



Metadata Management

Providing Dynamic Metadata of Robotic Telescopes to Stellaris¹

Deliverable	D2.7
Authors	Working Group Metadata Management
Editors	Frank Breitling
Date	April 29, 2008
Document Version	1.0.0
Current Version	1.0.0
Previous Versions	

A: Status of this Document

Public release.

B: Reference to project plan

Deliverable 7 of working group 2. It refers to the task TA II-7 *Provision of dynamic metadata to Stellaris*, describing the integration of dynamic metadata of robotic telescopes into the AstroGrid-D information service. Software components are discussed and documented.

C: Abstract

Two examples for dynamic metadata of telescopes are discussed, i.e. observation schedules and weather information. Data formats, the transfer of data to the AstroGrid-D information service via database triggers and the retrieval of this information via SPARQL queries are presented. Also usage of the *AstroGrid-D Timeline* for showing scheduling information is explained. It is a special web browser user interfaces of AstroGrid-D for visualizing dynamic metadata.

¹This work is part of the AstroGrid-D project and D-Grid. The project is funded by the German Federal Ministry of Education and Research (BMBF).

D: Change History

Version	Date	Name	Brief summary
0.1.0	17.03.2008	Frank Breitling	First draft.
0.2.0	18.03.2008	Frank Breitling	Second draft.
0.3.0	19.03.2008	Frank Breitling	Third draft, including references.
0.4.0	21.03.2008	Frank Breitling	Release to the working group. Minor corrections.
0.5.0	04.04.2008	Frank Breitling	Project release. Minor corrections also by S. White.
1.0.0	23.04.2008	Frank Breitling	Public release. Minor changes to ref. to B: project plan, C: abstract, wording, figure, etc.

E:

Contents

Abstract	1
Change History	2
1 Introduction	4
2 Dynamic Metadata Management	4
2.1 Weather Reports	4
2.2 Monitoring Observations and Schedules	5
2.3 Database Triggers	6
2.4 Retrieving Metadata with SPARQL Queries	6
3 Visualization	7
4 Conclusion	8
A Listings	8
References	14

1 Introduction

Dynamic metadata of robotic telescopes such as weather data is important for a telescope broker in a robotic telescope network, since it affects response of robotic telescopes. Here dynamic metadata management is outlined and an implementation for the robotic telescope STELLA-II [8] as an example is discussed. The implementation builds on the static metadata management described in [1].

2 Dynamic Metadata Management

Dynamic metadata is metadata which is subject to change. Therefore dynamic metadata management relies on monitoring, which provides a continuous update of information. Here the metadata management is illustrated for the robotic telescope STELLA-II. STELLA-II records its monitoring information to a PostgreSQL [5] database. The transfer of new information to the AstroGrid-D information service Stellaris is accomplished via database triggers. Currently the monitoring information includes weather reports and observation schedules.

2.1 Weather Reports

Weather information is important for the selection of telescopes for observations with short time windows. RTML [2] already provides a format for weather information. An example is shown in listing 1. STELLA-II uses RTML for the exchange of weather information with Stellaris. The exchange is a two step process. First the weather information is written to the STELLA-II database. Second a database trigger is released on every new entry updating an RDF/XML template which is transferred to Stellaris. The RDF/XML template is obtained via XSLT from RTML as for static data described in [1].

