





SKA1 SDP DELIVERY PROTOTYPING REPORT

Document Number..... SKA-TEL-SDP-0000152
 Document Type..... REP
 Revision..... 01
 Author..... R. Simmonds, S. Sanchez, D. Aikema, S. Goliath, S. Gaudet
 Date..... 2018-10-31
 Document Classification..... Unrestricted
 Status..... Released

Name	Designation	Affiliation	Signature
Authored by:			
Rob Simmonds	SDP DELIV Lead	UCT	 <small>RWJ Simmonds (Oct 31, 2018)</small>
			Date:
Owned by:			
			Date:
Approved by:			
			Date:
Released by:			
Paul Alexander	SDP Project Lead	University of Cambridge	 <small>Paul Alexander (Oct 31, 2018)</small>
			Date:

DOCUMENT HISTORY

Revision	Date Of Issue	Engineering Change Number	Comments
01	2018-10-31		Prepared for M21, SDP CDR review

DOCUMENT SOFTWARE

	Package	Version	Filename
Word processor	Google Docs		SKA-TEL-SKO-0000000-01_GenDocTemplate
Block diagrams			
Google docs Add-ons	Cross Reference Table of contents List of figures		Used for figure & table numbering and references. Used for heading numbering. Used to generate list of figures and tables

ORGANISATION DETAILS

Name	SDP Consortium
Lead Organisation	The Chancellor, Masters and Scholars of the University of Cambridge The Old Schools Trinity Lane Cambridge CB1 1TN United Kingdom
Website	www.ska-sdp.org

© Copyright 2018 University of Cape Town



This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

Table of Contents

List of Abbreviations	4
1 Introduction	5
1.1 Purpose of the document	5
1.2 Scope of the document	5
2 References	5
2.1 Applicable documents	5
2.2 Reference documents	5
3. Data Distribution and Management Tools	6
4. Review of the Next Generation Archive System (NGAS)	7
4.1 NGAS description	7
4.2 NGAS subscription function	7
4.3 Tests	8
4.3.1 Software installation	8
4.3.2 Creating a subscription with several data streams assigned	8
4.3.3 Creating a subscription using a Filter Plug-In	9
5. Review of the Apache Object Oriented Database Technology	11
5.1 Apache OODT	11
5.2 OODT Services for SDP Data Transfer Service	11
5.2.1 OODT CAS File Manager	11
5.2.2 OODT CAS Workflow Manager	12
5.2.3 OODT PushPull Service	12
6. Review of the File Transfer Service technology	12
6.1 Description	12
6.2 Use with MeerKAT	13
6.3 Performance testing	14
7. IVOA Data Discovery and Access Services and Tools	14
8. Catalogue Sizing	14
8.1 Estimation of the Science Data Product Catalogue growth	15
8.1.1 Estimation of the amount of ODPs generated per year	15
8.2 Conclusions	19
Appendix A: Data Distribution Tool Assessment	20
Appendix B: Assessment of IVOA Data Discovery and Access Services	28

List of Abbreviations

CERN	The European Centre for Nuclear Research
COTS	Commercial Off-The-Shelf
DELIV	Sub-element of the Science Data Processor consortium
FTS	File Transfer Service
GFAL	Grid File Access Library
GridFTP	A high-performance, secure, reliable data transfer protocol
ICRAR	International Centre for Radio Astronomy Research
iRods	Integrated Rule-Oriented Data System
IVOA	International Virtual Observatory Alliance
NGAS	Next Generation Archive System
ODP	Observatory Data Product
ODT	Apache Object Oriented Data Technology
ODT CAS	ODT Catalog and Archive Service
PhEDEx	Physics Experiment Data Export
REST	Representational State Transfer Technology
SCP	Secure copy
SDPC	Science Data Product Catalogue
SRM	Storage Resource Manager
SRC	SKA Regional Centre
SSH	Secure Shell
TDDM	Tiered Data Delivery Manager
VM	Virtual Machine

1 Introduction

1.1 Purpose of the document

The SDP consortium followed a system engineering process that included the creation of a Product Tree, which can be defined as an architectural view of the software elements included in the SDP platform, as a way of identifying and estimating the level of risk of not being able to cost effectively provide each element. Once the Product Tree was created, the next step was to prototype those Product Tree elements where there was a significant risk.

This report provides a summary of the work performed as part of the prototyping activities for the SDP. It should be noted that none of the items in this element carry high risk for the project. A great deal of the work for this element has involved testing to make sure that systems currently in use (and performing well for grid projects) could also be used to fulfil requirements in the SKA1 System Baseline Design [RD1].

1.2 Scope of the document

The Delivery System has evolved considerably during the design process. It was reduced in scope, with some components moved to the Observatory Support Tools. This is because, while these components are believed to be needed to run the Observatory, resourcing the development of these to scale for the use of all SKA users was determined to be out of scope of the SDP. This document provides information on tools considered by the DELIV team within the SDP consortia for prototyping and provides information on work done to ensure that risks were minimised.

The following sections consider Data Transfer Tools and IVOA based tools that will support access to the Science Data Product Catalogue.

2 References

2.1 Applicable documents

There are no applicable documents concerning the SDP Delivery Prototyping Report.

2.2 Reference documents

The following documents are referenced in this document. In the event of conflict between the contents of the referenced documents and this document, **this document** shall take precedence.

- [RD1] SKA-TEL-SKO-0000002, SKA1 SYSTEM BASELINE DESIGN V2, Rev 03 . P. Dewdney
2016-02-26
- [RD4] SKA-TEL-SDP-0000025, SDP Delivery System Design Rev 02, R. Simmonds et al.,
2016-07-15

- [RD5] Optimising NGAS for the MWA Archive, <https://arxiv.org/abs/1308.6083>, C. Wu et al, 2013
- [RD6] NGAS documentation: <https://ngas.readthedocs.io>
- [RD7] Presentation material:
http://archive.apachecon.com/na2013/presentations/27-Wednesday/Apache_in_Science/11:15-ODDT_GRAVITE.pptx
- [RD8] Evaluating Cloud Computing in the NASA DESDynI Ground Data System, J J Tran et al, <https://confluence.ska-sdp.org/display/WBS/Benchmarking+results+for+ODDT>
- [RD9] FTS3 Documentation (<http://fts3-docs.web.cern.ch/>)
- [RD10] SKA-TEL-SDP-0000038, SDP System Sizing Estimates, Rev 03

3. Data Distribution and Management Tools

The matrix in Appendix A shows the initial analysis of data management tools that were considered the start of the SDP design process. In the end only NGAS and OODT from this table were given extensive analysis, though additional tools were considered later. Given that PhEDEx has been largely deprecated in place of FTS before we prototyped this, it was not considered for prototyping. Also iRods provided an end-to-end data management system that made parts of it outside the scope of the SDP. In addition, previous experience from members of the DELIV team indicated that it was not going to be as straightforward to get the same level of WAN transfer performance using this as with other tools being considered. Since the aim here was just to reduce risk, iRods was not prototyped as part of this work.

The Tiered Data Delivery Manager (TDDM) represents the services in charge of managing the transfers between of SDP sites and the SKA Regional Centres (SRCs). A detailed description of this element can be found in the SDP Delivery System Design Rev 02 [RD4].

