annotation server in order to retrieve the server's recommendations on formatting annotations retrieved from it. These recommendations are not normative. A viewer is free to use any display format it chooses.

**Scope:** Annotation Servers

**Command:** *stylesheet*

**Format:** PREFIX/das/DSN/stylesheet

**Arguments:** None.

**Return Document:**

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE DASSTYLE SYSTEM "http://biодas.org/dtd/dasstyle.dtd">
<DASSTYLE>
  <STYLESHEET version="X.XX">
    <CATEGORY id="default">
      <TYPE id="default">
        <GLYPH zoom="high"> <ID>
          <ATTR>value</ATTR>
          <ATTR>value</ATTR>
          ...
        </ID> </GLYPH>
      </TYPE>
    </CATEGORY>
    <CATEGORY id="category1">
      <TYPE id="default">
        <GLYPH zoom="medium"> <ID>
          <ATTR>value</ATTR>
        </ID> </GLYPH>
      </TYPE>
      <TYPE id="type1">
        <GLYPH zoom="low"> <ID>
          <ATTR>value</ATTR>
```

```

        ...
        </ID> </GLYPH>
    </TYPE>
    ...
</CATEGORY>
<CATEOGRY id="group">
    <TYPE id="group_id1">
        <GLYPH> <ID>
            <ATTR>value</ATTR>
        </ID> </GLYPH>
    </TYPE>
</CATEGORY>
<CATEGORY id="category2">
    ...
</CATEGORY>
    ...
</STYLESHEET>
</DASSTYLE>

```

**<!DOCTYPE> (required; one only)** The doctype indicates which formal DTD specification to use. For the stylesheet query, the doctype DTD is “dasstyle.dtd”.

**<DASSTYLE> (required; one only)** The appropriate doctype and root tag is DASSTYLE.

**<STYLESHEET> (required; one only)** There is a single <STYLESHEET> tag. Its **version** (required) attribute indicates the current version of the stylesheet, and can be used for caching purposes.

**<CATEGORY> (required; one or more)** There are one or more <CATEGORY> tags, each providing information on the display of a high-level feature category. The **id** (required) tag uniquely names the category. A special name is “default”, which tells the annotation viewer what format to use for categories that are not otherwise specified in the stylesheet. Another special id, “group”, identifies which style to use for a particular group of features.

**<TYPE> (required; one or more per CATEGORY)** There are one or more <TYPE> tags per <CATEGORY>, each providing display suggestions for one type of annotation. The **id** (required) uniquely identifies the type. A special id is “default”, which, if present, identifies a default style for the enclosing category.

**<GLYPH> (required; one per TYPE)** There is a single <GLYPH> tag per <TYPE>. It provides information on what glyph (graphical widget) to use to display the indicated annotation type. The optional **zoom** attribute implements a simple form of semantic zooming, and allows the client to select the glyph and its attributes based on the current zoom level. Possible values are “high”, “medium” or “low”. If multiple <GLYPH> tags are present, this attribute **must** be present in order to select among them. A “high” zoom means that there are fewer base pairs per pixel (high magnification), “low” zoom shows more base pairs, and “medium” is intermediate. It is left to the client to determine the boundaries for “high”, “medium” and “low” since it is a function of graphics rendering.

**<ID> (required; one per GLYPH)** The ID tag is a particular instance of one of the recognized glyph from the glyph types list. (for example <BOX> is an instance of <ID>)

**<ATTR> (optional; one or more per ID)** The recognized ATTR (attributes) are determined by which glyph ID is specified.

**Glyphs and Groups** Glyphs and their attributes are typically applied to individual features. However, they can be applied to entire groups as well (via the <GROUP> type attribute). In this case, the glyph will apply to the connecting regions between the components of the group.

For example, to indicate that the exons in a “transcript” group should be drawn with a yellow box, that the utrs should be drawn with a blue box, and that the connections between exons should be drawn with a hat-shaped line:

```
<CATEGORY id="Transcription">
  <TYPE id="exon">
    <GLYPH>
      <BOX>
        <BGCOLOR>yellow</BGCOLOR>
      </BOX>
    </GLYPH>
  </TYPE>