Listing 1: RTML template for a weather report.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <RTML version="3.2" mode="update" uid="rtml://STELLA-II.aip.de/WeatherReport
  "
3   xmlns="http://www.rtml.org/v3.2" xmlns:xsi="http://www.w3.org/2001/
4     XMLSchema-instance" xsi:schemaLocation="http://www.rtml.org/v3.2">
5   <History>
6     <Entry timeStamp="2007-02-07T15:30:00">
7       <Agent name="STELLA-II"/>
8     </Entry>
9   </History>
10  <WeatherReport begin="2007-02-07T15:30:00" end="2007-02-07T15:35:00">
11    <Seeing units="arcseconds">2</Seeing>
12    <Particles units="parts_per_million">3</Particles>
13    <Precipitation>0</Precipitation>
14    <RelativeHumidity units="percent">20</RelativeHumidity>
15    <Temperature units="degrees_celcius">20</Temperature>
16    <WindDirection units="degrees">20</WindDirection>
17    <WindSpeed units="kilometers_per_hour">23</WindSpeed>
18  </WeatherReport>
19  <Telescope name="STELLA-II" uref="rtml://STELLA-II.aip.de"/>
20 </RTML>
```

2.2 Monitoring Observations and Schedules

Monitoring of ongoing observations is important for the operation of telescopes. Therefore two additional XML formats have been investigated for communication exchange: the usage record [6] and the iCalendar [3] format. The former is preferred.

The advantage of the usage record format lies in its compatibility to job monitoring of AstroGrid-D and other grid communities. Consequently tools such as the *AstroGrid-D Timeline* which are based on usage records will also apply to monitoring of observations. An example for a usage record XML file describing a STELLA-II observation is shown in listing 2.

iCalendar is a compact format for the representation of timing information and has a corresponding RDF/XML representation as RDF Calendar [7]. Examples are shown in listing 3 and 9 respectively. Moreover iCalendar is compatible to wide-spread programs such as Google Calendar or Mozilla Sunbird. So graphical user interfaces already exist and need not be developed. However, compatibility to the usage record format was considered more important within AstroGrid-D.

Listing 2: Usage Record template describing an observation.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <JobUsageRecord xmlns="http://www.gridforum.org/2003/ur-wg#"
3   xmlns:urf="http://schema.ogf.org/urf/2003/09/urf"
4   xmlns:grddl="http://www.w3.org/2003/g/data-view#"
5   grddl:transformation="http://www.astrogrid-d.org/xml2rdf.xsl">
6
7   <RecordIdentity urf:recordId="$OBSERVATIONID" urf:createTime="$START"/>
8   <JobIdentity>
9     <LocalJobId>$OBSERVATIONID</LocalJobId>
10    <job_grid_id>$OBSERVATIONID</job_grid_id>
11  </JobIdentity>
12  <UserIdentity>
13    <LocalUserId>$PI</LocalUserId>
14    <GlobalUserName>$PI</GlobalUserName>
15  </UserIdentity>
16  <JobName>$OBSERVATION</JobName>
17  <Status>$STATUS</Status>
18  <creation_time>$STOP</creation_time>
19  <StartTime>$START</StartTime>
20  <EndTime>$STOP</EndTime>
21  <Host>STELLA-II.aip.de</Host>
22  <Resource>telescope</Resource>
23 </JobUsageRecord>

```

Listing 3: iCalendar template describing an observation.

```

1 BEGIN:VCALENDAR
2 PRODID:-//Globotel//Version 0.1//EN
3 VERSION:2.0
4 X-WR-NAME:OpenTel/STELLA-II.aip.de/monitoing/$CALENDAR
5
6 BEGIN:VEVENT
7 DTSTART;TZID=Europe/Berlin:$START
8 DTEND;TZID=Europe/Berlin:$STOP
9 URL:OpenTel/STELLA-II.aip.de/observations/$OBSERVATION
10 UID:$START
11 DESCRIPTION:$STATUS

```

```

12 LAST-MODIFIED: $START
13 SUMMARY: $OBSERVATIONID
14 END:VEVENT
15
16 END:VCALENDAR

```

2.3 Database Triggers

Often data from telescopes is recorded to databases. An example is STELLA-II, which uses a PostgreSQL database. Database software provides trigger mechanisms which can execute functions for transfer of new data to the information service. Two database triggers have been developed to transfer new weather data and observation schedules of STELLA-II to Stellaris. These PostgreSQL triggers are shown in the appendix in listing 5 and 6.

The main function is separate from the trigger, since this allows for an execution without the release of the trigger by new data. Now on demand the full observation history can be restored by the SQL command

```

select schedule(obs.dateobs, obs.date, obs.object, done.object, proposal.pi, done.success)from
obs, done, task, proposal where obs.objid=done.object and done.objid=task.object and task.propid
=proposal.propid and obs.date < '2008-03-01 08:20:00'.

```

The main function is embedded in SQL but written in Perl which allows the exchange of information with web services through HTTP requests. The corresponding examples can be found in listing 7 and 8 of the appendix. They serve as templates and are provided in the “dbtemplates” folder of the OpenTel software package [4]. The software can be installed into PostgreSQL with the command

```
psql -h gavo3 -d stella -p 5433 stella -f sii_schedule.sql.
```

2.4 Retrieving Metadata with SPARQL Queries

As described in [1] the metadata in Stellaris can be retrieved through SPARQL queries. Example queries for retrieving the weather and scheduling information are provided in the SPARQL folder of the OpenTel software package. An example and the output is shown in listing 4.