The objective of the prototyping summarized in the following sections is to analyse how NGAS and Apache OODT can provide a set of features required to implement one of the service in TDDM, the Data Transfer Service. In particular, these features are:

- **Data transfer request handler:** The system should be able to handle different data transfer requests, and group them according to a set of attributes like e.g. the project /user to which the data request belongs.
- **Data distribution policy based on priorities.** The transfer tasks will be ordered according to a set of pre-defined user/project priorities. The system will avoid starvation, implementing, for example some kind of scheduling in which the network time allocated to each task depends on the task priority.

Later on we moved on to looking at CERN's FTS service in conjunction with GridFTP. This is the solution that we recommend as the best solution at this time for the SKA baseline design. We acknowledge that going forward more extensive analysis of what tools will be used and shared

between SKA and CERN will take place, which could lead to new recommendations for solutions that meet the needs of our and other communities.

4. Review of the Next Generation Archive System (NGAS)

4.1 NGAS description

The Next-Generation Archive System is a Python-based open source software initially developed in the European Southern Observatory including the ALMA Archive, the NRAO eVLA archive and the La Silla Paranal Observatory [RD5]. It is also used to develop the Murchison Widefield Array (MWA) archive, however some optimisations were required:

1. Support for a data ingestion rate of approximately 400MB/s
2. Efficient flow control and throughput-oriented data transfer through public network.
3. Integration with the commercial Hierarchical Storage Management: data staging and replacement based on user access patterns
4. Support processing-aware data staging in a way to optimise the parallelism between data movement and data processing

The optimized NGAS has, a priori, some features that are useful for implementing a SDP Data Transfer Service. In particular, the subscription function allows users to indicate which data (according to a set of attributes) they want to receive. In addition, different priority level can be assigned to each subscriptions.

4.2 NGAS subscription function

NGAS subscription service provides a method to control the dataflow. As stated in [RD6] : this service makes it possible to synchronize a full or partial set of data files to remote Data Subscribers which can be other NGAS nodes. A client subscribing for data is referred to as a Data Subscriber. An NGAS Server, which delivers data to such a Subscriber, is referred to as a Data Provider. A Data Subscriber can specify to receive data files from a certain point in the past to the present day. If the time is not defined then only newly archived files will be delivered to the subscriber.

The NGAS subscription service can be configured through the following parameters (as described in [RD6]):

- **Start date:** Date from which the data to deliver is taken into account. If not specified the time when the SUBSCRIBE command was received is taken as start date
- **Concurrent threads:** A single dataflow (linked to a NGAS subscription) supports multiple concurrent data streams, multiplexing the same TCP link to maximise the network bandwidth utilisation.
- **Priority:** Priority for delivering data to this Data Subscriber. The lower the number, the higher the priority. Clients with a higher priority, get more CPU time in connection with the data delivery
- **Filter plug-in:** python script that defines a filter that will be applied to data files before they are delivered.

The start date, priority and the number of concurrent can be updated “on the fly”, after the subscription is created, using the command UNSUBSCRIBE.

4.3 Tests

4.3.1 Software installation

NGAS was installed on a Virtual Machine (VM) following the instructions provided in [RD6]. An automatic *install and configure* script is provided via *Fabric*, a tool that allows to perform commands in one or more hosts, local or remote (through SSH). The NGAS software was installed using this tool. It was needed the support from the ICRAR team, who fixed some issues that were found during the installation process.

In order to test some of the NGAS functionalities, two NGAS servers were configured in the same VM, each one running on a different port.

4.3.2 Creating a subscription with several data streams assigned

Several data streams can be assigned to NGAS subscription. It seems this means that in case several files needs to be delivered to the subscriber, NGAS will be able to transfer in parallel an amount of files equal to the number of data streams assigned to the subscription.

In order to check this functionality we created a new subscription with an old date, 6 concurrent threads and S21DEC_6 as identifier:

```
curl \
'http://localhost:7777/SUBSCRIBE?priority=1&url=http://localhost:7778/QARCHIVE&subscr_id=S21DEC_6&concurrent_threads=6&start_date=2016-09-13T00:00:00.000'
```

In this way, all the files stored in the NGAS serving on port 7777 will start to be delivered to the second NGAS server on port 7778. Since the amount of stored files are more than 6 (there are about 20 files), NGAS will use the 6 threads to copy them.

Check of the active network connections were made to localhost:7778, just after creating this subscription in order to know how many connections are created and if this number is the same as the number of data streams assigned to the subscription

For this, we executed the commands “`sudo ps -aux | grep python`”, to see the PID of each NGAS server, and “`sudo netstat -putan | grep 7778`”, to monitor the connections that are being created to the port 7778.

Next figure shows a snapshot of the results of the previous commands. In the upper terminal, we can see that the PID of one of the NGAS server is 20902 and the other one is 25981. In the lower terminal we can count up to 18 established connections: 9 incoming and 9 outgoing connections. Apparently the extra connections (we expected just 6 incoming/outgoing connections) are due to previous subscriptions.


```

Applications Places
student@student-VirtualBox: ~
dit View Search Terminal Help
3,0s: sudo ps -aux | grep python
Wed Dec 21 11:02:40 2016

19919 0.0 0.1 75364 2984 pts/21 S+ 10:52 0:00 sudo watch -n 3 sudo ps -aux | grep python
19920 0.1 0.0 18360 1996 pts/21 S+ 10:52 0:01 watch -n 3 sudo ps -aux | grep python
nt 20902 4.3 62.6 5423120 1282744 pts/17 SL+ 10:58 0:09 /home/student/ngas_rt/bin/python /home/student/ngas_rt/bin/ngamsServer -cfg /home/student/NGAS/cfg/nga
ver.conf -v 3 -autoonline
21517 0.0 0.0 18356 552 pts/21 S+ 11:02 0:00 watch -n 3 sudo ps -aux | grep python
21518 0.0 0.0 4448 708 pts/21 S+ 11:02 0:00 sh -c sudo ps -aux | grep python
21520 0.0 0.1 15944 2160 pts/21 S+ 11:02 0:00 grep python
nt 25981 0.5 0.9 722984 19352 pts/1 SL+ dtc12 68:21 /home/student/ngas_rt/bin/python /home/student/ngas_rt/bin/ngamsServer -cfg /home/student/NGAS2/cfg/ng
rver.conf -v 3 -autoonline

<?xml version="1.0" ?>
<!DOCTYPE NgamsStatus
<NgamsStatus>
  <Status Date="2016-1
  E" Status="FAILURE" Su
  </NgamsStatus>(ngas_rt)
File Edit View Search Terminal Help
Every 6,0s: sudo netstat -putan| grep 7778
Wed Dec 21 11:02:41 2016

tcp 0 0 0.0.0.0:7778 0.0.0.0:* LISTEN 25981/python
tcp 0 0 127.0.0.1:7778 127.0.0.1:44084 ESTABLISHED 25981/python
tcp 0 0 127.0.0.1:44082 127.0.0.1:7778 ESTABLISHED 20902/python
tcp 0 0 127.0.0.1:44084 127.0.0.1:7778 ESTABLISHED 20902/python
tcp 0 0 127.0.0.1:7778 127.0.0.1:44118 ESTABLISHED 25981/python
tcp 0 0 127.0.0.1:7778 127.0.0.1:44086 ESTABLISHED 25981/python
tcp 0 0 127.0.0.1:44124 127.0.0.1:7778 ESTABLISHED 20902/python
tcp 0 0 127.0.0.1:7778 127.0.0.1:44088 ESTABLISHED 25981/python
tcp 0 0 127.0.0.1:7778 127.0.0.1:44082 ESTABLISHED 25981/python
tcp 0 0 127.0.0.1:7778 127.0.0.1:44122 ESTABLISHED 25981/python
tcp 0 0 127.0.0.1:44088 127.0.0.1:7778 ESTABLISHED 20902/python
tcp 0 0 127.0.0.1:44126 127.0.0.1:7778 ESTABLISHED 20902/python
tcp 0 0 127.0.0.1:44086 127.0.0.1:7778 ESTABLISHED 20902/python
tcp 0 0 127.0.0.1:44118 127.0.0.1:7778 ESTABLISHED 20902/python
tcp 0 0 127.0.0.1:7778 127.0.0.1:44124 ESTABLISHED 25981/python
tcp 0 0 127.0.0.1:44122 127.0.0.1:7778 ESTABLISHED 20902/python
tcp 0 0 127.0.0.1:7778 127.0.0.1:44094 ESTABLISHED 25981/python
tcp 0 0 127.0.0.1:44094 127.0.0.1:7778 ESTABLISHED 20902/python
tcp 0 0 127.0.0.1:7778 127.0.0.1:44126 ESTABLISHED 25981/python