  <TYPE id="utr">
    <GLYPH>
      <BOX>
        <BGCOLOR>blue</BGCOLOR>
      </BOX>
    </GLYPH>
  </TYPE>
</CATEGORY>

<CATEGORY id="group">
<TYPE id="transcript">
  <GLYPH>
    <LINE>
      <STYLE>hat</STYLE>
      <FGCOLOR>black</FGCOLOR>
    </LINE>
  </GLYPH>
</TYPE>
...
```

## Fetching Sequence Assemblies

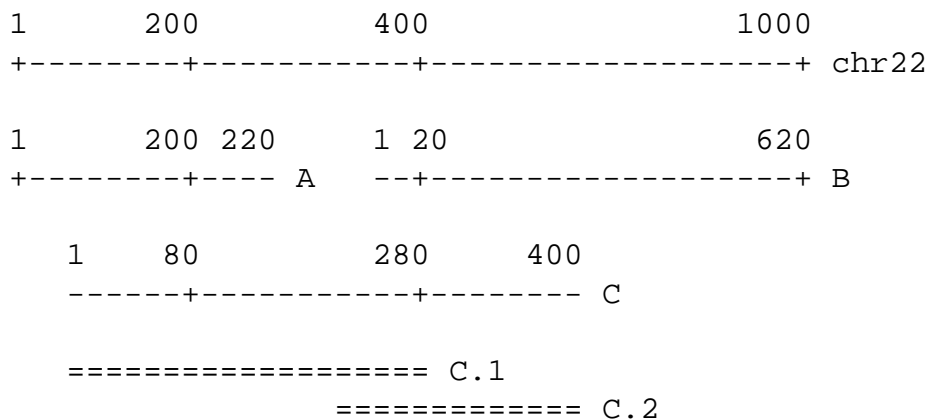
Reference servers, but not annotation servers, must represent and serve genome assemblies.

The components of an assembly are treated as a set of features with a type category attribute of "component" and a reference attribute of "yes". Intermediate components of the assembly will also have a subparts attribute of "yes" and/or *superparts* attribute of "yes". Components that are the parents of the reference sequence in the assembly have a category attribute of "supercomponent".

## Moving Downwards in an Assembly

For those components that have subparts, the start and end of the feature give the feature's position in the requested segment's coordinate system, and the start and end of the <TARGET> element gives the feature's position in its native coordinates.

For example:



A request for this assembly will look like the following:

<http://www.wormbase.org/db/das/elegans/features?segment=chr22:1,1000;category=component>

The reference server will return the following (abbreviated) document:

```
<SEGMENT id="chr22" start="1" stop="1000">

<FEATURE id="chr22">
  <START>1</START>
  <STOP>1000</STOP>
  <TYPE id="Contig" category="component" reference="yes"
        superparts="no" subparts="yes">chr 22</TYPE>
  <TARGET id="chr22" start="1" stop="1000">Contig A</TARGET>
  ...
</FEATURE>
```

```

<FEATURE id="Contig:A">
  <START>1</START>
  <STOP>200</STOP>
  <TYPE id="Contig" category="component" reference="yes"
        superparts="yes" subparts="no"> a contig</TYPE>
  <TARGET id="A" start="1" stop="200">Contig A</TARGET>
  ...
</FEATURE>
<FEATURE id="Contig:B">
  <START>400</START>
  <STOP>1000</STOP>
  <TYPE id="Contig" category="component" reference="yes"
        superparts="yes" subparts="no"> a contig</TYPE>
  <TARGET id="B" start="20" stop="620">Contig A</TARGET>
  ...
</FEATURE>
<FEATURE id="Contig:C">
  <START>200</START>
  <STOP>400</STOP>
  <TYPE id="Contig" category="component" reference="yes"
        superparts="yes" subparts="no"> a contig</TYPE>
  <TARGET id="C" start="80" stop="280">Contig A</TARGET>
  ...
</FEATURE>
</SEGMENT>

```

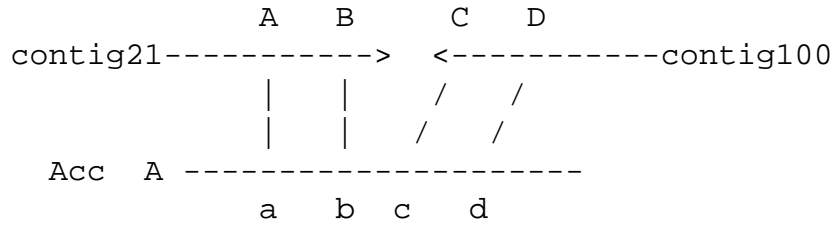
Notice that contig C is marked as having subparts. This is an indication to the client that it should emit a features request that includes segment C:80,280 in order to discover its components (C.1 and C.2).

Notice also that chr22 appears as a component of itself with the attribute **superparts="no"** and **subparts="yes"**. This is a side effect of providing information about the component parent.

## Moving Upwards in an Assembly

It is desirable for a client to fetch the parent of a segment, so as to accommodate the situation in which the user enters the browser at a contig or sequenced clone, and wants to “zoom out.”

This situation is complicated by rough draft issues, in which a single rough draft sequence segment may have multiple parents, and some sections of the segment may not belong in the assembly at all. For example:



Here, the segment “Acc A” contains two fragments, one of which is located on contig21 and the other on contig100.

To retrieve this information, the client requests the category supercomponent. For segments that are in the middle of the assembly, one or more assembly parents will be returned in addition to subcomponents. The parent <START>, <STOP> and <ORIENTATION> tags are presented in the coordinate system of the requested segment, as always. The start and stop attributes of the <TARGET> tag, denote the corresponding segment in the coordinate system of the parent. As always, start is less than stop, for both the feature and the target.