Listing 4: SPARQL query for retrieving RTML weather information of telescopes.

```

1 PREFIX rtml: <http://www.rtml.org/v3.2>
2 PREFIX x2r: <http://www.astrogrid-d.org/2007/08/14-xml2rdf#>
3
4 SELECT ?tel ?timeStamp ?Precipitation ?Temperature ?RelativeHumidity
5 ?WindSpeed ?WindDirection
6 WHERE { graph ?g {
7   ?head rtml:WeatherReport ?wr .
8   ?head rtml:History ?h . ?h rtml:Entry ?e . ?e rtml:timeStamp ?timeStamp .
9   ?head rtml:Telescope ?uref . ?uref rtml:uref ?tel .
10  ?wr rtml:Temperature ?T . ?T x2r:value ?Temperature .
11  ?wr rtml:RelativeHumidity ?hum . ?hum x2r:value ?RelativeHumidity .
12  ?wr rtml:Precipitation ?Precipitation .
13  ?wr rtml:WindSpeed ?ws . ?ws x2r:value ?WindSpeed .
14  ?wr rtml:WindDirection ?wd . ?wd x2r:value ?WindDirection .
15 }}

```

Table 1: Weather report as obtained through the SPARQL query of listing ??.

Telescope UID	rtml://de.aip.STELLA-II
timeStamp	2008-03-14 17:35:28
Precipitation	0
Temperature	10.564
RelativeHumidity	26.336
WindSpeed	6.129
WindDirection	180.86

3 Visualization

Since we chose the usage record format for scheduling metadata, the AstroGrid-D Timeline, a graphical user interface for monitoring of grid jobs, could be directly applied. Only minor modifications were necessary to display telescope observations like compute jobs. The result is show in Fig. 1.

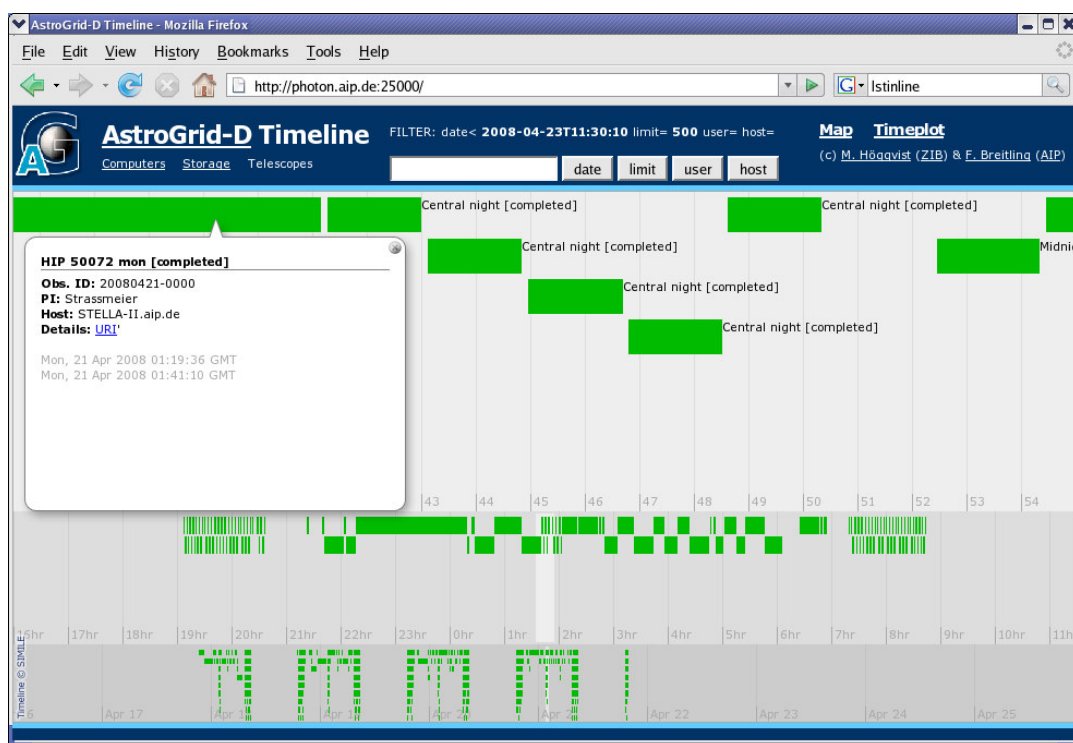


Figure 1: Screen shot of the *AstroGrid-D Timeline*, an interactive web browser user interface for displaying time based information. It displays scheduling information of the robotic telescope STELLA-II. Filters to search for special information are implemented.