```

Figure 1: Snapshot showing the output from the commands “sudo ps -aux | grep python” and “sudo netstat -putan| grep 7778”

4.3.3 Creating a subscription using a Filter Plug-In

When a NGAS subscription is created, it is possible to specify a filter that will be applied to datafiles before being delivered to the subscriber. In this way it is possible, for example, to create a subscription that only retrieve data owned by a specific project or user.

We have created a Filter Plug-In, which basically is a python script, in order to create a subscription that only will retrieve data files with the format FITS. Next table shows this script:

```

import os
from ngamsLib import ngamsPlugInApi
from ngamsLib.ngamsCore import warning
import pccFits.PccSimpleFitsReader as fitsapi

def is_a_FITS(fileId):
    return (fileId.lower().endswith('.fits') )

def is_a_FITS(fileId):
    return (fileId.lower().endswith('.fits') )

def ngams_Filter_FITS(srvObj,

```

```

        plugInPars,
        filename,
        fileId,
        fileVersion = -1,
        reqPropsObj = None):

    """
    srvObj:      Reference to NG/AMS Server Object (ngamsServer).

    plugInPars:  Parameters to take into account for the plug-in
                 execution (string).

    fileId:      File ID for file to test (string).

    filename:    Filename of (complete) (string).

    fileVersion: Version of file to test (integer).

    reqPropsObj: NG/AMS request properties object (ngamsReqProps).

    Returns:    0 if the file does not match, 1 if it matches the
                 conditions (integer/0|1).

    """
    if (not is_a_FITS(fileId)):
        return 0
    else:
        return 1

```

Figure 2. Filter Plug-In to create a subscription that only will retrieve data files with the format FITS

This script was placed in a specific folder of the NGAS directory tree (<NGAS_HOME>/src/ngamsPlugIns/ngamsPlugIns/) and installed (cd <NGAS_HOME>/src/ngamsPlugIns & python setup.py install.)

After that, a new subscription was created using this filter:

```

curl \
'http://localhost:7777/SUBSCRIBE?priority=1&url=http://localhost:7778/OARCHIVE&subscr\_id=7777\_to\_7778\_Ffiltering&filter\_plug\_in=ngams\_Filter\_FITS&concurrent\_threads=3&start\_date=2016-09-13T00:00:00.000'

```

One thing that we wanted to do but could not find a way of doing in the version of NGAS that was experimented with, was to be able to queue large numbers of outstanding data transfer requests, while controlling how many transfers were active at any time. We do want this functionality in for SKA so that we can keep the most urgent transfers getting a large amount of the available network bandwidth and any time.

5. Review of the Apache Object Oriented Database Technology

5.1 Apache OODT

OODT is a framework for building distributed science data management environments, that has been applied to build data management systems in various projects, including Joint Polar Satellite

Systems (JPSS) project [RD7]. The framework provides components useful in two areas: a) Data Processing and Computation and b) Information Integration [RD8] .

The **OODT Catalog and Archive Service (OODT CAS)** is the main component of the framework providing for data processing functions. In particular, it groups tools for :

- Cataloging data (CAS File Manager); metadata extractors, data validators, etc.
- Archiving data; support for different data and metadata backend stores, (which may be located on remote sites).
- Processing data (CAS Workflow Manager and CAS Resource Manager); services for managing jobs that operate on ingested data, allocating them into computational resources.

In addition, the CAS crawler framework can be used to inspect a folder, read metadata from files, identify which files should be ingested according to a set of conditions, and ingest the selected files and their metadata into the CAS File Manager service. The identification and extraction of metadata can be further customized using extension points as the Metadata Extractors. The CAS crawler can retrieve the files from the local file system or from the CAS Push and Pull component, which is a service able to download/receive remote content through different protocols (e.g. FTP).

The **OODT Grid Services** is the main component that provides useful tools to federate an heterogeneous network of data providers. For example the Product Server uses different Handler connectors to communicate with heterogeneous data stores, providing hence an homogeneous access to those data stores.

5.2 OODT Services for SDP Data Transfer Service

5.2.1 OODT CAS File Manager

The OODT File Manager is capable of storing data on different back-end stores (from a folder to a local disk through a NFS partition or a S3 storage) and they can be located remotely.

This functionality could be used to implement the SDP Data Transfer Service; connecting the File Manager with the SRCs' data stores, so that when the File Manager stores the data products, what it would be doing actually is to delivery the data products to the SRCs. The OODT Crawler framework could be used to select/filter which data products should be delivered (i.e. ingested to the File Manager so it would perform the file delivery).

However, it is not clear if in this case the performance will be good given that the SRCs stores will be remotely located and in some cases connected through high latency network.

A team from the SDP consortium performed a benchmarking for OODT [RD8]. This team also stated that *although the File manager has the capability to archive a file to a remote location, it was not designed for this purpose*. Even though they made some tests in order to evaluate the File Manager performance on archiving files remotely. They concluded that even using the optimal chunk size for transferring files, the performance was not good, getting very low transfer rate (6MB/s

approximately). In conclusion it seems this component does not suit the needs of the SDP Data Transfer Service.

5.2.2 OODT CAS Workflow Manager

The OODT CAS component (File manager + Workflow manager + Resource Manager) may be used to implement the SDP Data Transfer service in this way:

- Upon the arrival of new data and metadata that fulfill the conditions to be delivered to the an specific SRCs, the File Manager would trigger the execution of a workflow.
- This workflow would consist of a job for transferring the specific data product.
- The resource manager would allocate workflow jobs on computational resources according to the workflow priority, thereby applying a data distribution policy based on priorities.