```
<SEGMENT id="Acc A" start="1" stop="1000">
  <FEATURE id="contig21_goldenpath_map">
    <START>a</START>
    <STOP>b</STOP>
    <ORIENTATION>+</ORIENTATION>
    <TYPE id="Contig" category="supercomponent" reference="yes"
      superparts="yes" subparts="yes">a contig</TYPE>
    <TARGET id="contig21" start="A" stop="B"></TARGET>
    ...
  </FEATURE>
  <FEATURE id="contig100_goldenpath_map">
    <START>c</START>
    <STOP>d</STOP>
    <ORIENTATION>-</ORIENTATION>
    <TYPE id="Contig" category="supercomponent" reference="yes"
      superparts="yes" subparts="yes">a contig</TYPE>
    <TARGET id="contig100" start="D" stop="C"></TARGET>
    ...
  </FEATURE>
</SEGMENT>
```

To continue following the parents upward in the assembly, the client will issue further features requests for the target IDs, in this case “contig21” and “contig100”. In the general case, following parents will project the requested segment onto a discontinuous set of regions, potentially on different chromosomes. The client may wish to alert the user and refuse to proceed further when it encounters a segment with multiple parents.

## Feature Types and Categories

Features are associated with a graphical representation, a glyph, by the type id and category. The “category” is designed to describe broad annotation types (i.e. transcribed). The “id” refers to a more specific instance within its category (i.e. exon). An annotation provider is free to use any categories and types that they feel are appropriate to their annotations. If an annotation is described by a type id and category which are not defined by the accompanying stylesheet, programmatic defaults are used. The following is a list of generic feature categories and specific feature types within them.

### Feature Types and Categories

- **component** – indicate that the feature is a child component of the reference sequence in the current assembly. When combined with **reference = “yes”** attribute this indicates that the feature can be used as a reference point to retrieve subfeatures contained within it (including subcomponents).
- **supercomponent** – indicate that the feature is the parent of the reference sequence in the current assembly. When combined with **reference = “yes”** attribute this indicates that the feature can be used as a reference point to retrieve features that completely contain the selected range of the reference sequence.
- **translation** – features that relate to regions of the sequence that are translated into proteins. (ex: stop, ATG, CDS, 5’UTR, 3’UTR, misc\_translated)
- **transcription** – features that relate to regions of the sequence that are transcribed into RNA. (ex: exon, intron, tRNA, mRNA, ncRNA, 5’Cap, PolyA, Splice5, Splice3, misc\_transcribed)
- **variation** – features that relate to regions of the sequence that are polymorphic (ex: insertion, deletion, substitution, misc\_variation)
- **structural** – features that relate to mapping, sequencing, and assembly, as well as various landmarks that carry no intrinsic biological information. (ex: clone, primer\_left, primer\_right, oligo, misc\_structural)
- **similarity** – areas of sequence that are similar to other sequences. Similarity features should have a <METHOD> tag that indicates the algorithm used for the sequence comparison, and a <TARGET> tag in the <GROUP> field that indicates the target of the match. (ex: NN (nucleotide to nucleotide), NP (nucleotide to peptide), PN (peptide to nucleotide), PP (peptide to peptide), misc\_homology)
- **repeat** – areas that contain repetitive DNA. This is used for both low complexity regions and for more biologically interesting features such as transposon insertion sites. (ex: microsatellite, inverted, tandem, transposable\_element, LINE, misc\_repeat)

- **experimental** – used to flag areas where there is interesting experimental data of one sort or another. (ex: knockout, expression\_tag, microarrayed, RNAi\_result, transgenic, mutant, misc\_experimental)