4 Conclusion

Dynamic metadata of robotic telescopes has been integrated into the metadata management of AstroGrid-D and into the information service Stellaris. An implementation for the robotic telescope STELLA-II has been discussed. Templates are provided for the integration of other telescopes. The retrieval of dynamic metadata is accomplished with SPARQL queries for which corresponding templates are provided as well. A graphical user interface of AstroGrid-D has also been adapted for real time monitoring of ongoing observations. The presented implementations provide an extension of static metadata management of robotic telescopes, that allows for an advanced brokering algorithm for time critical observations in a network of robotic telescope.

A Listings

Listing 5: Database trigger for calling a function that provides weather information of STELLA-II to Stellaris.

```
1 CREATE OR REPLACE FUNCTION sii_weather() RETURNS trigger AS $mark$
2
3 DECLARE
4     date            timestamp;
5     taussen         real;
6     wind            real;
7     winddir         real;
8     haussen         real;
9     rain            real;
10
11 BEGIN
12 PERFORM weather(NEW.date, NEW.taussen, NEW.wind, NEW.winddir, NEW.haussen,
13     NEW.rain);
14 RETURN NEW;
15 END;
16
17 $mark$ LANGUAGE plpgsql;
18
19 CREATE TRIGGER sii_weather_trig
20     BEFORE INSERT OR UPDATE ON env
21     FOR EACH ROW
22     EXECUTE PROCEDURE sii_weather();
```

Listing 6: Database trigger for colling a function that provides scheduling information of STELLA-II to Stellaris.

```
1 CREATE OR REPLACE FUNCTION sii_schedule() RETURNS trigger AS $mark$
2
3 DECLARE
4     dateobs         timestamp;
5     date            timestamp;
6     object          character varying(255);
7     objid           character varying(255);
8     propid          character varying(255);
9     pi              character varying(255);
10    success          boolean;
```

```

11
12 BEGIN
13 PERFORM schedule(NEW.dateobs, NEW.date, NEW.object, done.object, proposal.pi
    , done.success) FROM done, task, proposal where NEW.objid=done.object
    and done.objid=task.object and task.propid=proposal.propid;
14 RETURN NEW;
15 END;
16
17 $mark$ LANGUAGE plpgsql;
18
19 CREATE TRIGGER sii_schedule_trig
20     AFTER INSERT OR UPDATE ON obs
21     FOR EACH ROW
22     EXECUTE PROCEDURE sii_schedule();

```

Listing 7: Database function for weather information.

```

1 CREATE OR REPLACE FUNCTION weather(timestamp, real, real, real, real, real)
    RETURNS integer AS $function$
2
3   ### Perl: start ###
4   use strict;
5   use URI;
6   use HTTP::Request; use LWP::UserAgent;
7
8   my $timestamp = substr(@_[0],0,19);
9   my $temperature = @_[1];
10  my $windspeed = @_[2];
11  my $winddir = @_[3];
12  my $humidity = @_[4];
13  my $precipitation = int(@_[5]);
14
15  my $lt = localtime(time());
16
17  elog( NOTICE , "weather($timestamp, ...) started at $lt" );
18
19  my $RDFtemplate = <<"EOF";
20  <?xml version="1.0"?>
21  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
22    <ns1:RTML xmlns:ns1="http://www.rtml.org/v3.2" rdf:about="">
23    <ns1:version>3.2</ns1:version>
24    <ns1:mode>update</ns1:mode>
25    <ns1:uid>rtml://STELLA-II.aip.de/WeatherReport</ns1:uid>
26    <xsi:schemaLocation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
    ">http://www.rtml.org/v3.2 file:RTML-v3.2.xsd</xsi:schemaLocation>
27    <ns1:History>
28      <ns1:History rdf:about="#History">
29        <ns1:Entry>
30          <ns1:Entry rdf:about="#History/Entry">
31            <ns1:timeStamp>$timestamp</ns1:timeStamp>
32            <ns1:Agent>
33              <ns1:Agent rdf:about="#History/Entry/Agent">
34                <ns1:name>STELLA-II</ns1:name>
35              </ns1:Agent>
36            </ns1:Agent>
37          </ns1:Entry>
38        </ns1:Entry>
39      </ns1:History>
40    </ns1:History>