5.2.3 OODT PushPull Service

OODT PushPull component framework provides a client architecture for accessing an array of remote resources, either by downloading data or by accepting a delivery of remote data. As described before, this component is used in conjunction with the OODT Crawler framework: the PushPull service gets remote files and gives them to the Crawler, which selects the files to be ingested to the File Manager.

Given that this service only allows to download data, it can not be used by the SDP to send data products to the SRCs, but the SRCs could use this service to retrieve (download) data from the SDP. In this case, the Crawler functionality for selecting files based on metadata is not useful since this selection should be performed before sending the files.

The previously mentioned SDP team also made some test for evaluating the performance of the OODT PushPull service [RD8].

6. Review of the File Transfer Service technology

6.1 Description

The File Transfer Service (FTS) [RD9] was originally developed at CERN and is used for managing low-level data transfers for the Worldwide LHC Computing Grid (LHCG), thus providing validation of its capability to perform at scale. It provides a centralised solution for managing transfers, with a scalable pool of transfer nodes being used to manage requests.

The service aims to provide a highly flexible solution for handling transfers. The servers running FTS are accessible through a variety of interfaces, including a REST API, a graphical web interface and bindings in Python. They are also able to handle transfers using a variety of protocols - supporting any for which a GFAL2 plugin is available, including GridFTP, xroot, SRM, webdav/http, S3 - see FTS3 documentation in [RD9]. FTS can handle third-party transfers, wherein the FTS server manages the transfer but the data flows directly from source to destination. It can also handle transfers for protocols lacking this support by relaying the traffic through the FTS node as an intermediary, or it

can arrange multihop transfers where data is transferred using other servers as relay points from source to destination.

It is important to ensure that transferred data makes it in its entirety to the destination node and is not corrupted. FTS will monitor the state of transfers and automatically retry them (based on configured limits) if they fail to complete successfully at first. In addition to any protection against corruption built into the transfer protocols used, FTS can also be used to verify checksums of transferred files. If a precomputed checksum is available at the time a transfer is submitted the service can verify that any transferred products on the destination server for a transfer match this checksum without having to recompute it. A precomputed checksum can also be automatically used on the source server for a particular transfer to ensure that no error crept in while

There is an optimizer built into FTS, which automatically manages the number of connections on paths to try to maximize throughput while avoiding overloading the link. A user wishing to transfer a large amount of data can then submit it all for transfer at one time, with FTS ensuring that the link is kept busy.

6.2 Use with MeerKAT

The MeerKAT precursor is serving not only as a precursor for the telescope itself but also has provided an opportunity to gain experience working with tools that might later be useful in building a transfer system workable for the SKA at large.

In this MeerKAT context FTS was incorporated into transfer scheduling prototypes and used for shipping data from the MeerMAT archive to other locations for processing, including to the Inter-university Institute for Data-Intensive Astronomy (IDIA) and to ASTRON. It has been tested in that context as the transfer component of system which enabled data to be staged from a central archive to a transfer node in the data centre housing the archive before being transferred. Transfers were then done by FTS with the checksumming also being taken care of by FTS. The service was modified slightly to better fit the MeerKAT use case for production work, but the experience has served to validate the potential use of FTS for transfers in the SKA.

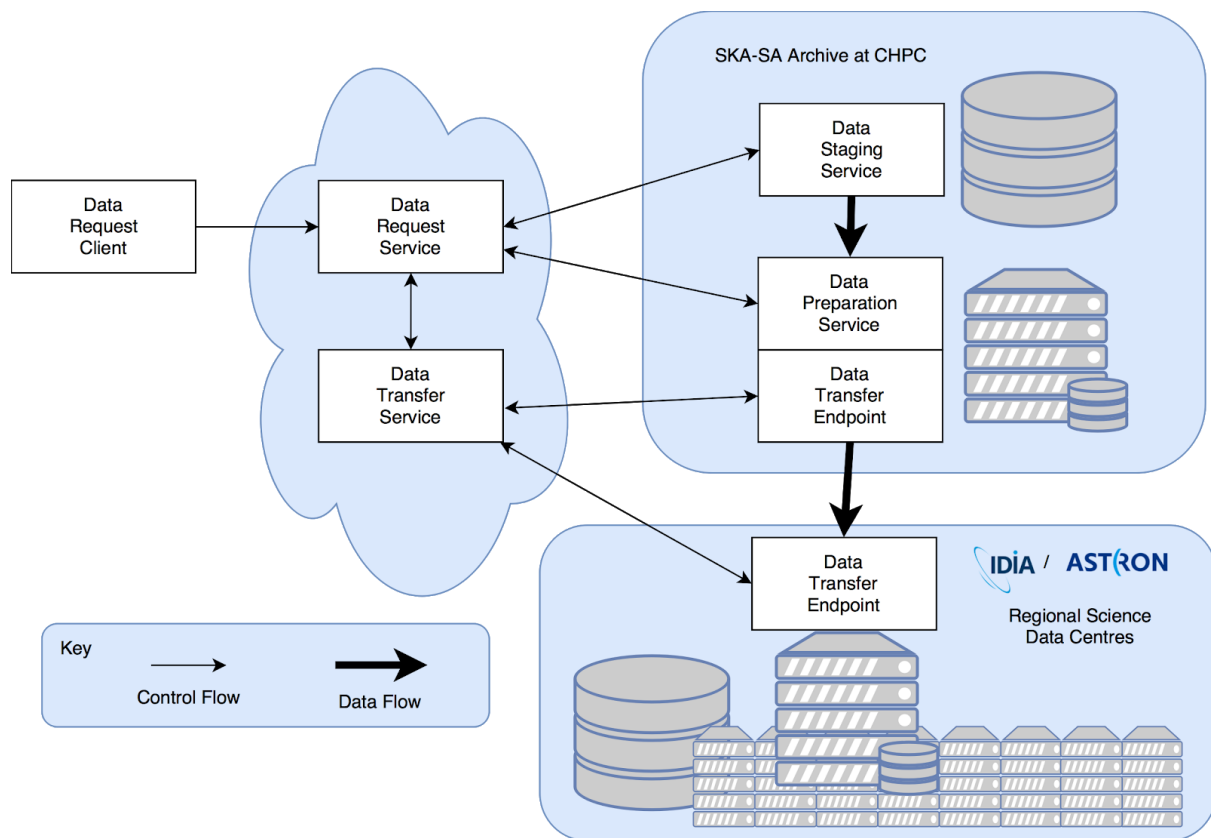


Figure 3. Overview of the MeerKAT transfer process.

6.3 Performance testing

As mentioned in Section #.1, FTS is able to support a wide variety of data transfer tools. In our case we tested this, using GridFTP servers to conduct transfers between the MeerKAT DTN based in the Centre for High Performance Computing data centre in Cape Town and the IDIA system at the University of Cape Town, well as between IDIA and several servers in the Netherlands. It was demonstrated that more than 90% of the 10G link capacity for our data transfer node could be utilized but that long-distance transfers - i.e., connections with a high bandwidth-delay product, required a large number of parallel transfer connections for this to happen.