An annotation provider is free to use any category and type name. It is recommended, but not required, that the <GROUP> section contain <LINK> and/or <NOTE> tags that provide further information on the all features.

## Glyphs

This section defines a set of generic “glyphs” that can be used by sequence display programs to display the position of features on a sequence map. The annotation server may use these glyphs to send display suggestions to the viewer via the stylesheet document. Each glyph has a set of possible attributes associated with it. Attributes come in a few basic data types:

- INT (an integer)
- STRING (a text string)
- COLOR (specified as the #RRGGBB format commonly used for HTML or as one of the 16 IBM VGA colors)
- BOOL (either “yes” or “no”)
- FONT (any of the fonts recognized by Web browsers)
- FONT\_STYLE (“bold”, “italic”, or “underline”).
- LINE\_STYLE (“hat”, “solid”, “dashed”).

Some attributes are shared by all glyphs. Others are glyph-specific. The following are attributes shared in common:

- **BGCOLOR** – (*COLOR*) The background color of the glyph. For hollow glyphs, such as boxes, this is the color of the interior of the box. For solid glyphs, such as text, this is ignored.
- **BUMP** – (*BOOL*) Determines if the viewer should “bump” intersecting glyphs so that they do not overlap.
- **FGCOLOR** – (*COLOR*) The foreground color of the glyph. This is the line and outline color for graphical glyphs and the font color for text glyphs.
- **HEIGHT** – (*INT*) The height of the glyph. The height is orthogonal to the axis that defines the extent of the feature on the sequence map. For a text glyph, this is equivalent to the **FONTSIZE** attribute.
- **LABEL** – (*BOOL*) Whether the glyph should be labeled with its name, as dictated by the <FEATURE> label attribute in the DASGFF document.

**ARROW** – A double-headed arrow with an axis either orthogonal or parallel to the sequence map.

- **PARALLEL** – (*BOOL*) Arrows run either parallel or orthogonal to the sequence axis.

**ANCHORED\_ARROW** – A arrow that has an arrowhead at one end and an “anchor” (typically a diamond or line) at the other. The arrow points in the direction indicated by the `<ORIENTATION>` tag.

- **PARALLEL** – (*BOOL*) Arrows run either parallel or orthogonal to the sequence axis.

**BOX** – A rectangular box

- **LINEWIDTH** – (*INT*) Width of the glyph outline.

**CROSS** – A cross “+”. Commonly used for point mutations and other point-like features. (*no glyph-specific attributes*)

**DOT** – A small circle. Commonly used for point mutations and other point-like features. (*no glyph-specific attributes*)

**EX** – “X” marks the spot. Commonly used for point mutations and other point-like features. (*no glyph-specific attributes*)

**HIDDEN** – A feature that is invisible, intended to support semantic zooming schemes in which a feature is hidden at particular zooms. (*no attributes*)

**LINE** – A line. Lines are equivalent to arrows with both the `NORTHEAST` and `SOUTHWEST` attributes set to “no”.

- **STYLE** – (*LINE\_STYLE*) A style of “hat” draws an inverted V (commonly used for introns). A type of “solid” draws a horizontal solid line in the indicated color. A type of “dashed” draws a dashed horizontal line in the indicated color.

**SPAN** – A spanning region, the recommended representation is a horizontal line with vertical lines at each end. (*no glyph-specific attributes*)

**TEXT** – A bit of text.

- **FONT** – (*FONT*) The font.
- **FONTSIZE** – (*INT*) The font size.

- **STRING** – (*STRING*) The text to render.
- **STYLE** – (*FONT\_STYLE*) The style in which to render this glyph. Multiple **FONT\_STYLE** attributes may be present.

**TOOMANY** – Too many features than can be shown. Recommended for use in consolidating sequence homology hits. The recommended visual presentation is a set of overlapping boxes.

- **LINEWIDTH** – (*INT*) Width of the glyph outline.

**TRIANGLE** – A triangle. Commonly used for point mutations and other point-like features.

- **LINEWIDTH** – (*INT*) Width of the glyph outline.