```

```
41 <ns1:WeatherReport >
42   <ns1:WeatherReport rdf:about="#WeatherReport">
43     <ns1:begin>$timestamp</ns1:begin>
44     <ns1:end>$timestamp</ns1:end>
45     <ns1:Seeing >
46       <ns1:Seeing rdf:about="#WeatherReport/Seeing">
47         <ns1:units>arcseconds</ns1:units>
48         <x2r:value xmlns:x2r="http://www.astrogrid-d.org/2007/08/14-
49           xml2rdf#">undetermined</x2r:value>
50       </ns1:Seeing >
51     </ns1:Seeing >
52     <ns1:Particles >
53       <ns1:Particles rdf:about="#WeatherReport/Particles">
54         <ns1:units>parts_per_million</ns1:units>
55         <x2r:value xmlns:x2r="http://www.astrogrid-d.org/2007/08/14-
56           xml2rdf#">undetermined</x2r:value>
57       </ns1:Particles >
58     </ns1:Particles >
59     <ns1:Precipitation>$precipitation</ns1:Precipitation>
60     <ns1:RelativeHumidity >
61       <ns1:RelativeHumidity rdf:about="#WeatherReport/RelativeHumidity">
62         <ns1:units>percent</ns1:units>
63         <x2r:value xmlns:x2r="http://www.astrogrid-d.org/2007/08/14-
64           xml2rdf#">$humidity</x2r:value>
65       </ns1:RelativeHumidity >
66     </ns1:RelativeHumidity >
67     <ns1:Temperature >
68       <ns1:Temperature rdf:about="#WeatherReport/Temperature">
69         <ns1:units>degrees_celcius</ns1:units>
70         <x2r:value xmlns:x2r="http://www.astrogrid-d.org/2007/08/14-
71           xml2rdf#">$temperature</x2r:value>
72       </ns1:Temperature >
73     </ns1:Temperature >
74     <ns1:WindDirection >
75       <ns1:WindDirection rdf:about="#WeatherReport/WindDirection">
76         <ns1:units>degrees</ns1:units>
77         <x2r:value xmlns:x2r="http://www.astrogrid-d.org/2007/08/14-
78           xml2rdf#">$winddir</x2r:value>
79       </ns1:WindDirection >
80     </ns1:WindDirection >
81     <ns1:WindSpeed >
82       <ns1:WindSpeed rdf:about="#WeatherReport/WindSpeed">
83         <ns1:units>kilometers_per_hour</ns1:units>
84         <x2r:value xmlns:x2r="http://www.astrogrid-d.org/2007/08/14-
85           xml2rdf#">$windspeed</x2r:value>
86       </ns1:WindSpeed >
87     </ns1:WindSpeed >
88   </ns1:WeatherReport >
89 </ns1:WeatherReport >
90 <ns1:Telescope >
91   <ns1:Telescope rdf:about="#Telescope">
92     <ns1:name>STELLA-II</ns1:name>
93     <ns1:uref rdf:resource="rtml://STELLA-II.aip.de"/>
94   </ns1:Telescope >
95 </ns1:Telescope >
96 </ns1:RTML >
97 </rdf:RDF >
98 EOF
99
```

```

94 my $req = HTTP::Request->new
95   ("PUT", "http://stellaris.zib.de:24030/OpenTel/STELLA-II.aip.de/weather?
      modification=replace",
96   HTTP::Headers->new(Content_Length=>length($RDFtemplate),
97   Content_Type => "application/rdf+xml"),
98   $RDFtemplate);
99 my $ua = LWP::UserAgent->new(); my $res = $ua->request($req);
100
101 if ($res->is_error) {
102   print STDERR "PUT ERROR: ".$res->status_line.", using POST instead.\n";
103   my $req = HTTP::Request->new
104     ("POST", "http://stellaris.zib.de:24030",
105     HTTP::Headers->new(Content_Length=>length($RDFtemplate),
106     Content_Type => "application/rdf+xml",
107     Slug=>"/OpenTel/STELLA-II.aip.de/weather"),
108     $RDFtemplate);
109   my $ua = LWP::UserAgent->new(); my $res = $ua->request($req);
110   if ($res->is_error) { print STDERR "POST ERROR: ".$res->status_line."\n"
      ; }
111 }
112
113 #open FILE, ">/tmp/weather/". $timestamp. ".xml";
114 #print FILE "status_line: ".$res->status_line."\n";
115 #print FILE "is_error: ".$res->is_error."\n";
116 #print FILE $RDFtemplate;
117 #close FILE;
118
119 #print length($RDFtemplate)."\n";
120
121 elog( NOTICE , "weather($timestamp, ...) finished @ $!t" );
122
123 return 0;
124
125 ### Perl: end ###
126
127 $function$ LANGUAGE plperl;