7. IVOA Data Discovery and Access Services and Tools

We performed an assessment of current IVOA services based on the the same criteria used to assess data distribution tools. The primary IVOA discovery and access protocols are:

- **Table Access Protocol¹ (TAP):** Defines a service protocol for accessing general table data, including astronomical catalogs as well as general database tables. Access is provided for both database and table metadata as well as for actual table data. This protocol includes support for multiple query languages, including queries specified using the Astronomical Data Query Language² (ADQL) within an integrated interface. It also includes support for both synchronous and asynchronous queries. Special support is provided for spatially

¹ <http://www.ivoa.net/documents/TAP/>

² <http://www.ivoa.net/documents/ADQL/>

indexed queries using the spatial extensions in ADQL. A multi-position query capability permits queries against an arbitrarily large list of astronomical targets, providing a simple spatial cross-matching capability.

- Simple Image Access (SIA) Protocol³: Provides capabilities for the discovery, description, access, and retrieval of multi-dimensional image datasets, including 2-D images as well as datacubes of three or more dimensions. SIA data discovery is based on the ObsCore Data Model⁴ (ObsCoreDM), which primarily describes data products by the physical axes (spatial, spectral, time, and polarization). SIA provides capabilities for image discovery and access. Data discovery and metadata access (using ObsCoreDM) are defined here. SIA also allows for direct access to retrieval.
- Simple Spectral Access (SSA) Protocol⁵: Defines a uniform interface to discover and access one dimensional spectra. SSA is based on a more general data model capable of describing most tabular spectrophotometric data, including time series and spectral energy distributions (SEDs) as well as 1-D spectra; however the scope of the SSA interface as specified in this document is limited to simple 1-D spectra, including simple aggregations of 1-D spectra. Spectrum datasets may conform to a standard data model defined by SSA, or may be native spectra with custom project-defined content. Spectra may be returned in any of a number of standard data formats.

Other IVOA discovery and access protocols such as Simple Cone Search (SCS)⁶, DataLink⁷ and Server-Side Operations for Data Access⁸ (SODA) are subsets or infrastructure for the above services. These services were not assessed as it was deemed that the many implementations already in use de-risk their use within SKA.

A table containing the assessment of the main IVOA discovery and access services is in Appendix B. There was no need to prototype these services specifically for the SDP since they are already used in many data centres for accessing large catalogues and data collections.

We also performed an analysis of the sizing of the Science Data Product Catalogue, considering whether existing COTS RDBMs can handle the derived requirements.

8. Catalogue Sizing

8.1 Estimation of the Science Data Product Catalogue growth

Assuming the Science Data Product Catalogue (SDPC) will contain an entry for each Observatory Data Product (ODP) generated at the SDP (both SDP sites), an estimation of the SDPC growth can be calculated from the amount of ODPs that are expected to be generated each year and the size of metadata to be included in the SDPC per each ODP.

³ <http://www.ivoa.net/documents/SIA/>

⁴ <http://www.ivoa.net/documents/ObsCore/>

⁵ <http://www.ivoa.net/documents/SSA/>

⁶ <http://www.ivoa.net/documents/latest/ConeSearch.html>

⁷ <http://www.ivoa.net/documents/DataLink/>

⁸ <http://www.ivoa.net/documents/SODA/>

8.1.1 Estimation of the amount of ODPs generated per year

For the following calculations, we have considered the [System Sizing document](#) [RD10], which calculates the growth rate for the SKA archive, and the size of the data products for High Priority Science Objective (HPSO) experiments using the example observing-schedule proposed in the [SKA 5-year in the life of the SKA](#).

The System Sizing document [RD10] separates the ODPs into those products which are *imaging* products (i.e., images or visibility grids) and those which are not, the *non-imaging* products.

We can estimate the number of imaging products generated per year, taking the following assumptions:

- The observing time is divided into 6-hour blocks and each one is devoted to a single HPSO
- If the observing time per pointing (T_{point}) is less than 6 hours, then a set of image products is generated every T_{point} . If T_{point} is more than 6 hours, then a set is generated every 6 hours
- For HPSOs on Low that use more than one beam per station, there will be N_{beam} sets of image products every T_{image}

Table 1 lists the existing HPSOs showing their code and the telescope to be used, according to the [System Sizing document](#) [RD10]

MID	LOW
U.HPSO-4a Pulsar Search MID SPF1	U.HPSO-1 EOR Imaging
U.HPSO-4b Pulsar Search MID SPF2	U.HPSO-2a EOR imaging + power spectrum
U.HPSO-5a Pulsar Timing MID SPF2	U.HPSO-2b EOR imaging + power spectrum
U.HPSO-5b Pulsar Timing MID SPF3	U.HPSO-4c Pulsar Search LOW
U.HPSO-13 Hi Kinematics and Morphology	U.HPSO-5c Pulsar Timing LOW
U.HPSO-14 Hi MID	
U.HPSO-15 Studies of the ISM in our Galaxy	
U.HPSO-18 Transients MID	
U.HPSO-22 Cradle of Life MID Band 5	
U.HPSO-27 All Sky Magnetism	
U.HPSO-37a Continuum Survey MID band 2	
U.HPSO-37b Continuum Survey MID band 2 (deep)	

U.HPSO-37c Continuum survey, band 2 wide	
U.HPSO-38a Continuum Survey MID band 5	
U.HPSO-38b Continuum Survey MID band 5	

Table 1: HPSOs and concomitant telescope

Table 2 shows the factors involved in the calculation of the number of imaging data products generated per year, according to the previous assumptions.

HPSOs	Total hours	N. pointing centres	Fraction of time	Hours per year	T_point	Total of DPs	DPs/ Year rate
1	5000	5	0,16	1401,6	1000,0	834	234
2a	5000	50	0,16	1401,6	100,0	834	234
2b	5000	500	0,16	1401,6	10,0	834	234
13	5000	5	0,07	613,2	1000,0	834	103
14	2000	10	0,03	262,8	200,0	334	44
15	12600	2842	0,19	1664,4	4,4	2842	376
22	6000	10	0,09	788,4	600,0	1000	132
27 (and 33)	10000	81579	0,15	1314	0,1	81579	10720
37c	10000	2632	0,15	1314	3,8	2632	346
37a	2000	21	0,03	262,8	95,2	334	44
37b	2000	1	0,03	262,8	2000,0	334	44
38a	1000	61	0,01	87,6	16,4	167	15
38b	1000	1	0,01	87,6	1000,0	167	15
Total						92725	12541

Table 2. Factors involved in the calculation of the number of imaging data products generated per year.

In this table the 1st, 2nd and 3rd columns correspond to the HPSO number, total number of hours on the telescope allocated to the corresponding HPSO and number of pointing centres required. The 4th column is the time fraction on this HPSO. The 5th column shows the number of hours that would be dedicated to the corresponding HPSO in one year, assuming 100% observing efficiency, as in the "A model schedule for five years of SKA1 observations" J Wagg et. al. The 6th column shows the total number of pointings for the corresponding HPSO. The 7th column shows the total number of DPs that would be generated for this HPSO. The 8th column shows an estimation of the amount of DPs

generated by the corresponding HPSO, assuming that each year the HPSO gets its corresponding fraction of time, regardless of whether the total amount of hours for this HPSO has been reached.

Concerning the number of data products generated by the non-imaging pipelines we should take into account the following considerations, included in the document [SDP Memo 42: Data Model Summary for Pulsar/Transient Search & Timing \(March, 2018\)](#) :

- An SKA sub-array comprising between 1-16 observing beams, is allocated for each pulsar timing mode scan. Given a timing target/group of targets, a single scan will observe for between 10-1800 seconds (user configurable), producing a time-frequency-polarisation data product.
- Pulsar/transient search mode outputs a data product per beam and anywhere from 1-1500 observing beams may be used for SKA1-Mid. In SKA-Low each scan will contain 500 beams

Table 3 shows the factors involved in the calculation of the number of non-imaging data products generated per year, according to the previous considerations:

HPSO	Total hours	Sec/Scan		Total scans		DP/scan		Total DPs		
		min	max	max	min	Max	Min	max	min	Max scan & Min DP per scan
4a	800	180	1800	16000	1600	1500	1	24000000	1600	16000
4b	2400	180	1800	48000	4800	1500	1	72000000	4800	48000
4c	12750	180	1800	255000	25500	500	1	127500000	25500	255000
5a	1600	10	1800	576000	3200	16	1	9216000	3200	576000
5b	1600	10	1800	576000	3200	16	1	9216000	3200	576000
5c	4300	10	1800	1548000	8600	16	1	24768000	8600	1548000
18	10000	180	1800	200000	20000	1	1	200000	20000	20000
Totals								266900000	66900	3219000

Table 3. Factors involved in the calculation of the number of non-imaging data products generated per year.

In this table the 1st and 2nd columns correspond to the HPSO number and total number of hours on the telescope that the HPSO requires. The 3rd column shows the minimum and the maximum number of seconds to complete one scan. The 4th columns shows the results of calculating the number of scans for each HPSO depending on the total amount of hours allocated to the HPSO and the seconds per scan column. The 5th columns shows the number of DPs produced in each scan, which depends on the maximum and minimum number of beams used in a scan. The column titled "Total DPs" shows the number of DPs generated in different cases: in case the observation uses the maximum number of scans and the maximum number of beams (column 'max'); in case the

observation uses the minimum number of scans and the minimum number of beams (column 'min') ; in case the observation uses maximum number of scans and the minimum number of beams (column 'Max scan & Min DP per scan').

Concerning the HPSO-18, the maximum number of DPs is actually estimated as one per beam per second of observing, which would mean that for SKA1-Mid the maximum number of DPs generated per scan (5th column in previous table) would be 1500x1800. These DPs will be grouped in a transient catalogue, which will be a single DP as far as the Science Data Catalogue is concerned, so we assume here just 1 DP per scan will be produced. In addition, the number of observing hours allocated for this experiment is out of date, since at the time of writing current SDP pipelines design plans that pulsar and single-pulse search will be done together for the same amount of time. Taking this into account the number of observing hours for HPSO-19 would be 15950. However this would increment less than 1% the total number of DP/year (119000).

We can estimate the amount of non-imaging DP generated per year by dividing the total DPs in the previous table (columns 9th,10th,11th) by 5. Notice this method is different than the method for estimating the number of imaging DPs. If we consider the case in which the observation uses the maximum number of scans and the maximum number of beams (column 9th), the result will be **53380000** DPs generated per year.

Estimation of the size of a single entry in the Science Data Product Catalogue

The attributes included in the SDPC depends on the data model used. For our estimations we have taken into account the work [carried out in DATA group mapping different data models](#). This spreadsheet shows a list of 142 attributes that potentially could be included in the catalogue. Next table shows the different data types considered for these 142 attributes, the number of attributes of a specific data type and an estimation of the size in bytes of these data types:

Data type	Total amount of attributes	Data type size (Bytes)
String	69	80
Long	5	11
Integer	5	6
URI	6	80
date	2	7
position	1	12

double	52	9
AstroCoordSystem	1	12
AstroCoordArea	1	12

Table 4: Size of SDPC entry

According to Table 4, the size of a SDPC entry will be 6600 bytes approximately (~7KB).

This is lower than the size of metadata stored in actual catalogues. In particular, the Hubble Space Telescope and Dominion Radio Astrophysical Observatory catalogues, hosted at the Canadian Astronomy Data Centre (CADDC), contain 53KB and 91KB of metadata per observation respectively.

8.2 Conclusions

According to the previous estimations, an upper limit for the number of DPs generated per year would be approximately 53 million (**12541+ 53380000**) and the lower limit would be approximately 25 thousands (**12541 + 13380**). If we assume 90KB as the size of the metadata for each DPs in the catalogue, the growth per year of catalogue would range from 0.002 to 4.5TB.

A search for information on databases confirms that a terabyte scale databases can be managed by current COTS database systems. For example, PostgreSQL can handle table sizes of 32TB and can handle databases of [multiples of this size](#). In 2010, the web comparebusinessproducts.com published the [Top 10 Largest Databases in the World](#), which includes databases that manage more than 100 terabytes of data. According to [Begeman et al.](#) ORACLE is used as the relational database for Astro-WISE, an information system used for different astronomical archives. ORACLE provides additional functionalities that are suitable for the catalogue requirements. For example, ORACLE GoldenGate, provides an efficient synchronization between databases. In particular [a case of study shows how GoldenGate supports 1.6TB of data movement per day to read-only servers](#). This shows that current databases will be able to support SKA for many years even without the improvements that they should go through during the project.

Appendix A: Data Distribution Tool Assessment

	iRODS (CyberSKA DMS)	PhEEx	NGAS	OODT
<p>Scalability - Data demand: DELIV expects there to be potentially 3722059.23 TBytes of data downloads (TBD), by X user requests (TBD) over the life-time of the project. The system must be able to cope with the data demand in terms of data transfers and user requests.</p>	<p>Estimated 5PB managed in a single installation at end of 2012.</p>	<p>Up to 77PB replicas in production and ~450,000 transfers / day with roughly 1PB transfer / week. Scaled to 100X these sizes / rates in simulations.</p>	<p>5 PB managed for MWA across several sites. Up to 20 TB/day transferred between Perth and MIT.</p>	<p>Currently used by NASA JPL project, DESDynI, which produces 5 TB/day</p>
<p>Scalability - Data size: SKA will produce 11 data products (TBD), with an average size of 338369 TBytes (TBD), with a linear growth (TBD) per year. The system must be able to cope with future data product sizes.</p>	<p>Max filesize estimated at 16 exabytes - likely to hit limits with the underlying storage systems first</p>	<p>Average file size: ~2.5GB. Grouped into file blocks and datasets ranging from 0.1-100TB.</p>	<p>It is very unlikely that the SKA will produce single files of thousands of PB size. The management of logical containers in NGAS is supported.</p>	<p>Only a maximum of 10GB was used for benchmarking purposes on DESDynI</p>