```

Listing 8: Database trigger for scheduling information.

```

1 CREATE OR REPLACE FUNCTION schedule(timestamp, timestamp, character varying
      (255), character varying(255), character varying(255), boolean) RETURNS
      integer AS $test$
2
3 ### Perl: start ###
4 use strict;
5 use URI;
6 use HTTP::Request; use LWP::UserAgent;
7
8 my $DATEOBS = @_ [0]; $DATEOBS =~ s/ /T/;
9 my $DATE = @_ [1]; $DATE =~ s/ /T/;
10 my $OBSERVATION = @_ [2];
11 my $OBSERVATIONID = @_ [3];
12 my $PI = @_ [4];
13 my $SUCCESS = @_ [5];
14
15 my $STATUS; if ($SUCCESS=='t') { $STATUS="completed"; } else { $STATUS="queued
      "; }
16 my $YEAR = substr($DATE,0,4);
17 my $MONTH = substr($DATE,5,2);

```

```

18 my $DAY = substr($DATE,8,2);
19 my $START = substr($DATEOBS,0,19);
20 my $STOP = substr($DATE,0,19);
21
22 my $t = time();
23 my $lt = localtime(time());
24
25 elog( NOTICE , "schedule($START, ...) started at $lt" );
26
27 my $RDFtemplate = <<"EOF";
28 <?xml version="1.0"?>
29 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
30   <ns1:JobUsageRecord xmlns:ns1="http://www.gridforum.org/2003/ur-wg#" rdf:
      about="">
31     <grddl:transformation xmlns:grddl="http://www.w3.org/2003/g/data-view#">
          http://www.astrogrid-d.org/xml2rdf.xsl</grddl:transformation>
32     <ns1:RecordIdentity>
33       <ns1:RecordIdentity rdf:about="#RecordIdentity">
34         <urf:recordId xmlns:urf="http://schema.orgf.org/urf/2003/09/urf">
              $OBSERVATIONID</urf:recordId>
35         <urf:createTime xmlns:urf="http://schema.orgf.org/urf/2003/09/urf">
              $START</urf:createTime>
36       </ns1:RecordIdentity>
37     </ns1:RecordIdentity>
38     <ns1:JobIdentity>
39       <ns1:JobIdentity rdf:about="#JobIdentity">
40         <ns1:LocalJobId>$OBSERVATIONID</ns1:LocalJobId>
41         <ns1:job_grid_id>$OBSERVATIONID</ns1:job_grid_id>
42       </ns1:JobIdentity>
43     </ns1:JobIdentity>
44     <ns1:UserIdentity>
45       <ns1:UserIdentity rdf:about="#UserIdentity">
46         <ns1:LocalUserId>$PI</ns1:LocalUserId>
47         <ns1:GlobalUserName>$PI</ns1:GlobalUserName>
48       </ns1:UserIdentity>
49     </ns1:UserIdentity>
50     <ns1:JobName>$OBSERVATION</ns1:JobName>
51     <ns1:Status>$STATUS</ns1:Status>
52     <ns1:creation_time>$STOP</ns1:creation_time>
53     <ns1:StartTime>$START</ns1:StartTime>
54     <ns1:EndTime>$STOP</ns1:EndTime>
55     <ns1:Host>STELLA-II.aip.de</ns1:Host>
56     <ns1:Resource>telescope</ns1:Resource>
57   </ns1:JobUsageRecord>
58 </rdf:RDF>
59 EOF
60
61 my $req = HTTP::Request->new
62   ("PUT", "http://stellaris.zib.de:24030/OpenTel/STELLA-II.aip.de/
      UsageRecords/" .
63     "$YEAR/$MONTH/$DAY/$START?modification=replace",
64     HTTP::Headers->new(Content_Length=>length($RDFtemplate),
65                         Content_Type => "application/rdf+xml"),
66     $RDFtemplate);
67 my $ua = LWP::UserAgent->new(); my $res = $ua->request($req);
68
69 if ($res->is_error) {
70   print STDERR "PUT ERROR: ".$res->status_line.", using POST instead.\n";
71   my $req = HTTP::Request->new