<p>Scalability - High-speed data transfers: DELIV expects <1 Tbit/s WAN connections (TBD). The system must perform well on high-speed network connections. (An average sustained throughput of 314.74 Gbit/s would allow all data to be transferred? (TBD)).</p>	<p>Larger files split into parallel streams for transfer</p>	<p>Operates as a nearly-fully connected mesh, less hierarchically than originally expected. 10Gb/s+ links common. Total average data rate across all links ~2GB/s last year.</p>	<p>Demonstrated saturation of 40 Gbps IB connections. Saturating 10 Gbps WAN link regularly and sustained. Configurable number of parallel streams and stream priorities.</p>	<p>From JPL to AWS-east, GridFTP achieved maximum rate of 83MB/s</p>
<p>Scalability - Storage expandability: SKA will produce 3722 PBytes (TBD) over the life-time of the project. The system must be able to expand storage and potentially distribution mechanisms to cope (i.e. data replication and data striping).</p>		<p>Phedex layer tested without problems to 100x current size / rate</p>	<p>Scaling is built-in to the NGAS architecture. There are efforts under way to remove additional scalability bottlenecks.</p>	<p>Uses Amazon S3 cloud computing storage system</p>
<p>Object storage: The SKA may use an object-based file system. Software needs to be extensible or compatible with such storage systems (TBD).</p>	<p>Has an available S3 resource driver</p>	<p>? (Data service can return objects, unknown underlying compatibility)</p>	<p>NGAS is an object storage system, but typically uses file systems in the background. Removing the (not required) file system layer has been demonstrated using a S3 backend.</p>	<p>AWS S3 is an object storage system where objects contain from 1 byte to 5 terabytes of data each</p>

<p>Scalability - Architecture: DELIV expects there to be ~10 (TBD) RI's and X end users (TBD). The system must be to scale, allowing for future growth and availability/redundancy across the RI's.</p>		<p>Less hierarchical than originally expected with significant exchanges amongst the tier 2 facilities for example. 1 tier 0, 7 tier 1s and ~50 tier 2 facilities.</p>	<p>System has been distributed across the whole globe with multiple installations at ESO, ALMA, MWA and NRAO. Multi-site mirroring and subscription.</p>	<p>Reference architecture of comprising components and connector.</p>
<p>Data integrity: The system must reliably and correctly transfer (and store) data sets. Systems should employ integrity checks, such as checksums to avoid data corruption during transfer, and help minimise silent data corruption (TBD).</p>	<p>Integrity checks can be added through use of microservices</p>	<p>O</p>	<p>NGAS is using external and internal checksums (algorithms configurable) for transfer and data integrity testing.</p>	<p>According to Amazon, S3 store data with up to 99.999999999% durability, with 99.99% availability.</p>
<p>Interoperability - System: RI's and end users may use a number of different underlying Operating System platforms (TBD). The system should be compatible with mainstream Operating Systems.</p>	<p>Binaries available for various Linux-based systems. The Wiki notes building on OS X and support for Windows client tools.</p>	<p>Focused on RHEL-derivatives (CERN Scientific Linux</p>	<p>NGAS servers had been installed mainly on *NIX systems (Linux, Solaris, HP-UX, MacOSX). NGAS clients are available in Python, Java and plain C.</p>	<p>Can be used on Linux, Windows, Solaris, MacOSX. Grid components in Python while CAS are in Java.</p>

<p>Interoperability - Services: There are a number of Web Services (or REST) that may need to be interfaced with software. The system should be able to interface with such services (TBD).</p>	<p>API interfaces available for different languages using an XML-based protocol. A REST API is also available (beta?).</p>	<p>O</p>	<p>NGAS is exposing a REST interface. The clients above are based on that.</p>	
<p>Data Discovery: Data may be spread across different geographical locations (i.e. SKA RI's) (TBD). The system should provide a search facility to locate stored data, without prior knowledge of its location.</p>		<p>O</p>	<p>NGAS is a globally distributed object storage system. Users or applications access the data using unique file IDs. These IDs can be used to request files from any NGAS server participating in the same environment.</p>	<p>Done by profile server.</p>

<p>Data Models: Data may be held in a particular Data Model (i.e. CAOM). Software needs to be compatible, or extensible to interface with such models (TBD).</p>		?	<p>Through its container concept NGAS can support collections. However, it does not get directly involved with logical relations between data objects directly. Through plugins it is extensible to implement something like this.</p>	
<p>Management: DELIV expects that RI's and other institutes will have their own admin teams. The system will need to allow for some communal management of data, between the SKA and RI's (TBD).</p>		O/? (Web-based admin monitoring interface)	<p>NGAS is being used successfully in such environments already (e.g. ALMA).</p>	
<p>Resource Management: SKA.OAL will use standards-based resource management (i.e. Openstack). Software will need to be compatible with such resource management platforms (TBD).</p>		?	<p>Deployments on EC2 are standard.</p>	<p>Globus toolkit</p>

<p>Fault tolerance: Network faults are inevitable and software needs to avoid a single point of failure. Data transfers must also have the ability to resume after failure has been rectified (TBD).</p>		<p>? (Close to a mesh network as currently operating. However it seems to operate using a central blackboard built using a distributed Oracle database).</p>	<p>Transfers are re-tried for a configurable number of times and then placed in a back-log. Even in the case of sudden black-outs the system will pick-up where it stopped.</p>	<p>Done by profile server</p>
<p>Location aware: Software that is aware of an end users geographical location can use the closest resources, while potentially providing better performance (TBD).</p>		<p>X (Has some routing capability. Use of ETS3 for file transfers enables alternate sources and destinations for files to be specified.)</p>	<p>NGAS is using a simple, but efficient location based algorithm to find the closest copy of a data object.</p>	<p>Uses a Profile-, Product- and Query server to locate data resources.</p>
<p>Support and maintenance: If projects are no longer actively in use, or maintained, then a systems sustainability to support new research is diminished. Projects that are no longer supported or maintained are not the focus of our review (TBD).</p>		<p>O</p>	<p>In active use at ESO, ALMA, NRAO, MWA, ASKAP holding many PBs of data in hundreds of millions of files.</p>	<p>Used by DESDynI</p>

<p>Software dependencies: Software that has few dependencies and can be easily deployed, requires less effort to use. It is important to understand the number of dependences required for software to run, including software libraries and network connectivity (TBD).</p>		<p>? (Globus?)</p>	<p>NGAS can be installed from a single stand-alone tar file. The list of standard system level libraries to be installed is very short and most of them are installed by default (e.g. gcc, make...). NGAS can also be installed in parallel on multiple machines (cluster) from a third machine.</p>	<p>Globus</p>
<p>Open and standard protocols: Open standards that are publicly available specifications allow for better compatibility and interoperability. The use of standard transport protocols such as TCP, UDP and HTTP for transmissions is essential to ensure use at a global scale (TBD).</p>		<p>O (Phedex built to be transfer protocol independent, though FTS assumed for CMS)</p>	<p>HTTP is default, UDP (UDT) and gridftp support has been implemented as well. Standard FTP data pushing is supported.</p>	<p>FTP, SCP, bbFTP, GridFTP, UDT are supported.</p>

<p>Open source license: Open Source compatible software licensing allows for scientific communities to modify, or debug software to meet their needs and increases its sustainability. Thus, it is important to determine what type of licensing a software package is released under (TBD).</p>	<p>Q</p>	<p>O (GPL v2)</p>	<p>LGPL</p>	<p>COBRA</p>
<p>Open standard security: Standard security mechanisms are required to promote interoperability with third party services, such as Transport Layer Security (TLS), or Secure Sockets Layer (SSL), or OAuth (TBD).</p>		<p>O (Grid-based)</p>	<p>Switching to SSL possible, basic authentication and authorisation. More advanced integration through plugins.</p>	<p>Grid-based</p>

Note:

In the above table an 'O' in the cell means the tool or software does have a feature, inline with the criteria's description the software lacks the functionality:

In the above table an '?' in the cell means XXX,

In the above table an 'X' in the cell means that a Selection Criteria is not applicable to a particular Function, a '?' indicates some doubts about if the tool/software supports the Selection criteriaXXX

Appendix B: Assessment of IVOA Data Discovery and Access Services

	Table Access Protocol/Astronomy Data Query Language (CADC)	Simple Image Access Protocol (CADC)	Simple Spectrum Access Protocol (CADC)
<p>Scalability - metadata volume Expected number of science data products? Data discovery services should be implementable so that query performance scales well (much better than linearly) with metadata volume.</p>	<p>ADQL implementation requires an SQL-based database engine, so metadata capacity will depend on the chosen RDBMS.</p> <p>In use on GAIA and SDSS catalogs and is planned for LSST.</p>		
<p>Scalability - complex queries Data discovery services should be able to support complex queries that take considerable time (hours) to execute.</p>	<p>The TAP specification includes support for asynchronous query execution, so long running queries can be successfully completed.</p> <p>In use on GAIA and SDSS catalogs and is planned for LSST.</p>	<p>SIA uses a simple parameter-based query syntax so complex queries are not really possible. Since only synchronous query execution is supported, queries that take longer than a few tens of seconds to start returning results will likely time out.</p>	<p>Same as SIA</p>
<p>Scalability - large query results Data discovery services should be able to support queries that stream large results (limited by manageability of the result set).</p>	<p>The TAP specification includes support for synchronous query execution that stream results directly to the caller. Tests at CADC show that simple queries can stream millions of rows (tens of GB) for hours at decent (10MB/sec) speed and very reliably.</p>	<p>SIA synchronous queries can, in principle, stream an arbitrary number of results.</p>	<p>Same as SIA</p>
<p>Data demand Data discovery services should be capable of supporting multiple simultaneous users (N?) and frequent queries (M?/day).</p>	<p>RDBMS engines running on common multi-core hardware can support up to a few tens of simultaneous users. TAP asynchronous query execution can be implemented using a batch or pool of connections to limit the impact of high usage on the underlying DB.</p>	<p>Same as TAP</p>	<p>Same as TAP</p>

<p>Architecture</p> <p>It should be feasible to distribute the metadata across multiple sites, either as partial or complete copies (mirrors) and still perform global searches.</p>	<p>IVOA and CADC data models have defined serialization that enable distribution. With checksums, meta-data integrity can be validated.</p> <p>For example, CAOM data model is used for mirroring HST metadata between STScI, CADC and ESAC.</p>	Same as TAP	Same as TAP
<p>Meta-data integrity</p> <p>It should be feasible to verify the consistency and integrity of metadata.</p>	<p>Data models can be augmented with checksums.</p> <p>For example, CAOM uses checksums to verify the integrity of transferred meta-data between STScI, CADC and ESAC.</p>	Same as TAP	Same as TAP
<p>Data Models</p> <p>Products may be described using one or more data models (e.g. CAOM, IVOA data models). The system should be capable of supporting multiple data model descriptions of the same products where applicable.</p>	<p>TAP services can expose any data models as long as they can be expressed using a relational schema. The CADC exposes the CAOM-2.0 data model and support IVOA models like ObsCore and the implicit SIAv1 and SIAv2 data models using views.</p>	SIA services provide discovery and access to multidimensional images in SIAv2 and to 2D images in SIAv1 using an implicit image model or the ObsCore data model.	SSA services provide access to spectral data using a single data model (the IVOA SpectrumDM)
<p>Interoperability</p> <p>Data discovery client tools should be usable on mainstream operating systems.</p>	<p>TAP provides a RESTful API that can be used from any client operating system. There are numerous cross-platform clients available.</p>	Same as TAP	Same as TAP
<p>Data Discovery</p> <p>Data and metadata may be distributed across different geographical locations (i.e. SKA RC's) (TBD). The system should provide the ability to discovery data independently of storage architecture.</p>	<p>IVOA services such as TAP enable different backend architectures for databases and storage.</p>	Same as TAP	Same as TAP
<p>Fault tolerance</p> <p>Network faults are inevitable and software needs to avoid a single point of failure.</p>	<p>The CADC runs multiple load-balanced TAP services that share a common backend DB; we have prototyped using multiple backend DBs (full copy) to improve scale performance and robustness against such outages.</p>	Same as TAP	Same as TAP

<p>Support and maintenance If projects are no longer actively in use, or maintained, then a systems sustainability to support new research is diminished. Projects that are no longer supported or maintained are not the focus of our review (TBD).</p>	<p>The OpenCADC software for implementing TAP web services is actively developed and maintained and will remain so since it is a core component of CADC operations.</p>	<p>Same as TAP</p>	<p>Same as TAP</p>
<p>Software dependencies Software that has few dependencies and can be easily deployed, requires less effort to use. It is important to understand the number of dependences required for software to run, including software libraries and network connectivity (TBD).</p>	<p>The OpenCADC software libraries require java, several actively maintained open-source java libraries, and a java application server (we use apache tomcat).</p>	<p>Same as TAP</p>	<p>Same as TAP</p>
<p>Open and standard protocols Open standards that are publicly available specifications allow for better compatibility and interoperability. The use of standard transport protocols such as TCP, UDP and HTTP for transmissions is essential to ensure use at a global scale.</p>	<p>TAP is a RESTful web service API using http/https.</p>	<p>Same as TAP</p>	<p>Same as TAP</p>
<p>Open source license Open Source compatible software licensing allows for scientific communities to modify, or debug software to meet their needs and increases its sustainability. Thus, it is important to determine what type of licensing a software package is released under (TBD).</p>	<p>All OpenCADC software is available under the GNU GPLv3.</p>	<p>Same as TAP</p>	<p>Same as TAP</p>

<p>Open standard security</p> <p>Standard security mechanisms are required to promote interoperability with third party services, such as Transport Layer Security (TLS), or Secure Sockets Layer (SSL), or OAuth (TBD).</p>	<p>All OpenCADC software can be deployed using https (TLS), with support for IVOA standard X.509 client certificates as well as other authentication schemes. CADC uses a variety of authentication technologies in the web service layer (username/password, cookies, X509 proxy certificates) and will be adding OpenID support soon.</p>	<p>Same as TAP</p>	<p>Same as TAP</p>
---	---	--------------------	--------------------