```

```

72     ("POST", "http://stellaris.zib.de:24030",
73     HTTP::Headers->new(Content_Length=>length($RDFtemplate),
74     Content_Type => "application/rdf+xml",
75     Slug=>"OpenTel/STELLA-II.aip.de/UsageRecords/" .
76     "$YEAR/$MONTH/$DAY/$START"),
77     $RDFtemplate);
78 my $ua = LWP::UserAgent->new(); my $res = $ua->request($req);
79 if ($res->is_error) { print STDERR "POST ERROR: ".$res->status_line."\n"
80 ; }
81
82 #open FILE, ">/tmp/schedule/". $START. ".xml";
83 #print FILE "status_line: ".$res->status_line."\n";
84 #print FILE "is_error: ".$res->is_error."\n";
85 #print FILE $RDFtemplate;
86 #close FILE;
87
88 #print length($usage_record)."\n";
89
90 elog( NOTICE , "schedule($START, ...) finished @ $!t" );
91
92 return 0;
93
94 ### Perl: end ###
95
96 $test$ LANGUAGE plperlu;

```

Listing 9: RDF Calendar template corresponding to listing 3.

```

1 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
2 xmlns="http://www.w3.org/2002/12/cal/icaltzd#"
3 <Vcalendar xmlns:x-wr="http://www.w3.org/2002/12/cal/prod/
4 Globotel_V_11e1c392a7blad96#"
5 rdf:about="OpenTel/STELLA-II.aip.de/monitoing/observatitons/$CALENDAR">
6
7 <prodid>-//Globotel//Version 0.1//EN</prodid>
8 <version>2.0</version>
9 <component>
10 <Vevent rdf:about="$START">
11 <dtstart rdf:datatype="http://www.w3.org/2002/12/cal/tzd/Europe/
12 Berlin#tz">
13 $START</dtstart>
14 <dtend rdf:datatype="http://www.w3.org/2002/12/cal/tzd/Europe/Berlin
15 #tz">
16 $STOP</dtend>
17 <url rdf:resource="OpenTel/STELLA-II.aip.de/observations/$
18 OBSERVATION"/>
19 <uid>$START</uid>
20 <description>$STATUS</description>
21 <lastModified rdf:datatype="http://www.w3.org/2002/12/cal/icaltzd#
22 dateTime">
23 $START</lastModified>
24 <summary>$OBSERVATIONID</summary>
25 </Vevent>
26 </component>
27 </Vcalendar>
28 </rdf:RDF>

```

F: References / Bibliography

References

- [1] F. Breitling, M. Braun. Providing Static Metadata of Robotic Telescopes to Stellaris. Technical Report D2.4, AstroGrid-D project, March 2007.
- [2] F. V. Hessman. Remote Telescope Markup Language (RTML). *Astronomische Nachrichten*, 327:751–+, September 2006.
- [3] Internet Calendaring and Scheduling Core Object Specification (iCalendar). <http://www.w3.org/2002/12/cal/rfc2445>.
- [4] Members of AstroGrid-D. Ottools. <http://www.gac-grid.org/project-products/RoboticTelescopes.html>, 2008.
- [5] PostgreSQL: The world's most advanced open source database. <http://www.postgresql.org/>, 2008.
- [6] R. Mach, R. Lepro-Metz, S. Jackson, L. McGinnis. Usage Record — Format Recommendation. <http://www.psc.edu/~lfm/PSC/Grid/UR-WG/UR-Spec-gfd.58.pdf>.
- [7] RDF Calendar Workspace. <http://www.w3.org/2002/12/cal/>.
- [8] K. G. Strassmeier, T. Granzer, M. Weber, M. Woche, M. I. Andersen, J. Bartus, S.-M. Bauer, F. Dionies, E. Popow, T. Fechner, G. Hildebrandt, A. Washuettl, A. Ritter, A. Schwoppe, A. Staude, J. Paschke, P. A. Stolz, M. Serre-Ricart, T. de la Rosa, and R. Arnay. The STELLA robotic observatory. *Astronomische Nachrichten*, 325:527–+, October 2004.